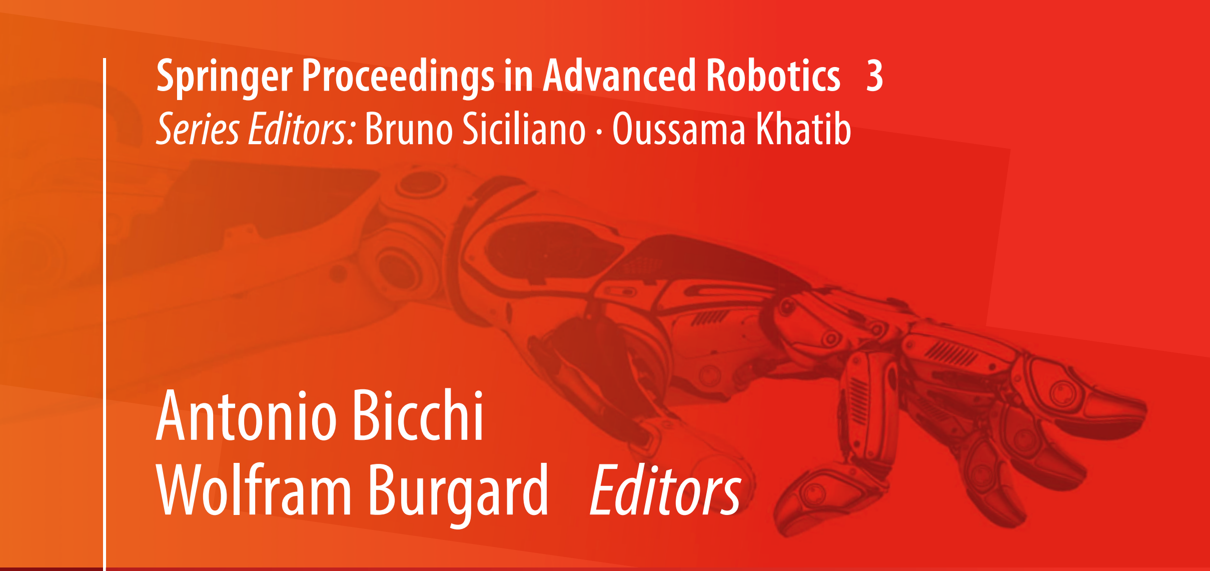
Springer Proceedings in Advanced Robotics 3
Series Editors: Bruno Siciliano · Oussama Khatib



Antonio Bicchi
Wolfram Burgard *Editors*

Robotics Research

Volume 2

spar

Springer

Springer Proceedings in Advanced Robotics

Volume 3

Series editors

Prof. Bruno Siciliano, Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy
e-mail: siciliano@unina.it
Prof. Oussama Khatib, Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305-9010, USA
e-mail: khatib@cs.stanford.edu

Editorial Advisory Board

Gianluca Antonelli, University of Cassino, Italy
Dieter Fox, University of Washington, USA
Kensuke Harada, Osaka University, Japan
M. Ani Hsieh, University of Pennsylvania, USA
Torsten Kröger, Karlsruhe Institute of Technology, Germany
Dana Kulić, University of Waterloo, Canada
Jaehung Park, Seoul National University, South Korea

The Springer Proceedings in Advanced Robotics (SPAR) publishes new developments and advances in the fields of robotics research, rapidly and informally but with a high quality.

The intent is to cover all the technical contents, applications, and multidisciplinary aspects of robotics, embedded in the fields of Mechanical Engineering, Computer Science, Electrical Engineering, Mechatronics, Control, and Life Sciences, as well as the methodologies behind them.

The publications within the "Springer Proceedings in Advanced Robotics" are primarily proceedings and post-proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. Also considered for publication are edited monographs, contributed volumes and lecture notes of exceptionally high quality and interest.

An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

More information about this series at http://www.springer.com/series/15556

Antonio Bicchi · Wolfram Burgard
Editors

Robotics Research

Volume 2

Springer

Editors

Antonio Bicchi
Istituto Italiano di Tecnologia
Genoa
Italy

and

Research Center “E.Piaggio”
University of Pisa
Pisa
Italy

Wolfram Burgard
Inst. für Informatik
Albert-Ludwigs-Universität Freiburg
Freiburg im Breisgau
Germany

ISSN 2511-1256 ISSN 2511-1264 (electronic)
Springer Proceedings in Advanced Robotics
ISBN 978-3-319-60915-7 ISBN 978-3-319-60916-4 (eBook)
DOI 10.1007/978-3-319-60916-4

Library of Congress Control Number: 2017930281

© Springer International Publishing AG 2018
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

Robots! Robots on Mars and in oceans, in hospitals and homes, in factories and schools; robots fighting fires, making goods and products, saving time and lives. Robots today are making a considerable impact from industrial manufacturing to health care, transportation, and exploration of the deep space and sea. Tomorrow, robots will become pervasive and touch upon many aspects of modern life.

The *Springer Tracts in Advanced Robotics (STAR)* was launched in 2002 with the goal of bringing to the research community the latest advances in the robotics field on the basis of their significance and quality. During the latest fifteen years, the STAR series has featured publication of both monographs and edited collections. Among the latter, the proceedings of thematic symposia devoted to excellence in robotics research, such as ISRR, ISER, FSR, and WAFR, have been regularly included in STAR.

The expansion of our field as well as the emergence of new research areas has motivated us to enlarging the pool of proceedings to be published in STAR in the last few years. This has ultimately led us to launching a sister series in parallel to STAR. The *Springer Proceedings in Advanced Robotics (SPAR)* is dedicated to the timely dissemination of the latest research results presented in selected symposia and workshops.

The twelfth edition of “Robotics Research” edited by Antonio Bicchi and Wolfram Burgard in its 8-part volume is a collection of a broad range of topics in robotics. The content of these contributions provides a wide coverage of the current state of robotics research: the advances and challenges in its theoretical foundation and technology basis, and the developments in its traditional and new emerging areas of applications. The diversity, novelty, and span of the work unfolding in these areas reveal the field’s increased maturity and expanded scope.

From its beautiful venue to its excellent program, the twelfth edition of ISRR culminates with this important reference on the current developments and new directions in the field of robotics—a true tribute to its contributors and organizers!

Stanford, USA
November 2016

Oussama Khatib
SPAR Editor

Preface

The 12th International Symposium of Robotics Research (ISRR 2015) was held from September 12–15, 2015, in Sestri Levante, Italy. The ISRR series on conferences began in 1983, and it is sponsored by the International Foundation of Robotics Research (IFRR), an independent organization comprised of top researchers around the world.

The goal of the ISRR is to bring together active, leading robotics researchers from academia, government, and industry, to assess and share their views and ideas about the state of the art of robotics and to discuss promising new avenues for future research exploration in the field of Robotics.

The choice of the location of ISRR 2015 reflects a tradition in ISRR, holding the conference in a beautiful place where the natural and cultural setting can inspire deeper and longer-sighted thoughts in the pauses of a very intense working program. Having the symposium in Italy was also meant to be suggestive of the ideal link between the most advanced robotics research with the ideas and dreams of the great engineers of the past. They, in particular those who are named “Renaissance Engineers,” thought and dreamed of realizing intelligent machines, including robots, but could not build them. Nowadays, robotics technology can make this possible. Some ideas, like the openings toward human sciences and the concept of human-centered design, are as much valid now as they were at that time.

Special emphasis in ISRR 2015 was given to the emerging frontiers, such as the fields of flying robots, soft robotics and natural machine motion, hands and haptics, multi-robot systems, cognitive robotics and learning, humanoids and legged locomotion, robot planning and navigation, and knowledge-based robots.

The goal of the ISRR Symposia is to bring together active leading robotics researchers and pioneers from academia, government, and industry to assess and share their views and ideas about the state of the art of robotics and to discuss promising new avenues for future research. Papers representing authoritative reviews of established research areas as well as papers reporting on new areas and pioneering work were sought for presentation at the symposium. In addition to the open call, a well selected number of leading researchers have been solicited to contribute by personal invitation.

A Greatest Hits track was introduced in ISRR 2015. A small number of research papers which have been selected for the most prestigious awards in the last year have been invited for presentation. This offered a unique possibility to have a synoptic view of what the robotics community considered to be the best of robotics research and put it in a larger context.

During the four-day symposium, 49 papers were presented in a single track, to cover the broad research area of robotics; two forum sessions integrated the program by facilitating group discussions. Poster sessions were also held, in a very informal interactive style. The procedure to select the papers and the participants was very strict. A number of selected leading researchers were invited to be part of the program committee, providing overview talks and participating in the review process. In addition to an open call for contributions, researchers who had made significant new contributions to robotics were invited to submit papers to a competitive review process. All papers were reviewed by the Symposium Program Committee and the International Foundation of Robotics Research (IFRR, the symposium sponsor) for final acceptance.

The symposium included visits to several beautiful sites in the area, as well as at encouraging greater participant interaction, also by stimulating cultural discussions and reflection on robotics, its historical background, and its future challenges. It furthermore included a technical tour to the Instituto Italiano di Technologia where a large variety of leading edge robotics science and systems were presented to the participants.

This book collects the papers presented at the symposium, with authoritative introductions to each section by the chairs of the corresponding sessions.

The ISRR 2015 co-chairs/editors would like to thank Floriana Sardi, for their invaluable help in the organization of the program; Monica Vasco and Simona Ventriglia for their tireless secretarial work on local organization; Nick Dring for the management of the Web site; and Abhinav Valada for helping especially in the final assembly of this book.

Genoa/Pisa, Italy
Freiburg im Breisgau, Germany
August 2016

Antonio Bicchi
Wolfram Burgard

Contents

Part I Cognitive Robotics and Learning

Part II Humanoids and Legged Locomotion

Part III Robot Planning and Navigation

Part IV Knowledge-Based Robotics and Knowledge

Part I
Cognitive Robotics and Learning

Session Summary

The *Cognitive Robotics and Learning* mini symposium was held on the third day of the conference and consisted of eight contributions. Three of them were presented as 15-minute talks, and five were presented as 5-minute spotlight talks followed by a poster session. The papers address important topics related to learning from human observation, novel representations for linking perception and action, and machine learning methods for statistical generalization, clustering, and classification. In spite of ISRR, the papers focused on discussing open research questions, challenges, and lessons learned in the respective areas.

In Chapter "Bridging the Robot Perception Gap with Mid-level Vision," Bohren and Hager presented a method to jointly recognize, segment, and estimate the position of objects in 3D. The goal was to integrate popular computer vision methods for object segmentation with constrained rigid object registration required for robotics manipulation, thereby connecting mid-level vision with the perception in 3D space. This was achieved with a combination of efficient convolutional operations through integral images with RANSAC-based pose estimation. Very comprehensive tests were performed with RGB-D data of several industrial objects.

Valada, Spinello, and Burgard introduced a new method for terrain classification in Chapter "Deep Feature Learning for Acoustics-Based Terrain Classification." Instead of using popular vision classification methods, the problem was formulated with acoustic signals obtained from inexpensive microphones. A deep classifier based on convolutional networks was then constructed to automatically learn features from spectrograms to classify the terrain. Experiments were conducted by navigating outdoors with accuracy exceeding 99% for the recognition of grass, paving, and carpet, among other surfaces. This demonstrated the potential of the method to deal with highly noise data.

Learning from demonstration was an important topic in our mini symposium, with several papers directly addressing this problem. In Chapter "Generalizing Over Uncertain Dynamics for Online Trajectory Generation", Kim, Kim, Dai, Kaelbling, and Lozano-Perez described a novel trajectory generator to address problems with variable and uncertain dynamics. The key insight was to devise an active learning method based on anomaly detection to better search the state space and guide the acquisition of more training data. Experiments were performed for an aircraft control problem and a manipulation task.

In Chapter "Inverse KKT - Learning Cost Functions of Manipulation Tasks from Demonstrations," Englert and Toussaint proposed a framework based on constraint optimization, specifically quadratic programming, to execute manipulation tasks that incorporate contacts. The authors formulated the problem of learning parametrized cost functions as a convex optimization procedure from a series of demonstrations. Once the cost function is determined, trajectories can be obtained by solving a nonlinear constrained problem. The authors successfully tested the approach on challenging tasks such as sliding objects and opening doors.

Oberlin and Tellex presented a system to automate the collection of data for novel objects with the aim of making both object detection, pose estimation and grasping easier. The paper entitled "Autonomously Acquiring Instance-Based Object Models from Experience" shows how to identify the best grasp point by performing multiple attempts of picking up an object and tracking successes and failures. This was formulated as an N-armed bandit problem and demonstrated on a stock Baxter robot with no extra sensors, and achieve excellent results in detecting, classifying, and manipulating objects.

In Chapter "Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning," Krishnan and colleagues addressed the problem of segmenting trajectories of a set of surgical trajectories by clustering transitions between linear dynamic regimes. The new method, named transition state clustering (TSC), uses a hierarchical Dirichlet process on Gaussian mixture models on transition states from several demonstrations. The method automatically determines the number of segments through a series of merging and pruning steps and achieves performance approaching human experts on needle passing segments and suturing segments.

In Chapter "Robot Learning with Task-Parameterized Generative Models," Calinon discusses the generalization capability of task-parameterized Gaussian mixture models in robot learning from demonstration and the application of such models for movement generation in robots with high number of degrees of freedom. The paper shows that task parameters can be represented as affine transformation which can be used with different statistical encoding strategies, including standard mixture models and subspace clustering, and is able to handle task constraints in task space and joint space as well as constraint priorities. The approach was demonstrated in a series of problems in configuration and operational spaces, tested in simulation and on a Baxter robot.

In modeling objects as aspect transition graphs to support manipulation, Ku, Learned-Miller, and Grupen introduced an image-based object model that is able to categorize an object into a subset of aspects based on interactions instead of only on visual appearance. The resulting representation of an object model as aspect transition graph combines distinctive object views with the information of how actions change viewpoints or the state on the object. Further, an image-based visual servoing algorithm is presented that works with the object model. The novel object model and visual servoing algorithm are demonstrated on a tool grasping task on the Robonout 2 simulator.

Bridging the Robot Perception Gap with Mid-Level Vision

Chi Li, Jonathan Bohren and Gregory D. Hager

1 Introduction

Despite the substantial progress made in computer vision for object recognition over the past few years, practical "turn-key" use of object recognition and localization for robotic manipulation in unstructured environments remains elusive. Traditional recognition methods that have been optimized for large-scale object classification [1, 2] from contextless images simply do not provide the reliability and precision for object pose estimation necessary to support physical manipulation. In robotics, both the identity and 3D pose of an object must be highly reliable since errors can result in costly failures. Subsequently, roboticists are either forced to redesign tasks to accommodate the capabilities of the available object registration algorithms, or they need to modify the objects used in a task for easier recognition. Modifications to the objects usually involve adding easily-classifiable colors, artificial texture, or easily-recognizable artificial planar markers or marker constellations [3, 4]. Unfortunately, such modifications are often impractical and sometimes even infeasible – for example, in manufacturing and assembly applications, robotic search-and-rescue, and any operation in hazardous or extreme environments.

As with many industrial automation domains, we face an assembly task in which a robot is required to construct structures from rigid components which have no discriminative texture, as seen in Fig. 1a. The lattice structures are built out of truss-like "links" which are joined together with coupling "nodes" via gendered magnetic surfaces, as seen in Fig. 1b. While these components were originally designed for

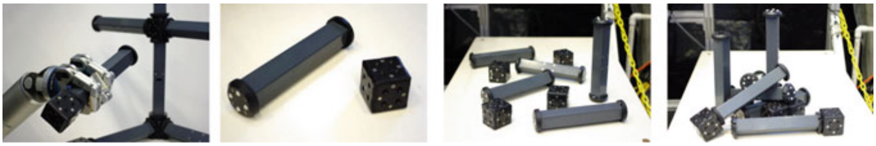
C. Li (✉) · J. Bohren · G.D. Hager
Johns Hopkins University, 3400 N. Charles St, Baltimore, MD 21218, USA
e-mail: chi_li@jhu.edu

J. Bohren
e-mail: jbo@jhu.edu

G.D. Hager
e-mail: hager@cs.jhu.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_1



(a) Robotic assembly. (b) Atomic objects. (c) Well-separated. (d) Densely cluttered.

Fig. 1 In a robotic assembly scenario (**a**), for the objects (**b**) that have little or no distinguishing texture features, most object recognition systems would rely on the objects being well-separated (**c**) and fail when objects are densely packed (**d**)

open-loop assembly via quadcopter robots [5], their mechanical properties make them ideal for autonomous and semi-autonomous [6] manipulation in assembly.

Unfortunately, many object registration algorithms [7–10] are developed to perform well only in "partially cluttered" scenes where individual objects are well-separated like that shown in Fig. 1c. Furthermore, few existing recognition algorithms are designed to reliably detect and estimate the poses of such textureless objects once they have been assembled into composite structures as shown in Fig. 1d. Even if the application allowed for it, augmenting the parts with 2D planar markers is still insufficient for precise pose estimation due to the small size of the parts and the range at which they need to be observed.

While object recognition for these dense, textureless scenes is a challenging problem, we can take advantage of the inherent constraints imposed by physical environments to find a solution. In particular, many tasks only involve manipulation of a small set of known objects, the poses of these objects evolve continuously over time [11], and often multiple views of the scene are available.

In this paper, we support this line of attack by describing the process of adapting a state-of-the-art RGBD object classification algorithm [12] to support robot manipulation. We show that by redesigning this algorithm to compute efficient semantic segmentation on cluttered scenes containing a small number of known objects, we can significantly improve a standard RANSAC-based pose estimation algorithm by exploiting semantic labels of the scene.

In the process of creating this algorithm we have introduced three critical innovations. First, we have adapted our previous state-of-the art feature-pooling-based architecture [12] to operate efficiently enough for on-line robotic use on small sets of known objects. Second, we have created and tested two variants on scene parsing, making use of a semantic segmentation provided by feature pooling, that improves the existing RANSAC-based recognition method [9] on highly cluttered and occluded scenes. Lastly, we quantitatively and qualitatively evaluate this hybrid algorithm on a new dataset including complex dense configurations of textureless objects in cluttered and occluded scenes. For this dataset, our method demonstrates dramatic improvement on pose estimation compared with the RANSAC-based algorithms without the object class segmentation.

The remainder of this paper is organized as follows. Section 2 provides a review of object instance detection and pose estimation. Section 3 introduces a new hybrid algorithm which performs scene semantic segmentation and object pose recognition. Experiments are presented in Sect. 4 and we conclude the paper in Sect. 5.

2 Related Work

Most existing 3D object recognition and pose estimation methods [8, 13–17] employ robust local feature descriptors such as SIFT [18] (2D) and SHOT/CSHOT [19] (3D) to reduce the search space of object hypotheses, combined with some constraints imposed by 3D object structures. In particular, Hough voting [8, 13, 20] has been applied to find hypotheses that preserve consistent geometric configurations of matched feature points on each model. Hand-crafted global feature descriptors which model partial views have also been examined [15, 16] to filter hypotheses. A more principled framework is proposed by [14, 21] to select the optimal subset of hypotheses yielding a solution that is globally consistent with the scene while handling the interactions among objects. Other approaches only rely on simple geometric features [9, 10, 22] for fast model-scene registration. However, this pipeline of work suffers from the limited power of local features to robustly infer accurate object poses, especially in the context of textureless objects, occlusions, foreground and background clutter and large viewpoint variations.

Another line of work exploits detection results on 2D images for pose estimation. One representative is the LINE-MOD system [10] which uses gradient templates to match sliding windows to object partial views and initialize Iterative Closest Point (ICP) for pose refinement. This template-based design does not capture fine-grained visual cues between similar objects and does not scale well to multiple object instances which occlude and / or are in close contact with each other. Furthermore, the precision of LINE-MOD's similarity measure decreases linearly in the percentage of occlusion [23]. Additionally, some unsupervised segmentation techniques [24, 25] partition the scene into different objects without any prior model knowledge. [11] updates scene models for generic shapes like boxes and balls over time. These methods are hard to generalize to segment objects with arbitrary shape.

Recently, the use of deep convolutional architectures [26–31] and the availability of huge datasets with hundreds to thousands of object classes have led to rapid and revolutionary developments in large-scale object recognition. Although these frameworks significantly boost object classification performance, accurate and efficient pose estimation still remains an unsolved problem. Our recent work [12] shows how color pooling, within a convolutional architecture, is effective for developing robust object representations which are insensitive to out-of-plane rotations. This is the foundation of the proposed method in this paper, which combines the advantages of a pose-insensitive convolutional architecture with an efficient pose estimation method [9].

2.1 RANSAC-Based Object Pose Recognition

In this section, we briefly review an efficient pose estimation algorithm originally reported in [9]. We used it as one option of object registration in our pipeline due to its efficiency and robustness to complex occlusions. The reference implementation of this algorithm is called "ObjRecRANSAC" and is available for academic use under an open-source license.[1]

ObjRecRANSAC is designed to perform fast object pose prediction using oriented point pair features $((p_i, n_i), (p_j, n_j))$ where p_i, p_j are the 3D positions of the points and n_i, n_j are their associated surface normals. In turn, a simple descriptor $f(i, j)$ is defined by:

$$f(i, j) = \begin{pmatrix} \|p_i - p_j\| \\ \angle(n_i, n_j) \\ \angle(n_i, p_j - p_i) \\ \angle(n_j, n_i - p_i) \end{pmatrix} \quad (1)$$

where $\angle(a, b)$ denotes the angle between a and b. Then a hash table is constructed for fast matching of point pairs from object models to the scene. We refer the reader to [9] for more details.

In ObjRecRANSAC, only oriented point pair features with fixed predefined distance d are used for RANSAC sampling. This prevents the algorithm from recognizing scenes composed of objects with significantly different characteristic lengths. If one object has a highly eccentric shape, it is best localized by sampling point pairs which span it's widest axis. This large pair separation, however, prevents any smaller objects in the scene from being recognized. Moreover, for objects situated in cluttered and occluded scenes, the probability of sampling point pairs from single object instances significantly decreases, which leads to the strong degradation in performance. One failure case of ObjRecRANSAC is shown in Fig. 2.

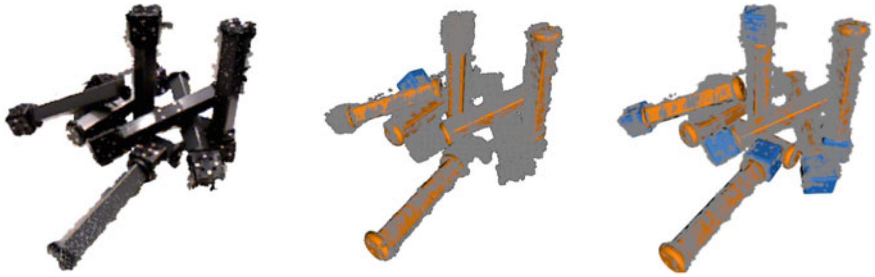


Fig. 2 The illustration of failure cases of ObjRecRANSAC. Figures from the *left* to *right* are the testing scene, estimated poses from ObjRecRANSAC and groundtruth

[1] See http://github.com/tum-mvp/ObjRecRANSAC.git for the reference implementation of [9].

From a high-level perspective, the information needed to improve the recognition accuracy in heterogeneous scenes is the object class membership. If such class membership could be determined independently from object pose, it could be used to partition the input data into independent RANSAC pipelines which are specifically optimally parameterized. Semantic segmentation techniques are well-suited to provide this crucial information.

3 Mid-Level Perception and Robust Pose Estimation

This section presents details of a two-stage algorithm in which semantic segmentation first partitions the scene and the ObjRecRANSAC, or one of its variants is applied to estimate the poses of instances within each semantic class. Figure 3 shows the flow chart for this hybrid algorithm.

3.1 Semantic Scene Segmentation Using Mid-Level Visual Cues

There are three major challenges to achieve accurate and efficient semantic segmentation. First, robust and discriminative features need to be learned to distinguish different object classes, even for those with similar textureless appearances. Second, "mid-level" object models should be produced in order to handle clutter and occlusions caused by interactions among objects. Third, the state-of-the-art recognition techniques in large-scale setting typically do not operate at the time scales consistent with robot manipulation.

Here, we develop an algorithm for semantic segmentation based on the idea of color pooling in our previous work [12], but modified to make use of integral images to speed up the color pooling for sliding windows in the image domain. This enables the algorithm to perform efficient dense feature extraction in practice. We also detail how we exploit adaptive scales of sliding windows to achieve scale invariance for

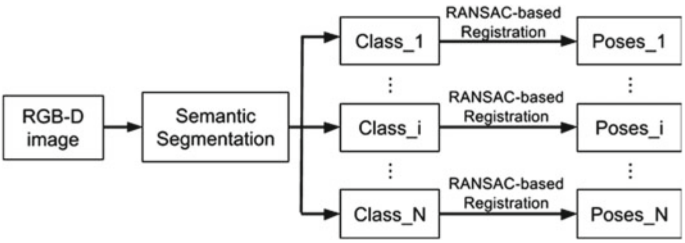

RGB-D image
Semantic Segmentation
Class_1
Class_i
Class_N
RANSAC-based Registration
RANSAC-based Registration
RANSAC-based Registration
Poses_1
Poses_i
Poses_N

Fig. 3 Overview of the hybrid algorithm for object detection and pose estimation

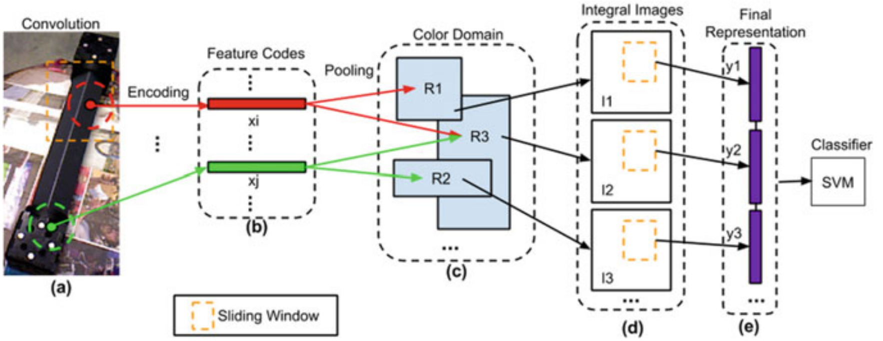


Fig. 4 Illustration of the feature extraction for semantic segmentation, including from *right* to *left* convolution of local feature codes (**a**→**b**), color pooling (**b**→**c**), integral image construction (**c**→**d**), and final feature concatenation (**d**→**e**)

dense scene classification. The overview of the entire semantic segmentation pipeline is illustrated in Fig. 4.

3.1.1 Review of Color Pooling

Pooling, which groups local filter responses within neighborhoods in a certain domain, is a key component in convolutional architectures to reduce the variability of raw input signals while preserving dominant visual characteristics. Color pooling [12] yields features which achieve superior 3D rotation invariance over pooling in the spatial domain.

First, the convolutional feature is constructed as follows. Given a point cloud[2] $P = \{p_1, \dots, p_n\}$, we compute the rotationally invariant 3D feature CSHOT [19] for each 3D point p_i. Next, the CSHOT descriptor is divided into color and depth components: f_c and f_d. Dictionaries $D = \{d_1, d_2, \dots, d_K\}$ with K filters for each component are learned via hierarchical K-means for randomly sampled CSHOT features across different object classes. Finally, each CSHOT component f is transformed into a feature code $\mu = \{\mu_1, \dots, \mu_K\}$ by the hard-assignment encoder[3] and the final local feature $x_i = [\mu_c, \mu_d]$ for each p_i are constructed by concatenating the two transformed CSHOT codes μ_c and μ_d. The hard assignment coding is defined as follows:

$$\mu_j = \begin{cases} 1 & : d_j \in \mathcal{N}_1(f) \\ 0 & : d_j \notin \mathcal{N}_1(f) \end{cases} \tag{2}$$

[2] For efficiency purpose, raw point clouds are downsampled via octree with the leaf size as 0.005 m.

[3] We replace the soft encoder used in [12] with the hard encoder to speed up the computation.

where $\mathscr{N}_1(f)$ returns the set of the first nearest neighbor of the CSHOT component x in dictionary D. The convolution and encoding process is shown in stage (a→b) in Fig. 4.

Next, we pool features over LAB color space S because it achieves better recognition performance than both RGB and HSV based on our experiences reported in [12]. For a set of pooling regions $R = \{R_1, \ldots, R_m\}$ where each R_j $(1 \leq j \leq m)$ occupies certain subspaces in S, the pooled feature vector y_j associated with R_j is computed by sum pooling over the local feature codes $\{x_i\}$:

$$y_j = \sum_i x_i \cdot \mathbf{1}(c_i \in R_j) \tag{3}$$

where $\mathbf{1}(.)$ is the indicator function to decide if the color signature c_i (a LAB value) at p_i falls in R_j. We choose sum pooling instead of max pooling (used in [12]) because it is computationally expensive to compute maximum values over the integral structure. Lastly, each y_j is L2-normalized in order to suppress the noise [26] and the pooled feature vector $Y = [y_1, \ldots, y_m]$ is constructed as the final representation for the given point cloud.

3.1.2 Efficient Computation via Integral Images

Integral images are often used for fast feature computation in real-time object detection such as [32]. We build the integral image structure for fast dense feature extraction. To do so, we first project each scene point cloud onto a 2D image using the camera's intrinsic parameters.[4] Suppose we obtain the local feature vector x_i in Sect. 3.1.1 for each p_i. For each pooling region R_j, the corresponding integral image I_j is constructed as follows:

$$I_j(u, v) = \sum_i x_i \cdot \mathbf{1}(c_i \in R_j \wedge u_i \leq u \wedge v_i \leq v) \tag{4}$$

where (u, v) is the 2D coordinate of integral image and (u_i, v_i) is the projected 2D location of 3D point p_i in 3D point cloud.

The total complexity to construct all integral images is $O((K_d + K_c)WHm)$ where K_d and K_c are the number of codewords for color and depth components, respectively, and W and H are the width and height of integral images, respectively. Thus, with I_j, the pooled feature $y_j(B)$ for sliding window $B = \{u_l, v_l, u_r, v_r\}$ can be computed in $O(1)$:

$$y_j(B) = I_j(u_l, v_l) + I_j(u_r, v_r) - I_j(u_l, v_r) - I_j(u_r, v_l) \tag{5}$$

[4] In our implementation, PrimeSense Carmine 1.08 depth sensor is used. We found no difference in performance between using default camera parameters and manual calibration.

where (u_l, v_l) and (u_r, v_r) are 2D coordinates for top-left and bottom-right corners of window B on the projection of the 3D scene. Stages (c→ d) and (d→ e) in Fig. 4 show the process of integral image construction and pooled feature extraction respectively.

3.1.3 Scale Invariant Modeling for Object Parts

Modeling object partial views from complete object segments does not account for missing object parts due to occlusion and outliers from background clutter. To overcome this, we train object models based on generic object parts randomly sampled from object segments at different viewpoints. In order to achieve the scale invariance for the learned models, all sampled parts are encompassed by a predefined fixed-size 3D bounding box $\mathscr{B}$. In turn, the sliding windows extracted for testing scene adopt scales which are consistent with $\mathscr{B}$. Specifically, the scale of the ith sliding window (w_i, h_i) with center (u_i, v_i) is equal to the scale of the projected bounding box $\mathscr{B}$ onto the same location:

$$(w_i, h_i) = \frac{\widetilde{f}}{z_i}(w_{\mathscr{B}}, h_{\mathscr{B}}) \tag{6}$$

where $(w_{\mathscr{B}}, h_{\mathscr{B}})$ is the predefined size in (x,y) coordinate of $\mathscr{B}$ in 3D and z_i is the depth corresponding to (u_i, v_i). $\widetilde{f}$ is the focal length of the camera. We note that object parts here do not necessarily have specific semantic correspondences. Next, we directly train a discriminative classification model using a linear SVM over object parts with semantic labels inherited from corresponding partial views.

Given a new scene, we extract features with adaptive scales for all sliding windows on integral images. Each window is classified into one of the trained semantic classes and votes for all included 3D points. The final semantic label of each 3D point is the one with the maximum votes.

3.2 Recursive RANSAC-Based Registration for Pose Estimation

Although the semantic segmentation narrows down the space of RANSAC sampling within only a single semantic class, the ratio of inlier correspondences may be still small due to multiple adjacent or connected object instances. In this section, we introduce two recursive pipelines that improve the performance of the RANSAC-based registration algorithm detailed in Sect. 2.1 in terms of stability and the recall rate. In what follows, we denote the original ObjRecRANSAC as **B** short for Batch Matching and introduce two improved variants as **GB** and **GO**.

Greedy-Batch Matching(GB): In this approach, we run the ObjRecRANSAC recursively over the parts of the scene that have not been well explained by previous detected models. Specifically, the initial inputs to the ObjRecRANSAC are the set of

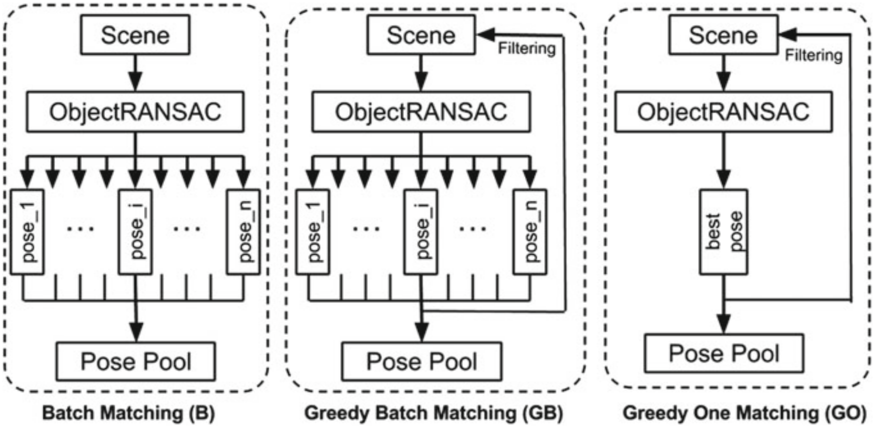


Fig. 5 Illustration of the algorithm pipelines of **B**, **GB** and **GO**

segmented points P_0 that share the same class label. At the *ith* round of recognition ($i \geq 1$), the working space P_i is constructed by removing the points in P_{i-1} that can be explained by the detected models M_{i-1} at $(i-1)th$ round:

$$P_i = \{p \mid \min_{m \in M_{i-1}} \|p - m\|_2 > T_d \ \wedge \ p \in P_{i-1}\} \tag{7}$$

where T_d is the threshold (set to 0.01m) to determine the inlier. The detected models M_{i-1} are the transformed point clouds that are uniformly sampled from full object meshes. Finally, this greedy registration pipeline stops once no more instances are detected. The final set of estimated poses is the union of all previously detected poses: $M_{final} = \cup_i M_i$.

Greedy-One Matching(GO): The **GB** approach can fail to discover some object instances because false positives in early iterations can lead to false negatives later on. In order to achieve higher precision and recall detection rates, we adopt a more conservative greedy approach in which we only choose the best detected object candidate with the highest confidence score from ObjRecRANSAC as the current detected model M_i at ith round. The rest follows the same implementation as in **GB**. The simple flow charts for **B**, **GB** and **GO** are illustrated in Fig. 5.

4 Experiments

In all our experiments, we use data collected with a PrimeSense Carmine 1.09 depth sensor. We choose 0.03 m as the radius for both normal estimation and CSHOT descriptor.[5] Depth and color components in the raw CSHOT feature are decoupled into two feature vectors. Dictionaries with 200 codewords for each component are

[5] The implementations of normal estimation and CSHOT come from PCL Library.

learned by hierarchical K-means. For the LAB pooling domain, we adopt a 4-level architecture where gridding over the entire domain at the k^{th} level is performed by equally dividing each channel of LAB into k bins. Therefore, in this 4-level architecture we have $100 = \sum_{k=1}^{4} k^3$ pooling regions. Pooled features in different levels and domains are concatenated as the final feature vector. Integral images are constructed with the size that is $\frac{1}{5}$ of the original RGB-D frame for efficiency. Sliding windows with step size of 1px are extracted on integral images. For ObjRecRANSAC, we build models separately for each object by setting the oriented point pair feature separation to 60% of the largest diameter of the corresponding object. The rest of the parameters are the same as the default ones used in [9]. Last, we capture object partial views under both fixed and random viewpoints as the training data for the SVM classifier in semantic segmentation. Specifically, three data sequences at fixed viewing angles of 30, 45 and 60 degrees as well as one random viewing sequence are captured. This follows the same procedure of data collection for JHUIT-50 dataset [12]. In each partial view, we randomly sample 30 object patches encompassed by a predefined 3D bounding box with size $w_{\mathscr{B}} = h_{\mathscr{B}} = 0.03\,\text{m}$ (see details in Sect. 3.1.3). This size is also applied to compute the scale of sliding windows on integral images for testing examples.

Next, unlike the matching function designed for LINE-MOD [10], we introduce a more flexible matching criterion to determine whether an estimated pose is correct. In the task of object manipulation, a good pose estimation needs to achieve high matching accuracy only with respect to the 3D geometry but not the surface texture. This implies that for objects with certain symmetrical structure (rotational and/or reflective), there should exist multiple pose candidates having perfect matching to the groundtruth. Thus, we design a new distance function between two estimated poses (i.e. 3D transformation in SE(3)) T_1 and T_2 for model point cloud P_M with N 3D points uniformly sampled from the full object mesh:

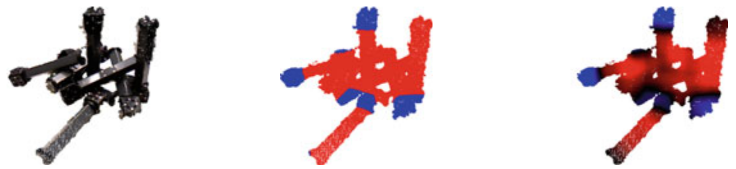
$$D(T_1, T_2; P_M) = \frac{\sum_{p_i \in P_M} \mathbf{1}(\min_{p_j \in P_M} \|T_1(p_i) - T_2(p_j)\|_2 < \delta_D)}{N} \tag{8}$$

where threshold δ_D controls the matching degree. Another threshold R_D is used to justify an estimated pose T with respect to the groudtruth T_g by the criterion: $D(T, T_g; P_M) \geq R_D$. We set $\delta_D = 0.01$ and $R_D = 0.7$ for all our experiments.

The algorithm presented in this paper is implemented in C++ and all tests are performed on a desktop with Intel Xeon CPU E5-2690 3.00 GHz.

4.1 LN-66 Dataset for Textureless Industrial Objects

We first evaluate our method on our new LN-66 dataset which contains 66 scenes with various complex configurations of the two "link" and "node" textureless objects shown in Fig. 1b. We combine the training and testing sequences (corresponding to fixed and random viewpoints) of "link" and "node" objects in JHUIT-50 [12] as the



(a) Testing Scene. (b) Semantic Labels. (c) Confidence Map.

Fig. 6 An example of semantic scene segmentation

training data so that each object has 300 training samples. We note that our algorithm can easily be applied to scenes composed of more than 2 objects by simply adding more training classes in the semantic classification stage. The LN-66 dataset and the object training data are available at http://cirl.lcsr.jhu.edu/jhu-visual-perception-datasets/. An example testing scene is shown in Fig. 6a. There are 6–10 example point clouds for each static scene from a fixed viewpoint, where each cloud is the average of ten raw RGB-D images. This gives a total of 614 testing examples across all scenes. In our dataset, the background has been removed from each example by RANSAC plane estimation and defining workspace limits in 3D space. Background subtraction can also be done with the semantic segmentation stage if object models are trained along with a background class. Therefore, the points in the remaining point cloud only belong to instances of the “link” or “node” objects. However, robust object detection and pose estimation are still challenging in such scenario due to similar appearances between objects, clutter, occlusion and sensor noise. To quantitatively analyze our method, we manually label the groundtruth object poses for each scene and propagate them to all testing examples. Then the groundtruth poses are projected onto 2D to generate the groundtruth for the semantic segmentation at each frame.

The overall segmentation accuracy is measured as the average ratio of correctly labeled 3D points versus all in a testing scene point cloud. By running the classification algorithm in Sect. 3.1 over all 614 testing frames, the average accuracy of the semantic segmentation achieves as high as 91.2%. One example of semantic scene labeling is shown in Fig. 6. The red and blue regions represent the “link” and “node” object classes, respectively. In Fig. 6c, we also show the confidence scores returned from the SVM classifier for each class. The brighter color in either red or blue indicates stronger confidence from the corresponding classifier. We could visually observe that the semantic segmentation obtains high classification accuracy.

Next, we report the means and standard deviations(std) of precision, recall and F-measure[6] of our algorithm on LN-66 in Table 1. For comparison, we run experiments for different variants of our algorithm whose names are formatted as ‘S+O’. The first entry ‘S’ indicates the degree of semantic segmentation used with three specific options ‘**NS**’, ‘**S**’ and ‘**GS**’ as *no segmentation*, *standard segmentation* (Sect. 3.1) and *groudtruth*. The second entry ‘O’ stands for the three choices of ObjRecRANSAC

[6] F-measure is a joint measurement computed by precision and recall as $\frac{2 \cdot precision \cdot recall}{precision + recall}$.

Table 1 Reported precision, recall and F-score by different methods on LN-66 dataset

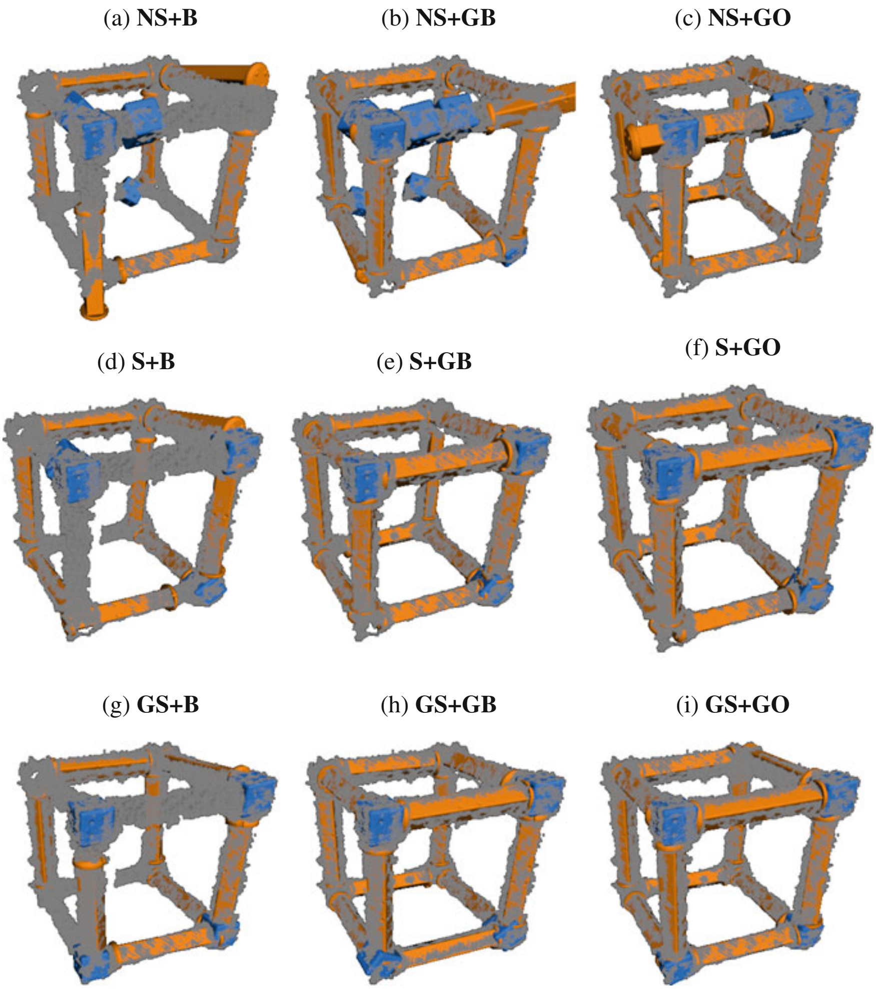
	Precision(%)	Recall(%)	F-measure
NS+B	84.47 ± 0.36	61.75 ± 0.27	71.30 ± 0.28
NS+GB	79.88 ± 0.47	79.42 ± 0.37	79.65 ± 0.40
NS+GO	88.63 ± 0.31	83.13 ± 0.32	85.80 ± 0.30
S+B	87.77 ± 0.20	81.31 ± 0.27	84.42 ± 0.22
S+GB	91.89 ± 0.24	89.27 ± 0.19	90.56 ± 0.21
S+GO	94.50 ± 0.16	91.71 ± 0.13	93.09 ± 0.12
GS+B	97.27 ± 0.06	87.03 ± 0.14	91.87 ± 0.08
GS+GB	95.29 ± 0.10	92.33 ± 0.13	93.79 ± 0.11
GS+GO	98.79 ± 0.20	94.33 ± 0.13	96.51 ± 0.16

including ‘**B**’, ‘**GB**’ and ‘**GO**’. Due to the randomized process in ObjRecRANSAC, we run 50 trials of each method over all testing data.

From Table 1, we observe that: (1) the semantic segmentation significantly improves all three RANSAC-based pose estimation methods in terms of precision/recall rates; (2) when using the segmentation computed by our algorithm, the RANSAC stage performs only $2 \sim 4\%$ behind in the final F-measure compared to using the groundtruth segmentation. (3) Both **GO** and **GB** are more accurate (higher F-measure) and stable (smaller standard deviation) than the standard ObjRecRANSAC (**B**) regardless whether they are supported by semantic labeling.

Furthermore, we show an example of comparison between different methods in Fig. 7 and more results from **S+GB** are shown in Fig. 8. In each sub-figure of Fig. 7, the gray points represent the point cloud of the testing scene. The estimated poses for the “link” and “node” objects are shown in yellow and blue meshes, respectively. We can see that methods that work on semantically segmented scenes achieve noticeable improvement over the ones without scene classification. In addition, the computed semantic segmentation yields similar results ((d), (e), (f) in Fig. 7) compared with the ground truth ((g), (b), (i) in Fig. 7), which shows the effectiveness of our semantic segmentation algorithm. Also, **GO** and **GB** outperform **B** whether or not semantic segmentation is used. From Fig. 8, we can see **S+GB** could reliably detect and estimate object poses in cluttered and occluded scenes. Finer pose refinement can be made by incorporating physical constraints between adjacent objects.

Finally, Table 2 reports the means and standard deviations of running times of all main modules in the semantic segmentation as well as **B**, **GB**, **GO** in two contexts: (**S**) and (**NS**) indicating with and without semantic segmentation, respectively. For semantic segmentation, we evaluate all three components: CSHOT extraction (**S-CSHOT**), integral image construction (**S-Int**) and classification of sliding windows (**S-Det**). From Table 2, we can see that the semantic segmentation is running efficiently compared to the overall runtime of the algorithm. Furthermore, all three sub-stages can be trivially parallelized and dramatically accelerated with GPU-based implementations. We also observe that the semantic segmentation reduces the run-



(a) **NS+B** (b) **NS+GB** (c) **NS+GO**

(d) **S+B** (e) **S+GB** (f) **S+GO**

(g) **GS+B** (h) **GS+GB** (i) **GS+GO**

Fig. 7 An example of the comparison of the estimated poses by different methods

time of **GO(NS)** by half because it decreases the number of RANSAC hypotheses in this greedy approach. For pose estimation, two proposed greedy approaches **GB** and **GO** are slower than the standard one **B** due to multiple runs of ObjRecRANSAC. Additionally, **GB** performs only slightly worse than **GO** (shown in Table 1) while being much more efficient. These times were also computed for the CPU-based implementation of ObjRecRANSAC, and did not use the GPU-accelerated implementation, which is already available under the same open-source license. The choice of these three methods in practice can be decided based on the specific performance requirements of a given application.

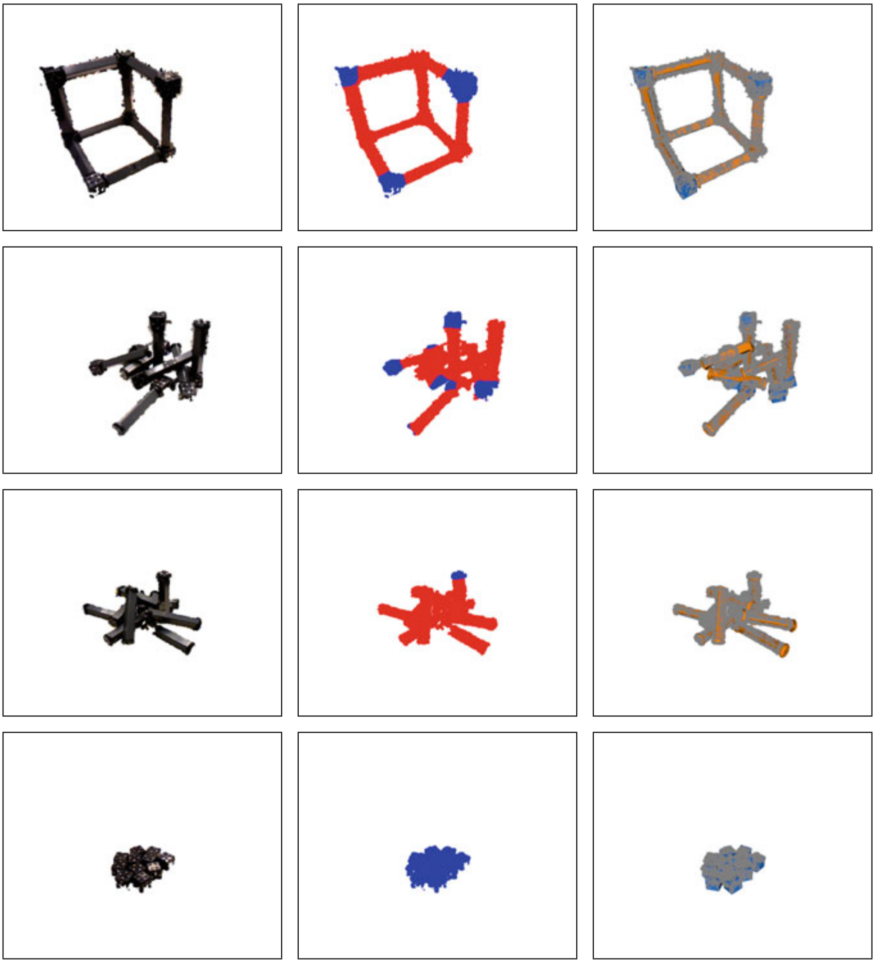


Fig. 8 Example results of **S+GB** on LN-66. The *left*, *middle* and *right* columns show the testing scenes, segmentation results and estimated poses, respectively

Table 2 Means and standard deviations of running times of different methods on LN-66 dataset

S-CSHOT	S-Int	S-Det	B(NS)	GB(NS)	GO(NS)	B(S)	GB(S)	GO(S)
0.39 ± 0.12	0.13 ± 0.03	0.31 ± 0.12	0.86 ± 0.16	1.49 ± 0.40	7.69 ± 3.43	0.85 ± 0.20	2.16 ± 0.68	4.40 ± 1.72

Although the overall runtime of the entire perception system takes more than 1 s even without the semantic segmentation, this may not be a main issue to integrate our algorithm into a real-time robotic system. First, GPU-based parallel programming techniques could significantly speed up the current implementation. Second, standard object tracking methods [7] can be initialized by our algorithm to track object poses in real time and reinitialized when they fail.

5 Conclusion

In this paper, we present a novel robot perception pipeline which applies advantages of state-of-the art large-scale recognition methods to constrained rigid object registration for robotics. Mid-level visual cues are modeled via an efficient convolutional architecture built on integral images, and in turn used for robust semantic segmentation. Additionally, two greedy approaches are introduced to further augment the RANSAC sampling procedure for pose estimation in the constrained semantic classes. This pipeline effectively bridges the gap between powerful large-scale vision techniques and task-dependent robot perception.

Although the recursive nature of our modification for RANSAC-based registration slows down the entire detection process, the better trade-off between time and accuracy would be achieved. Moreover, we could adaptively select the suitable strategy regarding the requirement of robotics systems in practice. We believe that this approach can be used with other registration algorithms which are hindered by large search spaces, and plan to investigate other such compositions in the future.

Acknowledgements This work is supported by the National Science Foundation under Grant No. NRI-1227277 and the National Aeronautics and Space Administration under Grant No. NNX12AM45H.

References

1. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: ICRA (2011)
2. Singh, A., Sha, J., Narayan, K.S., Achim, T., Abbeel, P.: BigBIRD: a large-scale 3D database of object instances. In: ICRA (2014)
3. Macias, N., Wen, J.: Vision guided robotic block stacking. In: IROS (2014)
4. Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., Barto, A.G.: Learning grounded finite-state representations from unstructured demonstrations. In: IJRR (2014)
5. Lindsey, Q., Mellinger, D., Kumar, V.: Construction with quadrotor teams. Auton. Robot. **33**(3), 323–336 (2012)
6. Bohren, J., Papazov, C., Burschka, D., Krieger, K., Parusel, S., Haddadin, S., Shepherdson, W.L., Hager, G.D., Whitcomb, L.L.: A pilot study in vision-based augmented telemanipulation for remote assembly over high-latency networks. In: ICRA (2013)
7. Pauwels, K., Ivan, V., Ros, E., Vijayakumar, S.: Real-time object pose recognition and tracking with an imprecisely calibrated moving RGB-D camera. In: IROS (2014)

8. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: efficient and robust 3D object recognition. In: CVPR (2010)
9. Papazov, C., Burschka, D.: An efficient RANSAC for 3D object recognition in noisy and occluded scenes. In: ACCV (2010)
10. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: ACCV, 2012 (2013)
11. Hager, G.D., Wegbreit, B.: Scene parsing using a prior world model. In: IJRR (2011)
12. Li, C., Reiter, A., Hager, G.D.: Beyond spatial pooling, fine-grained representation learning in multiple domains. In: CVPR (2015)
13. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3D SURF for robust three dimensional classification. In: ECCV (2010)
14. Aldoma, A., Tombari, F., Prankl, J., Richtsfeld, A., Di Stefano, L., Vincze, M.: Multimodal cue integration through Hypotheses Verification for RGB-D object recognition and 6DOF pose estimation. In: ICRA (2013)
15. Xie, Z., Singh, A., Uang, J., Narayan, K.S., Abbeel, P.: Multimodal blending for high-accuracy instance recognition. In: IROS (2013)
16. Tang, J., Miller, S., Singh, A., Abbeel, P.: A textured object recognition pipeline for color and depth image data. In: ICRA (2012)
17. Fischer, J., Bormann, R., Arbeiter, G., Verl, A.: A feature descriptor for texture-less object representation using 2D and 3D cues from RGB-D data. In: ICRA (2013)
18. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. In: IJCV (2004)
19. Tombari, F., Salti, S., Di Stefano, L.: A combined texture-shape descriptor for enhanced 3D feature matching. In: ICIP (2011)
20. Woodford, O.J., Pham, M.T., Maki, A., Perbet, F., Stenger, B.: Demisting the hough transform for 3D shape recognition and registration. In: IJCV (2014)
21. Aldoma, A., Tombari, F., Stefano, L.D., Vincze, M.: A global hypotheses verification method for 3D object recognition. In: ECCV (2012)
22. Rusu, R.B., Bradski, G., Thibaux, R., Hsu, J.: Fast 3d recognition and pose using the viewpoint feature histogram. In: IROS (2010)
23. Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., Lepetit, V.: Gradient response maps for real-time detection of textureless objects. PAMI **I**, 2012 (2012)
24. Richtsfeld, A., Morwald, T., Prankl, J., Zillich, M., Vincze, M: Segmentation of unknown objects in indoor environments. In: IROS (2012)
25. Uckermann, A., Haschke, R., Ritter, H.: Realtime 3D segmentation for human-robot interaction. In: IROS (2013)
26. Bo, L., Ren, X., Fox, D.: Unsupervised feature learning for RGB-D based object recognition. In: ISER (2013)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
28. Socher, R., Huval, B., Bhat, B., Manning, C.D., Ng, A.Y.: Convolutional-recursive deep learning for 3D object classification. In: NIPS (2012)
29. Gupta, S., Girshick, R., Arbelez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: ECCV (2014)
30. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
31. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: a deep convolutional activation feature for generic visual recognition. In: ICML (2014)
32. Viola, P., Jones, M.: Robust real-time object detection. In: IJCV (2001)

Deep Feature Learning for Acoustics-Based Terrain Classification

Abhinav Valada, Luciano Spinello and Wolfram Burgard

1 Introduction

Robots are increasingly being used for tasks in unstructured real-world environments and thus have to be able to deal with a huge variety of different terrains. As every terrain has a distinct physical property, it necessitates an appropriate navigation strategy to maximize the performance of the robot. Therefore, terrain classification is paramount to determine the corresponding trafficability. However, it is a highly challenging task to robustly classify terrain. Especially, the predominantly used vision-based approaches suffer from rapid appearance changes due to various factors including illumination variations, changes in weather, damping due to rain and camouflaging with leaves. Accordingly, researchers have also explored the utilization of alternative modalities such as ladars or vibrations measured using accelerometers. Each of these approaches have their own advantages and disadvantages. For example, optical sensors are quintessential when there is good illumination and distinct visual features, while accelerometer-based approaches are ideal to classify terrains with varying degrees of coarseness. However, the use of sound to classify terrains in the past has not been studied in a comparable depth, even though sound produced from vehicle-terrain interactions have distinct audio signatures even utilizable for fine-grained classification. Most importantly, the disturbances that affect other light-based or active sensors do not affect microphones, hence they can even be used as a complementary modality to increase robustness. We believe that utilization of a complementary set of sensing modalities is geared towards long-term autonomy.

In this paper, we present a novel multiclass terrain classification approach that uses only audio from the vehicle-terrain interaction to robustly classify a wide range of indoor and outdoor terrains. As in any pattern recognition task, the choice of features significantly dictates the classification performance. Vehicle-terrain interaction

A. Valada (✉) · L. Spinello · W. Burgard
Department of Computer Science, University of Freiburg, Freiburg, Germany
e-mail: valada@informatik.uni-freiburg.de

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_2

sounds are unstructured in nature as several dynamic factors contribute to the signal. Instead of using handcrafted domain specific features, our approach employs a deep convolutional neural network (DCNN) to learn them. DCNNs have recently been achieving state of the art performance on several pattern recognition tasks [13, 14, 18]. They learn unsupervised hierarchical feature representations of their input by exploiting spatial correlations. The additional advantage of this is that the features learned from this approach generalize effectively as DCNNs are relatively insensitive to certain input variations.

The convolutional neural network architecture we introduce is built upon recent advances in deep learning. Our network consisting of six convolution layers and six cascaded cross channel parametric pooling layers is depicted in Fig. 1. In order to make the learned feature representations invariant to certain signal variations and also to increase the number of training samples, we performed a number of transformations on the original signal to augment the data. We experimented with several hyperparameters for our network and show that it significantly outperforms classification methods using popular baseline audio features. To the best of our knowledge, this is the widest range of terrain classes successfully classified using

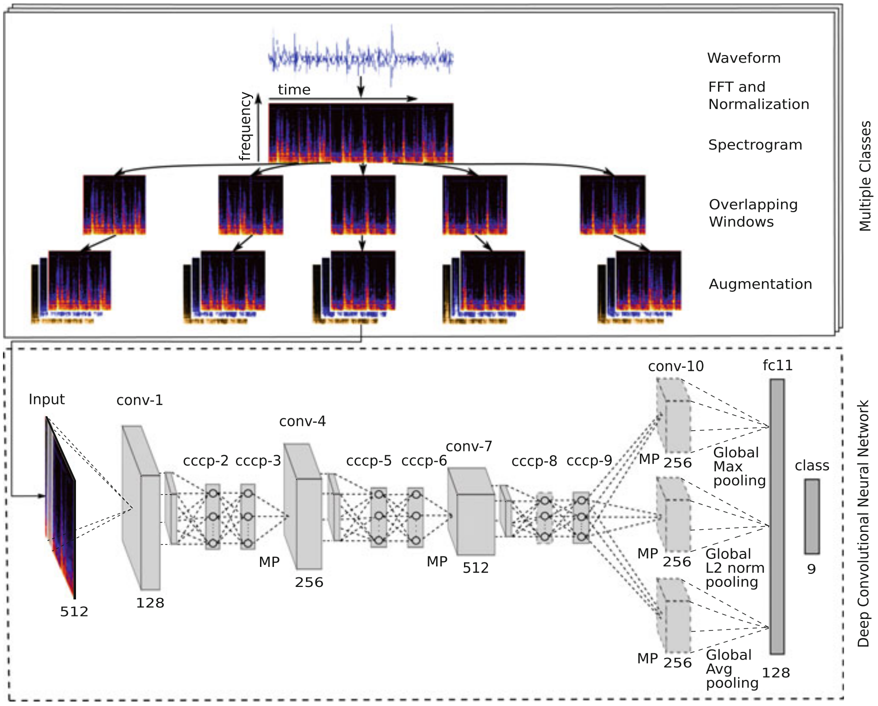


Fig. 1 Overview of our terrain classification pipeline. Raw audio signal of the terrain interaction is first transformed into its spectrogram representation and then piped into a DCNN for feature learning and classification. MP refers to max pooling

any proprioceptive terrain classification system. Additionally, our method achieves state of the art performance in classification using a proprioceptive sensor. Audio classification is susceptible to background noise to a great extent. We stress test our network with additive white Gaussian noise (WGN) at varying signal to noise ratios (SNR). We also perform noise aware fine-tuning to increase the robustness and show that our network performs exceptionally well even on audio data collected by the robot with a low quality mobile phone microphone which adds significant environmental noise.

2 Related Work

The use of sound as a modality for classifying vehicle-terrain interactions has very sparsely been explored. The following are the only related works using acoustics for terrain classification. Ojeda et al. [17], used a feedforward neural network and a suite of sensors for terrain classification, including a microphone, gyroscopes, accelerometers, motor current and voltage sensors, infrared, ultrasonics and encoders. They had five terrain classes and their classifier achieved an average classification accuracy of 60.3% using the microphone. They found that using the entire spectrum gave them the same performance as using only 0–50 Hz components of the discrete fourier transform. The authors concluded that overall the performance was poor using the microphone, other than for classifying grass.

More recently, Libby and Stentz [15] trained a multiclass sound-based terrain classifier that uses Support Vector Machines (SVMs). They evaluated the performance of various features using extraction techniques derived from the literature survey as input to the SVM. Their multidimensional feature vectors consists of spectral coefficients, moments and various other temporal as well as spectral characteristics. Their classifier achieves an average accuracy of 78% over three terrain classes and three hazardous vehicle-terrain interaction classes. They further increase the accuracy to 92% by smoothing over a window of 2 s.

A patent by Hardsell et al. [8] describes an approach to terrain classification where a classifier is trained on fused audio and video data. They extract scale invariant transformation features from the video data and use Gaussian mixture models with a time-delay neural network to represent the audio data. The classifier is then built using expectation-maximization.

The use of contact microphones for terrain classification has also been explored. Unlike air microphones that we use in our work, contact microphones pick up only structure-borne sound. Brooks and Iagnemma [2] use a contact microphone mounted on their analog rover's wheel frame to classify terrain. They extract the log-scaled Power Spectral Density (PSD) of the recorded vibrations and used them to train a pairwise classifier. Their classifier with three classes, achieves an average accuracy of 74% on a wheel-terrain testbed and 85.3% on the test bed rover. They also present a self-supervised classifier that was first trained on vibration data, which then provided the labels for training a visual classifier [3].

A number of methods have been developed for using accelerometer data to classify terrain [17, 19, 21]. Weiss et al. [21] use vibrations induced in the vehicles body during traversal to classify the terrain. They train a seven class SVM with features extracted from log-scaled PSD, discrete fourier transform and other statistical measures. Their classifier produced an average accuracy of 91.8% over all the classes. However, such approaches report a significant number of false positives for finer terrains such as asphalt and carpet. For another similar application, Eriksson et al. [5] employ a mobile sensor network system that uses hand selected features from accelerometer data to identify potholes and other road anomalies. Their system detects the anomalies over 90% of the time in real-world experiments.

There is a considerable amount of specialized audio features developed for speech recognition and music classification, but it remains unclear which of these features performs well for our application. We evaluated several traditional audio features from our literature survey and compared them as baseline approaches. Libby and Stentz [15] show that a combination of Ginna and Shape features perform the best for classification of vehicle-terrain interactions. Gina features, based on the work by Giannakopoulos et al. [6] is a 6D feature vector consisting of zero crossing rate (ZCR), short time energy (STE), spectral centroid, spectral rolloff and spectral flux. Shape features, based on the work by Wellman et al. [22], characterize the distribution of moments of the spectrum. It is a 4D feature vector consisting of spectral centroid, standard deviation, skewness and kurtosis.

Ellis [4] use a combination of mel-frequency cepstral coefficients (MFCCs) and chroma features. MFCCs are the most widely used features for audio classification and Chroma features are strongly related to the harmonic progression of audio signals. We use a combination of twelve bin MFCC's and twelve bin Chroma features for comparison. Trimbral features have been a popular set of features for various audio classification applications. Tzanetakis and Cook [20] use a 19D feature representation consisting of means and variances of spectral centroid, rolloff, flux, ZCR, low energy and means and variances of the first 5 MFCCs. For our final feature set comparison, we use a combination of 13 bin MFCC's, line spectral pair (LSP) and linear prediction cepstral coefficients (LPCCs) [1]. We call this Cepstral feature set in the later discussions.

3 Deep Convolutional Neural Network for Acoustic Based Terrain Classification

One of the main objectives of our work is to develop a new deep convolutional neural network architecture tailored to classifying unstructured vehicle-terrain interaction sounds. In this section, we detail the various stages of our classification pipeline shown in Fig. 1. Our approach can be split into two main stages. The first stage involves processing the raw audio samples into short windowed clips, augmenting the samples and spectrogram transformation. The second involves training our deep convolutional neural network with this data.

3.1 Preprocessing and Spectrogram Extraction

We first split the audio signals from each class into small “clips” of t_w seconds. We experimentally determine the shortest clip length that gives the best classification performance. Feature responses from each of these clips are then extracted and added as a new sample for classification.

Features derived from spectrogram representations of audio signals have been shown to outperform other standard features for environmental sound classification applications [12]. Therefore in our approach, we extract the Short Time Fourier Transform (STFT) based spectrogram of each clip in our dataset. We first block each audio clip into M samples with 75% overlap between each frame. Let $x[n]$ be the recorded raw audio signal with duration of N_f samples, f_s the sampling frequency, $S(i, j)$ be the spectrogram representation of the 1-D audio signal and $f(k) = kf_s/N_f$. By applying STFT on length M windowed frame of signal, we get

$$X(i, j) = \sum_{p=0}^{N_f-1} x[n]\, w[n-j] \exp\left(-p\frac{2\pi k}{N_f}n\right), \quad p = 0, \ldots, N_f - 1 \tag{1}$$

A Hamming window function $w[n]$ is used to compensate for Gibbs effect while computing STFT by smoothing the discontinuities at the beginning and end of the audio signal.

$$w[n] = 0.54 - 0.46 \cos\left(2\pi \frac{n}{M-1}\right), \quad n = 0, \ldots, M - 1 \tag{2}$$

We then compute the log of the power spectrum as

$$S_{log}(i, j) = 20 \log_{10}(|X(i, j)|) \tag{3}$$

We chose N_f as 2,048 samples, therefore the spectrogram contains 1,024 Fourier coefficients. By analyzing the spectrum, we found that most of the spectral energy is concentrated below 512 coefficients, hence we only use the lower 512 coefficients to reduce the computational complexity. The noise and intensity levels vary a fair amount in the entire dataset as we collected data in different environments. Therefore, we normalized the spectrograms by dividing by the maximum amplitude. We compute the normalized spectrogram as $S(i, j) = S_{\log}(i, j)/max_{i,j} S_{\log}(i, j)$. We then compute the mean spectrum over the entire dataset and subtract it from the normalized spectrogram to remove any temporal artifacts.

We created additional training samples by applying a set of augmentation strategies A_t on the audio signal in the frequency domain. Offsets in time and frequency was used to perform shifting to transform the spectrogram. The transformations were applied using 2D affine transform and warping, keeping the shape constant. Furthermore we created more samples using time stretching, modulating the tempo, using random equalization augmentation and by increasing as well as decreasing the volume gain. We also experimented with frequency and time normalization with a sliding window and local contrast normalization.

3.2 Network Architecture and Training

The extracted spectrograms in our training set are of the form $S = \{s^1, \ldots, s^M\}$ with $s^i \in \mathbb{R}^N$. Each of them are of size $v \times w$ and number of channels d ($d = 1$ in our case). We assume M to be the number of samples and y^i as the class label in one-hot encoding, $y^i \in \mathbb{R}^C$, where C is the number of classes. We then train the DCNN by minimizing the negative log likelihood of the training data. Our network shown in Fig. 1 has six Convolution layers, six Cascaded Cross Channel Parametric Pooling (CCCP) layers, two Fully-Connected (FC) layers and a Softmax layer. All the convolution layers are one dimensional with a kernel size of three and convolve along the time dimension. We use a fixed convolutional stride of one. CCCP layers follow the first, second and third convolution layers. CCCP layers was proposed by Lin et al. [16] to enhance discriminability for local patches within the receptive fields. CCCP layers are effectively employ 1×1 convolutions over the feature maps and the filters learnt are a better non-linear function approximator. A max-pooling layer with a kernel of 2, then follows the second and fourth CCCP layers. Max-pooling adds some invariance by only taking the high activations from adjacent hidden units that share the same weight, thereby providing invariance to small phase shifts in the signal.

DCNNs that are used for feature learning with images are designed to preserve the spatial information of objects in context, however for our application we are not interested to localize features in the frame, rather we are only interested to identify the presence or absence of features in the entire frame. Therefore, we added three different global pooling layers after CCCP-9 to compute the statistics across time. This global pooling approach is similar to that used for content based music recommendation by Oord et al. [18]. For global pooling layers, we use max pooling, L2 norm pooling and average pooling. We experimented with just one global pooling layer and combinations of two global pooling layers and the accuracy dropped over 3% while compared to using all three global pooling layers. We also investigated the effect of global stochastic pooling with the other three pooling combinations, but the network did not show any significant improvement. Finally, a fully connected layer is then used to combine outputs of all the global pooling layers.

Rectified linear units (ReLUs) have significantly helped in overcoming the vanishing gradient problem. They have been shown to considerably accelerate the training compared to *tanh* units. We use ReLUs $f(x) = \max(0, x)$, after the convolution layers and dropout regularization [10] on fully connected layers except the softmax layer. We used a dropout probability of 0.5. We also experimented with Parameterized Rectified Linear Units (PReLU) [9], which has shown to improve model fitting but it drastically affected our performance compared to ReLUs.

We used Xavier weight initialization [7] for the Convolution, CCCP and FC layers. The Xavier weight filler initializes weights by drawing from a zero mean uniform distribution from $[-a, a]$ and a variance as a function of the number of input neurons, where $a = \sqrt{3\ /\ n_{in}}$ and n_{in} is the number of input neurons. Using this strategy enables us to move away from the traditional layer by layer generative pre-training. Let $f_j(s^i; \theta)$ be the activation value for spectrogram s^i and class j, θ be the parameters of the network (weights W and biases b). The softmax function and the loss is computed as

$$P(y = j \mid s^i; \theta) = \text{softmax}(f(s^i; \theta)) = \frac{\exp(f_j(s^i; \theta))}{\sum_{k=1}^{K} \exp(f_k(s^i; \theta))} \tag{4}$$

where $P(y = j \mid s^i; \theta)$ is the probability of the j^{th} class and the loss can be computed as $\mathcal{L}(u, y) = -\sum_k y_k \log u_k$. Using stochastic gradient decent (SGD), we then solve

$$\min_\theta \sum_{i=1}^{N} \mathcal{L}(\text{softmax}(f(s^i; \theta)), y^i) \tag{5}$$

We use minibatch SGD with a momentum of 0.9 and a batch size of 128. Minibatch SGD refers to a more efficient way of computing the derivatives before updating the weights in proportion to the gradient, especially in large datasets such as ours. We improve the efficiency by computing the derivative on a random small minibatch of training samples, rather than the entire training set which would be computationally exhaustive. Furthermore, we optimize SGD by smoothing the gradient computation for minibatch t using a momentum coefficient α as $0 < \alpha < 1$. The update rule can then be written as

$$\Delta w_{ij}(t) = \alpha \Delta w_{ij}(t-1) - \epsilon \frac{\partial E}{\partial w_{ij}(t)} \tag{6}$$

We employ a weight decay of $\lambda = 5 \cdot 10^{-4}$ to regularize the network. We begin the training with an initial learning rate of λ_0 and reduced it every iteration by an inverse learning rate policy as $\lambda_n = \lambda_0 * (1 + \gamma * N)^{-c}$. Where λ_0 is the base learning rate, N is the number of iterations and c is the power. We use $c = 0.75$ and $\gamma = 0.1$. We determine the hyperparameter λ_0 by experimenting with different rates in an initial trial. The best performing rate of 10^{-2} was then ascertained. The entire training of 350 K iterations (~135 epochs) took about 4 days on a single GPU.

3.3 Noise Aware Fine-Tuning

Classification performance is often strongly affected by noise from the environment. Since the microphone is mounted on the robot and used in real-world environments, it is inevitable that the recorded signals include the robot's motor noise in addition to environmental noise. Fortunately deep networks have good generalization to real-world scenarios if they are trained with noisy samples. In order to quantify the performance in the presence of noise, we added WGN to training samples at various SNR's and measured the classification accuracy. WGN adds a very similar effect as various physical and environmental disturbances including wind and water sources.

From experiments detailed in Sect. 5.4, it can be seen that the classification performance of our network quickly drops below SNRs of 40 dB. As a solution to this problem, we augmented raw audio signals with additive WGN at SNRs ranging from 50 dB to -10 dB, in steps of 10 dB. We then performed noise adaptive fine-tuning of all the layers in our network with the training set containing both noised and original samples. The weights and biases are initialized by coping from our original model trained as described in Sect. 3.2. The new model is then trained by minimizing the negative log likelihood as shown in Eq. (5). We again use minibatch SGD with a learning rate $1/10th$ of the initial rate use for training the network, 10^{-3}. The learning rate was further reduced by a factor of 10, every 20,000 iterations.

4 Data Collection and Labeling

As we are particularly interested in analyzing the sounds produced from the vehicle-terrain interaction on both indoor and outdoor terrains, we use the Pioneer P3-DX platform which has a small footprint and feeble motor noise. Interference from nearby sound sources in the environment can drastically influence the classification. It can even augment the vehicle-terrain interaction data by adding its own attributes from each environment. In order to prevent such biases in the data being collected, we use a shotgun microphone that has a supercardioid polar pattern which helps in rejecting off-axis ambient sounds. We chose the Rode VideoMic Pro and mounted it near the right wheel of the robot as shown in Fig. 2. The integrated shock mount in the microphone prevents any unwanted vibrations from being picked up.

We collected over 15 h of audio data from a total of 9 different indoor and outdoor terrains. We particularly choose our terrain classes such that some of them have similar visual features (Fig. 3a, h, i) and hence pose a challenge to vision based approaches. The data was collected at several different locations to have enough generalizability, therefore even signals in each class have varying temporal and spectral characteristics. The robots speed was varied from 0.1 to 1.0 ms$^-1$ during the data collection runs. The data was recorded in the lossless 16-bit WAV format at 44.1 kHz to avoid any recording artifacts. Experiments were conducted by recording at various preamp levels and microphone mounting locations. There was no software level

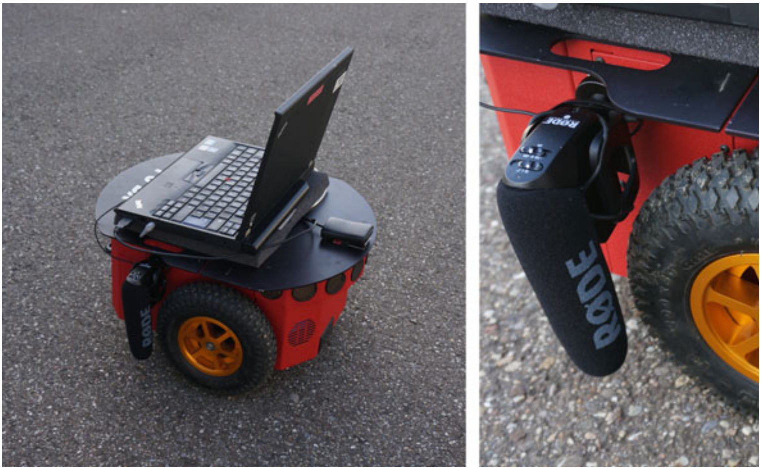


Fig. 2 The Pioneer P3-DX platform showing the shotgun microphone with the shock mount, mounted close to the wheel

boost added during the final recordings as they also tended to amplify the ambient noise significantly, instead the microphones 20 dB hardware level boost was turned on.

We manually label the data by looking at live tags with timestamps from recordings and we use a waveform analyzer tool to crop out any significant disturbances. The data from each class was then split into overlapping time windows, where each window is then used separately as a new data sample for feature extraction. As Libby et al. mention in [15], choosing an appropriate length for the time window is critical, as too short of a window might cut off a potential feature and by having too large of a window we will loose the classification resolution. We also analyzed the effect of different window sizes in our experiments. In order to train the classifier to be generalizable to different locations with the same terrain, a ten-fold cross validation approach was adopted. Furthermore, we ensured that all the sets and classes have approximately the same number of samples to prevent any bias towards a specific class.

5 Experimental Results

We performed the implementation and evaluations using the publicly available, Caffe [11] deep learning toolbox and ran all our experiments on a system with an Intel i7-4790K processor and a NVIDIA GTX 980M GPU. We used the cuDNN library for GPU acceleration. For all the baseline comparisons and noise robustness tests, we chose a clip window length of 300 ms and performed ten-fold cross-validation. The results from our experiments are described in the following sections.

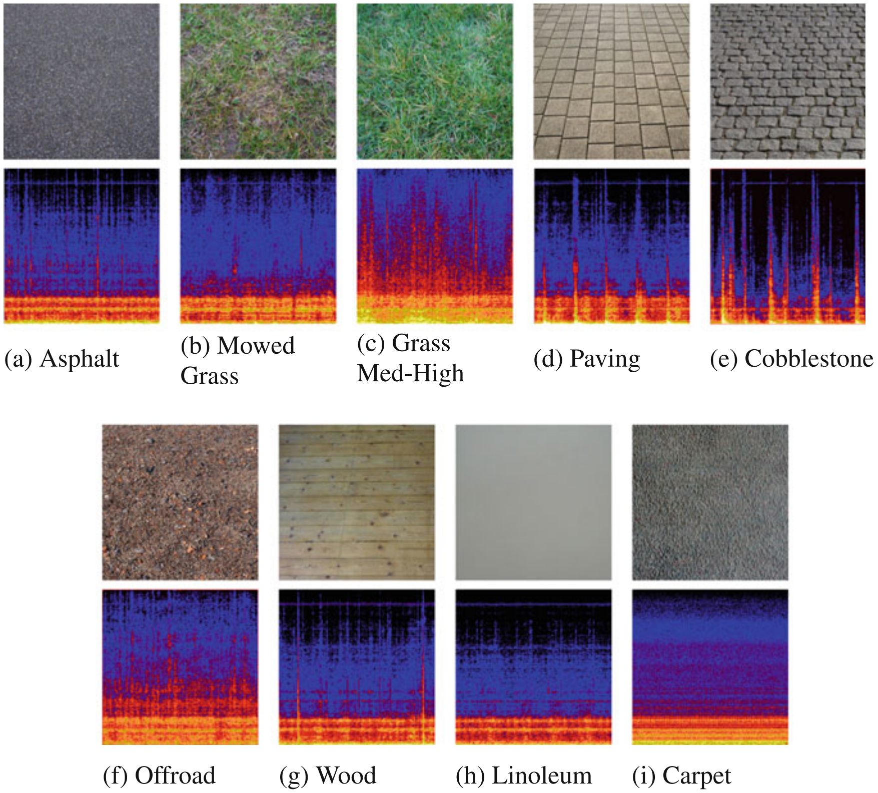


Fig. 3 Terrain classes and an example spectrogram of a 2,000 ms clip (colorized spectrograms are only shown for better visualization, spectrograms used for training are in gray scale)

5.1 Baseline Comparison

We chose two benchmark classifiers, k-Nearest Neighbors (kNNs) and SVMs. SVMs perform well in high dimensional spaces and kNNs perform well when there are very irregular decision boundaries. As a preprocessing step we first normalize the data to have zero mean. We use the one-vs-rest voting scheme with SVM to handle multiple classes and experimented with Linear and Radial Basis Function (RBF) kernels as decision functions. We used inverse distance weighting for kNNs and optimized the hyperparameters for both the classifiers by a grid-search using cross-validation. We empirically evaluated six popular feature combinations described in Sect. 2, with SVM and kNN. We used scikit-learn and LibSVM for the implementation. It was ensured that the training and validation sets do not contain the same audio split or the augmented clip. The results from this comparison are shown in Table 1.

The best performing baseline feature-classifier combination was Cepstral features using a linear SVM kernel, although the performance using Trimbral features are

Table 1 Classification accuracy of several baseline feature extraction approaches on our dataset

Features	SVM Linear	SVM RBF	k-NN
Ginna	44.87 ± 0.70	37.51 ± 0.74	57.26 ± 0.60
Spectral	84.48 ± 0.36	78.65 ± 0.45	76.02 ± 0.43
Ginna and Shape	85.50 ± 0.34	80.37 ± 0.55	78.17 ± 0.37
MFCC and Chroma	88.95 ± 0.21	**88.55 ± 0.20**	88.43 ± 0.15
Trimbral	89.07 ± 0.12	86.74 ± 0.25	84.82 ± 0.54
Cepstral	**89.93 ± 0.21**	78.93 ± 0.62	**88.63 ± 0.06**
DCNN (ours)	**97.36 ± 0.12**		

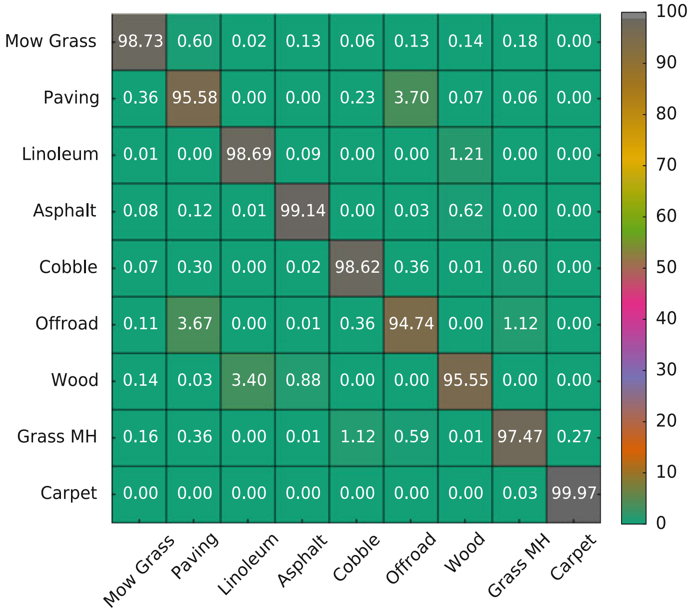
closely comparable. This feature set outperformed Ginna and Shape features by over 9%. Ginna and Shape features using an SVM RBF kernel was the best performing combination in the work by Libby and Stentz [15]. The worst performance was from Ginna features using an SVM RBF kernel. It can also be seen that the feature sets containing MFCCs show comparatively better results than the others.

Our DCNN yields an overall accuracy of 97.36 ± 0.12%, which is a substantial improvement over the hand-crafted feature sets. We get an improvement of 7% over the best performing Cepstral features and 12% over Ginna and Shape features using the same clip length of 300 ms. Furthermore, using a clip window size of 500 ms, our network achieves an accuracy of 99.41%, a 9% improvement over the best performing baseline approach. This strongly demonstrates the potential for using sound to classify vehicle-terrain interactions in a variety of environments.

5.2 *Overall DCNN Performance*

To further investigate classification performance of our network we computed the confusion matrix, which helps us understand the misclassifications between the classes. Figure 4 shows the confusion matrix for ten-fold cross validation.

The best performing classes were carpet and asphalt, while the most misclassified was offroad and paving, which were sometimes confused with each other. Both these classes have similar spectral responses when the clip window gets smaller than 500 ms. Our system still outperforms all baseline approaches by wide margin. We also compared the per-class recall as it gives an insight on the ratio of correctly classified instances. Figure 5 shows the per-class recall using ten-fold cross validation. The network achieves an overall recall of 97.61%.



	Mow Grass	Paving	Linoleum	Asphalt	Cobble	Offroad	Wood	Grass MH	Carpet
Mow Grass	98.73	0.60	0.02	0.13	0.06	0.13	0.14	0.18	0.00
Paving	0.36	95.58	0.00	0.00	0.23	3.70	0.07	0.06	0.00
Linoleum	0.01	0.00	98.69	0.09	0.00	0.00	1.21	0.00	0.00
Asphalt	0.08	0.12	0.01	99.14	0.00	0.03	0.62	0.00	0.00
Cobble	0.07	0.30	0.00	0.02	98.62	0.36	0.01	0.60	0.00
Offroad	0.11	3.67	0.00	0.01	0.36	94.74	0.00	1.12	0.00
Wood	0.14	0.03	3.40	0.88	0.00	0.00	95.55	0.00	0.00
Grass MH	0.16	0.36	0.00	0.01	1.12	0.59	0.01	97.47	0.27
Carpet	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	99.97

Fig. 4 Confusion matrix of our approach for ten-fold cross validation, using an audio clip length of 300 ms. The network seemed to get mostly confused with Offroad and Paving, as well as Linoleum and Wood

5.3 Varying Clip Length

We compared the average cross-validated accuracy of our network using varying audio clip lengths and execution times. Each clip is essentially a new sample for classification, therefore the shorter the clip, the higher is the rate at which we can infer the terrain. In addition, the shorter the clip, the faster is the execution time. For an application such as ours, fast classification and execution rates are essential for making quick trafficability decisions. Table 2 shows the overall classification accuracy using the DCNN approach with various window sizes.

From the above table it can be seen that the deep network approach significantly outperforms classification using hand-crafted feature sets. We get an improvement of 7% over the best performing Cepstral features and 12% over Ginna and Shape features using the same clip length of 300 ms. Furthermore, using a window size of 500 ms, our network achieves an accuracy of 99.41%, a 9% improvement over the best performing baseline approach.

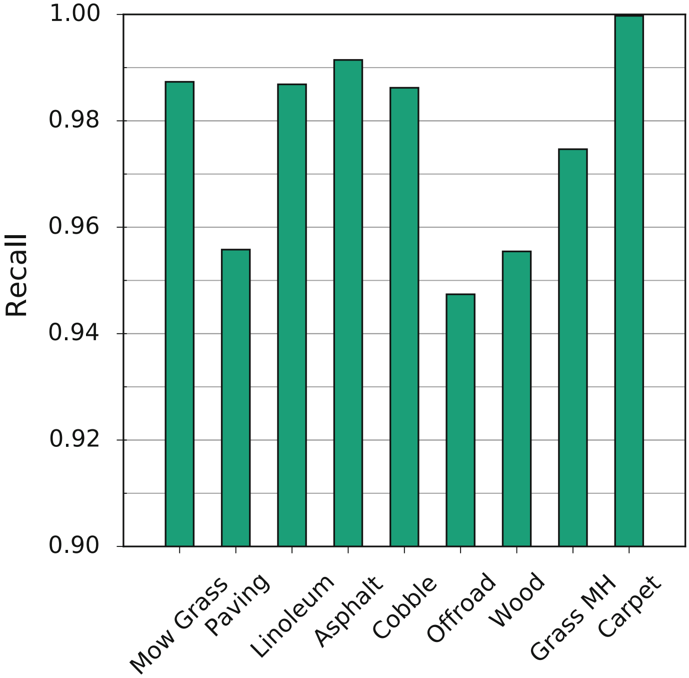


Fig. 5 Per-class recall of our network on ten-fold cross validation, using an audio clip length of 300ms. The class with the lowest recall was Paving

Table 2 Classification accuracy of our system at varying audio clip lengths and the corresponding time taken to process though the pipeline

Clip Length (ms)	2000	1500	1000	500	300
Accuracy (%)	99.86	99.82	99.76	99.41	**97.36**
Time (ms)	45.40	34.10	21.40	13.30	**9.15**

5.4 Robustness to Noise

For real-world applications such as ours, robustness to noise is a critical property. However models can only be insensitive to noise up to a certain level. We analyzed the effect of Gaussian white noise on the classification performance at several SNRs as shown in Fig. 6. It can be seen that for some classes such as carpet, grass and cobble, the performance decreases exponentially at different intensities, while for others such as linoleum and asphalt, the performance seems to be affected marginally compared to others. On the other extreme, wood and paving show remarkable robustness for SNRs upto 20 dB, thereafter the performance drops to zero. This can be attributed to the fact that spectral components are much wider for the classes that show more

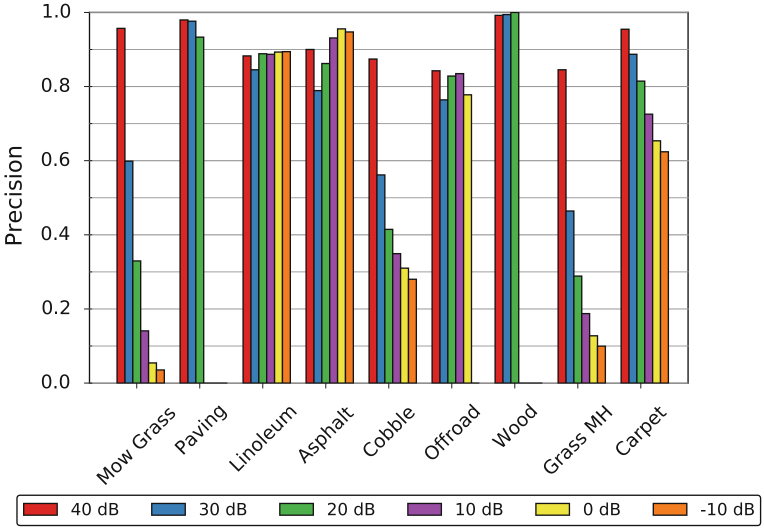


Fig. 6 Per-class precision of our network when subject to different levels of white Gaussian noise. The levels mentioned in the legend are SNRs

Table 3 Influence of white Gaussian noise onto the classification rate. SNR is in dB and accuracy is in percent. The standard deviations were less than 1%

SNR	40	30	20	10	0	–10
Before FT	91.42	76.45	70.66	45.06	41.91	32.01
After FT	99.49	99.12	98.56	97.97	97.09	95.90

FT = Fine-tuning

robustness and for the −10 dB SNR, only the classes that have certain pulses still over the noise signal are recognizable.

As a solution to this problem, we fine-tuned our trained model on samples with additive Gaussian white noise as described in Sect. 3.2. Table 3 shows the average cross-validated recognition accuracy of our network at different SNR, before and after fine-tuning. Our fine-tuned model significantly outperforms our base model on a wide range of SNRs. The best performing classes were mowed grass, linoleum, asphalt, wood and carpet, with over 99% accuracy in all the SNRs shown in Table 3. Paving, cobble and offroad classes yielded a recognition accuracy of about 95%, averaged over all the SNRs. The only class that was slightly negatively affected by the fine-tuning was wood at SNR of 20 dB, where there was a 0.2% loss in recognition performance.

We also tested our fine-tuned model on the test set with no noise samples and the average accuracy over all the classes was 99.57%, which is a 2.21% improvement over our base models performance, clearly showing that noise adaptive fine-tuning is a necessary step. This improvement can be attributed to the fact that by augmenting

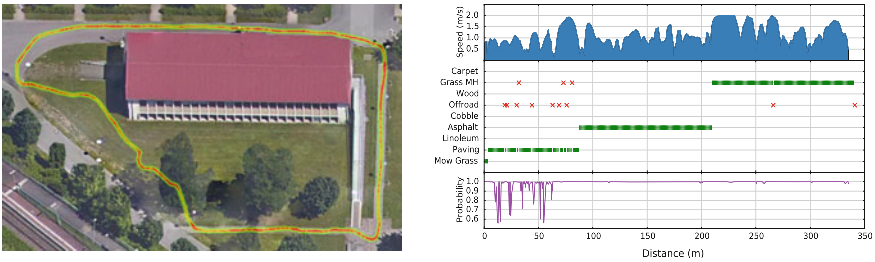
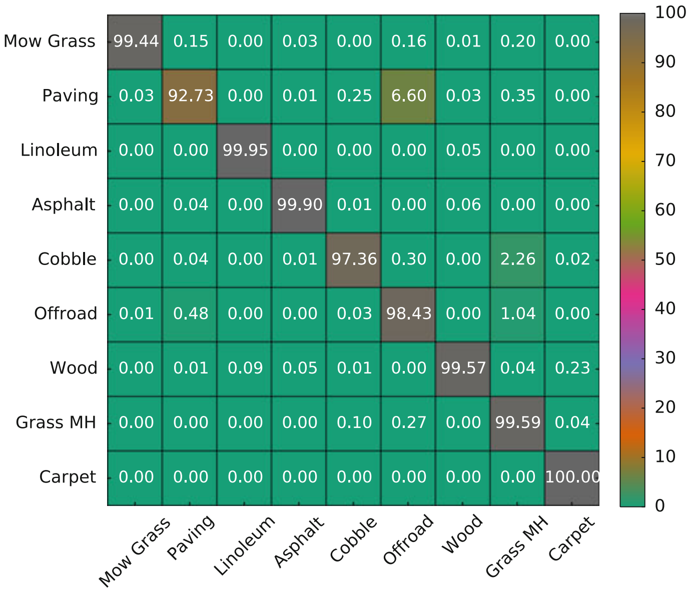


Fig. 7 The map on the left shows the trajectory taken by the robot during a classification test run using a mobile phone microphone. The variation in speed along the path is indicated in *red* and wider *red* points denote slower speed. The graph on the *right* shows the classification result, along with the corresponding probabilities for the path shown in the map. True positives are shown as *green* markers and false positives are shown are *red* markers



	Mow Grass	Paving	Linoleum	Asphalt	Cobble	Offroad	Wood	Grass MH	Carpet
Mow Grass	99.44	0.15	0.00	0.03	0.00	0.16	0.01	0.20	0.00
Paving	0.03	92.73	0.00	0.01	0.25	6.60	0.03	0.35	0.00
Linoleum	0.00	0.00	99.95	0.00	0.00	0.00	0.05	0.00	0.00
Asphalt	0.00	0.04	0.00	99.90	0.01	0.00	0.06	0.00	0.00
Cobble	0.00	0.04	0.00	0.01	97.36	0.30	0.00	2.26	0.02
Offroad	0.01	0.48	0.00	0.00	0.03	98.43	0.00	1.04	0.00
Wood	0.00	0.01	0.09	0.05	0.01	0.00	99.57	0.04	0.23
Grass MH	0.00	0.00	0.00	0.00	0.10	0.27	0.00	99.59	0.04
Carpet	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.00

Fig. 8 Confusion matrix for classification runs using data from a mobile phone microphone. Paving and Cobble show decreased performance due to false positives with Offroad and Grass

the signals with noise samples, we provide the network some prior knowledge about the distribution of the signals which boosts the recognition performance. The only significant misclassification was in the offroad class, which was 1% of the times misclassified as paving. The other classes had almost negligible misclassifications.

To further stress test our network, we collected noisy samples in a new environment using a mobile phone that also tagged each sample with a GPS location. The mobile phone has a condenser microphone, which unlike the shotgun microphone that we used before, collects sounds from every direction, thereby adding considerable amount of background noise. One of the test paths that the robot traversed is shown in the map in Fig. 7. The figure also shows then variation in speed (0–2 ms^{-1}) along the path. Thicker red lines in the map, indicate slower speed. Our network achieved an accuracy of 98.54% on the mobile phone dataset. This shows the recognition robustness, not only to real-world environments but also invariant to the type of microphone. In addition, the graph in Fig. 7 shows the false positives and true positives along the traversed path. It can be seen that most of the false positives are in the paving class and this primarily occurs when the speed is above 1 ms^{-1} and the height of the paving is highly irregular, thereby misclassifying as offroad. Interestingly, there is also significant fluctuations in the class probabilities of the false positives along the paving path when the speed is below 1ms^{-1}.

Figure 8 shows the confusion matrix for the entire mobile phone microphone dataset which contains about 2.15 h of audio data. The classes that show a dip in performance are paving, cobblestones and offroad. The paving class shows a non-negligible false positive rate as it is often misclassified as offroad. Part of this misclassification is due variation in speed and the false positives in the terrain transition boundaries.

6 Conclusion

In this paper, we introduced a novel approach that uses only sound from vehicle-terrain interactions to robustly classify a wide range of indoor and outdoor terrains. We evaluated several baseline audio features and presented a new deep convolutional neural network architecture that achieves state-of-the-art performance in proprioceptive terrain classification. Our GPU-based implementation operates on 300 ms windows and is 1,800 times faster than real-time, i.e., our system can classify a years worth of audio data in roughly 4.8 h. Additionally, our experiments in classifying audio signal corrupted with white Gaussian noise demonstrate our networks robustness to a great extent. We additionally show that our network fine-tuned with noisy samples performs exceptionally well even at low signal-to-noise ratios. Furthermore, our empirical evaluations with an inexpensive low-quality microphone shows that our approach is invariant to the type of microphone and can handle significant amount of real-world noise.

Acknowledgements This work has been partly supported by the European Commission under the grant numbers ERC-AGPE7-267686-LifeNav and FP7-610603-EUROPA2, and from the Ministry of Science, Research and Arts of Baden-Württemberg (Az: 32-7545.24-9/1/1) as well as by the German Ministry for Research and Technology under grant ZAFH-AAL.

References

1. Brijesh, V., Blumenstein, M.: Pattern Recognition Technologies and Applications: Recent Advances, IGI Global (2008)

2. Brooks, C.A., Iagnemma, K.: Vibration-based terrain classification for planetary exploration rovers. IEEE Trans. Robot. **21**(6), 1185–1191 (2005)
3. Brooks, C.A., Iagnemma, K.: Self-Supervised Classification for Planetary Rover Terrain Sensing. In: 2007 IEEE Aerospace Conference, pp.1–9 (2007)
4. Ellis, D.: Classifying music audio with timbral and chroma features. In: 8th International Conference on Music Information Retrieval (2007)
5. Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The pothole patrol: using a mobile sensor network for road surface monitoring. In: 6th Annual International conference on Mobile Systems, Applications and Services (2008)
6. Giannakopoulos, T., Dimitrios, K., Andreas, A., Sergios, T.: Violence content classification using audio features. In: Hellenic Artificial Intelligence Conference (2006)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
8. Hadsell, R., Samarasekera, S., Divakaran, A.: Audio based robot control and navigation, U.S. Patent 8532863 B2, 28 Sept 2010
9. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852 (2015)
10. Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv:cs/1207.0580v3 (2012)
11. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv:1408.5093 (2014)
12. Khunarsal, P., Lursinsap, C., Raicharoen, T.: Very short time environmental sound classification based on spectrogram pattern matching. J. Inf. Sci. **243**, 57–74 (2013)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
14. Lee, H., Largman, Y., Pham, P., Ng, A.Y.: Unsupervised feature learning for audio classification using convolutional deep belief networks. Adv. Neural Inf. Proces. Syst. **22**, 1096–1104 (2009)
15. Libby, J., Stentz, A.: Using sound to classify vehicle-terrain interactions in outdoor environments. In: 2012 IEEE International Conference on Robotics & Automation (2012)
16. Lin, M., Chen, Q., Yan, S.: Network in network. In: International Conference on Learning Representations (2014). arXiv:1409.1556
17. Ojeda, L., Borenstein, J., Witus, G., Karlen, R.: Terrain characterization and classification with a mobile robot. J. Field Robot. **29**(1) (2006)
18. Oord, A., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: Advances in Neural Information Processing Systems, vol. 26 (2013)
19. Trautmann, E., Ray, L.: Mobility characterization for autonomous mobile robots using machine learning. Auton. Robots **30**(4), 369–383 (2011)
20. Tzanetakis, G., Cook, P.: Musical genre classification of audio signals. IEEE Trans. Speech Audio Process. **10**(5), 293–302 (2002)
21. Weiss, C., Frohlich, H., Zell, A.: Vibration-based terrain classification using support vector machines. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4429–4434, Oct 9-15 2006
22. Wellman, M.C., Srour, N., Hillis, D.B.: Feature Extraction and Fusion of Acoustic and Seismic Sensors for Target Identification. In: Proceedings of SPIE 3081 (1997)

Generalizing Over Uncertain Dynamics for Online Trajectory Generation

Beomjoon Kim, Albert Kim, Hongkai Dai, Leslie Kaelbling and Tomas Lozano-Perez

1 Introduction

Given a known deterministic model of the dynamics of a system, a start and goal state, and a cost function to be minimized, trajectory optimization methods [1] can be used to generate a trajectory that connects the start and goal states, respects the constraints imposed by the dynamics, and (locally) minimizes the cost subject to those constraints. A significant limitation to the application of these methods is the computational time required to solve the difficult non-linear program for generating a near-optimal trajectory. In addition, standard techniques [1] require the transition dynamics to be known with certainty.

We are interested in solving problems online in domains that are not completely understood in advance and that require fast action selection. In such domains we will not know, offline, the exact dynamics of the system we want to control. Online, we will receive information that results in a posterior distribution over the domain dynamics. We seek to design an overall method that combines *offline* trajectory optimization and inductive learning methods to construct an *online* execution system that efficiently generates actions based on observations of the domain.

For example, we might wish a robot to move objects along a surface, potentially picking them up or pushing them, and making choices of grasps and contacts. The

B. Kim (✉) · A. Kim · H. Dai · L. Kaelbling · T. Lozano-Perez
MIT, Cambridge, MA, USA
e-mail: beomjoon@mit.edu

A. Kim
e-mail: alkim@mit.edu

H. Dai
e-mail: daih@mit.edu

L. Kaelbling
e-mail: lpk@mit.edu

T. Lozano-Perez
e-mail: tlp@mit.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_3

best way to achieve this depends on properties of the object, such as the coefficient of friction of the robot's contacts with the object and the object's center of mass (COM), which determine the system's dynamics. If we knew the friction and center of mass, it would be relatively straightforward to find an appropriate trajectory using trajectory optimization, but solving a non-linear optimization with large number of decision variables and constraints generally takes a significant amount of time.

This work builds on recent advances in supervised imitation learning [2, 3] to design a new learning-based online trajectory generation algorithm called TOIL (Trajectory Optimization as Inductive Learning). We present two general problem settings. In the *completely observable* setting, we assume that at execution time the world dynamics will be fully observed; in the manipulation domain, this would correspond to observing the friction and COM of the object. In the *partially observable* setting, we assume that properties of the domain that govern its dynamics are only partially observed; for example, observing the height and shape of an object would allow us to make a "guess" in the form of a posterior distribution over these parameters to the dynamics, conditioned on the online observations. In both cases, we desire the online action-selection to run much more quickly than would be possible if it were necessary to run a traditional trajectory optimization algorithm online.

More concretely, we aim to build a trajectory generator that, for a given initial state and goal, maps the values of the dynamics parameters to a trajectory in the observable setting, or maps from an observation to a trajectory in the partially observable setting. We do this by training a regression function that maps both the dynamics parameters and the current system state to an appropriate control action. The trajectories used for off-line training are generated by using an existing trajectory optimizer that solves non-linear programs. To minimize the number of training trajectories required, we take an active-learning approach based on MMD-IL [3], which uses an anomaly-detection strategy to determine which parts of the state space require additional training data.

The idea of reducing trajectory generation to supervised learning has been suggested before, but it is quite difficult to learn a single regressor that generalizes over a large number of trajectories. Our approach, instead, is to learn a number of local controllers (regressors) based on individual trajectories, for a given value of the dynamic parameter or observation. During training TOIL decides when additional controllers are needed based on a measure of distance between the states reached during execution and the existing set of controllers. During execution, TOIL uses the same distance criterion to select which controller among the learned controllers to use at each time step.

We evaluate TOIL in two domains: aircraft path-finding and robot manipulation. The aircraft domain is a path-planning task in which a sequence of control inputs that drive the aircraft to the goal must be found. For the observable case of this task, we show that TOIL is able to generate a trajectory whose performance is on par with the traditional trajectory optimizer while reducing the generation time by a factor of 44. In the partially observable case, we show TOIL's success rate is better than that of the trajectory optimizer, while the generation time is reduced by a factor of 52.

In the manipulation domain, a robotic hand needs to move a cylindrical object from an initial position to a target position. This involves high dimensional state and control input. We show that in this domain, for both the observable and partially observable settings, TOIL is able to reduce the trajectory generation time by a factor of thousands. Moreover, we show that TOIL can generalize over different shapes of the cylinder, and generate trajectories whose success rate is almost as same as traditional trajectory optimization.[1]

2 Related Work

The idea of combining multiple trajectories to obtain a control policy has a long history, for example [4–6]. Recently, there has been a surge of interest in learning-based methods for constructing control policies from a set of trajectories obtained from trajectory optimization [7–9]. These methods generalize a set of relatively expensive trajectory optimizations to produce a policy represented in a form that can be efficiently executed on-line.

Our goal is similar, except that we are trying to generalize over dynamic parameters. Our training examples are trajectories labeled either directly with dynamics parameters or with observations that give a distribution over dynamic parameters.

Our approach is based on the paradigm of imitation learning [10, 11]. This learning paradigm has had many successes in robotics, notably helicopter maneuvering, UAV reactive control, and robot surgery [12–14]. In imitation learning, the goal is to replicate (or improve upon) trajectories acquired from an “oracle,” usually an expert human. The work in this paper can be seen as replacing the oracle in imitation learning with a trajectory optimizer.

DAgger [2] is an influential algorithm that addresses a fundamental problem in standard supervised approaches to imitation learning. Direct application of supervised learning suffers from the fact that the state distribution induced by the learned policy differs from that of the oracle. DAgger adopts intuition from online learning, a branch of theoretical machine learning, to address this problem by iteratively executing the learned policy, collecting data from the oracle, and then learning a new policy from the aggregated data. An important drawback of Dagger is that it queries the oracle at each time step, which would require solving a non-linear optimization program every time step during training in our case.

Our work extends Maximum Mean Discrepancy Imitation Learning (MMD-IL) [3], a recently proposed imitation learning method designed to be efficient in its access to the oracle. MMD-IL learns a set of trajectories to represent a policy and uses the Maximum Mean Discrepancy criterion [15] as a metric to decide when to query the oracle for a new trajectory. In this paper, we also take this approach with a modification that makes it parameter free.

[1] The video of this can be found at: https://www.youtube.com/watch?v=r9o0pUIXV6w.

3 Completely Observable Dynamics

For clarity of exposition, we begin by defining the learning problem and the operation of TOIL in the completely observable case; in Sect. 4 we extend it to the more realistic partially observable case.

We assume a fixed initial state x_0, goal state x_g (or goal criterion) and cost functional $\mathscr{J}$. We also assume that the transition dynamics are drawn from a known class, with a particular instance determined by the parameter α drawn from set $\mathscr{A}$: $\dot{x} = f_\alpha(x, u)$. Our overall aim is to learn to map values of α to trajectories τ that go from x_0 to x_g while respecting the transition dynamics and optimizing the cost functional.

Rather than invent a direct parameterization of trajectories, we will represent a trajectory implicitly as a policy π that maps a state x to a control output u. Thus, we can think of the problem as learning a mapping $\Pi : \mathscr{A} \to (X \to U)$, which can be rewritten to a more traditional form: $\Pi : \mathscr{A} \times X \to U$. Given a set of example training trajectories of length H for a set of N different α values

$$(\alpha_j, \tau_j) = (\alpha_j, \{(x_t^{(j)}, u_t^{(j)})\}_{t=1}^{H}), \quad j = 1, \ldots, N$$

we can use traditional supervised learning methods to find parameters θ for a family of regression functions, by constructing the training set $\{((\alpha_j, x_t^{(j)}), u_t^{(j)})\}$, and using it as input to a regression method.

Because the training data represent trajectories rather than identically and independently distributed samples from a distribution, we find that it is more effective to use a specialized form of supervised learning algorithm and an active strategy for collecting training data. The remaining parts of this section describe these methods and the way in which the final learned regressor is used to generate action in the on-line setting.

3.1 Representation and Learning

The key idea in TOIL is to construct a set of *local trajectory generators* π_k and to appropriately select among them based on a two-sample test metric, MMD [15]. These trajectory generators are *local* in the sense that each of them specializes in a particular region of the space $X \times \mathscr{A}$ and can be expected to generalize well to query points that are likely to have been drawn from that same distribution of points. The final policy is a regressor that has the form $\pi(\alpha, x) = \pi_k(\alpha, x)$, where the particular π_k is chosen based on the distance between the query point (α, x) and the data that were used to train π_k. We then iterate between executing the current policy and updating it with the new data. This is to mitigate the problems associated with executing a learned policy without updating it, which has been shown to accumulate error and cause cause the trajectory to be highly erroneous [2]. Pseudo-code for the top level of TOIL is shown in Algorithm 1. It takes an initial state x_0, a goal state x_g

and a sample set $A = \{\alpha_1, \ldots, \alpha_N\}$ of dynamics parameters, and outputs a trajectory generator, Π, which is a mapping from the state at time t and the uncertain dynamics parameter α to the control to be applied at time t.

TOIL comprises three procedures: *LearnLocalTrajGenerator*, *SelectLocalTrajGenerator* and *Optimize*. The procedure *LearnLocalTrajGenerator*(τ_i, α_i) is simply a call to any supervised regression algorithm on the training set

$$\{((\alpha_i, x_t), u_t)\} \text{ for } (x_t, u_t) \in \tau_i$$

We explain the remaining procedures as we describe Algorithm 1 below.

The algorithm begins by generating a set of training trajectories using the procedure *Optimize*. This procedure is responsible for getting a training trajectory, by optimizing a nonlinear program for trajectory optimization. This nonlinear program is discussed in detail in Sect. 3.3. From each of these training trajectories, it builds a local trajectory generator and adds it to the set Π.

Once the initial training trajectories have been used to train local controllers, we begin an iterative process of ensuring coverage of the input space that will be reached by control actions generated by Π. For every value of α_i, we try to execute the trajectory that would be generated by Π starting from x_0. At each step of execution we find the local controller π that applies to x_t and α using the procedure *SelectLocalTrajGenerator*. This procedure, given in Algorithm 2, is responsible for selecting the most appropriate local trajectory generator based on the similarity metric, called MMD, between the current state and training data $\pi.D$ associated with each of the local controllers. This metric is described in detail in Sect. 3.2. If there is one, we use f_α to simulate its execution and get a new state x_{t+1}. If there is no local controller that covers the pair x_t, α, then we call the *Optimize* procedure to get a training trajectory from x_t to x_g with dynamics f_{α_i} and use it to train a new local controller, π, which we add to Π. This process is repeated until it is possible to execute trajectories for all the training α_i to reach the goal, without encountering any anomalous states. If the α_i have been chosen so that they cover the space of system dynamics that are likely to be encountered during real execution, then Π can be relied upon to generate effective trajectories.

3.2 Maximum Mean Discrepancy

The process of applying trajectory generator Π to generate an actual trajectory from an initial (x_0, α), as well as the process of actively collecting training trajectories, depends crucially on identifying when a local trajectory generator is applicable to an observed system state. This decision is based on anomaly detection using the MMD criterion [15, 16], which is a non-parametric anomaly detection method that is straightforward to implement. Other anomaly-detection methods might also be suitable in this context, as surveyed in [17].

Given two sets of data, $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ drawn i.i.d. from distributions p and q respectively, the maximum mean discrepancy (MMD) criterion

Algorithm 1 TOIL(x_0, f, x_g, A)

```
$\Pi = \{\}$
for $i = 1$ to $N$ do
  $\tau_i = Optimize(x_0, f, x_g, \alpha_i)$
  $\Pi = \Pi \cup LearnLocalTrajGenerator(\tau_i, \alpha_i)$
end for
farPtsExists = $True$
while farPtsExists do
  farPtsExists = $False$
  for $i = 1$ to $N$ do
    for $t = 0$ to $H$ do
      $\pi_t = SelectLocalTrajGenerator(\Pi, x_t, \alpha_i)$
      if isempty($\pi_t$) then
        farPtsExists = $True$
        $\tau = Optimize(x_t, f_{\alpha_i}, x_g)$
        $\pi = LearnLocalTrajGenerator(\tau, \alpha_i)$
        $\Pi = \Pi \cup \pi$
      end if
      Generate $x_{t+1}$ using $f_{\alpha_i}(x_t, \pi(x_t, \alpha_i))$
    end for
  end for
end while
return $\Pi$
```

Algorithm 2 SelectLocalTrajGenerator(Π, x_t, α)

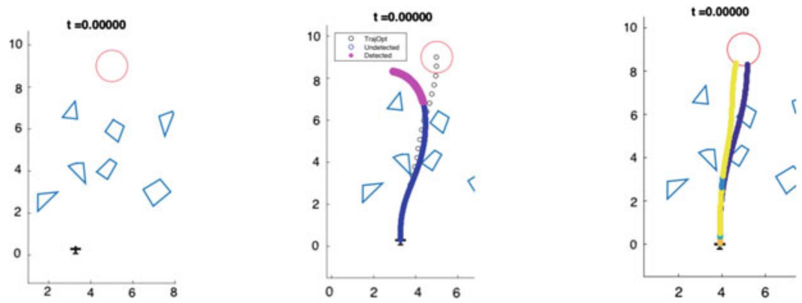
```
$candidates = \emptyset$
for $\pi_i \in \Pi$ do
  if MMD($\pi_i.D$, $(x_t, \alpha)$) < maxMMD($\pi_i.D$) then
    $candidates = candidates \cup \pi_i$
  end if
end for
if $size(candidates) == 0$ then
  return $\emptyset$
else
  return $\mathrm{argmin}_{\hat{\pi} \in candidates}$MMD($\hat{\pi}$, $(x_t, \alpha)$)
end if
```

determines whether $p = q$ or $p \neq q$, based on an embedding of the distributions in a reproducing kernel Hilbert space (RKHS).

Definition 1 (*from* [15]) Let $\mathscr{F}$ be a class of functions $f : \mathscr{X} \to \mathbb{R}$ and let p, q, X, Y be defined as above. Then MMD and its empirical estimate are defined as:

$$\mathrm{MMD}(\mathscr{F}, p, q) = sup_{f \in \mathscr{F}}(\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)])$$

$$\mathrm{MMD}(\mathscr{F}, X, Y) = sup_{f \in \mathscr{F}}\left(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i)\right)$$



(a) Initial position of the aircraft (black), obstacles (blue), and the goal region (red).

(b) Training trajectory from a trajectory optimizer (TrajOpt, black), and executed trajectory of TOIL. Magenta indicates the states that are detected as anomalies during the execution, while blue indicates those that were not detected.

(c) Training traj (navy) for an observation, and a traj found by TOIL for that observation (non-navy). Colors indicate the local trajectory generators selected by TOIL. Notice that none of the states on TOIL's trajectory is navy.

Fig. 1 Airplane control

MMD comes with an important theorem which we restate here.

Theorem 1 (from [15]) *Let $\mathscr{F}$ be a unit ball in a reproducing kernel Hilbert space $\mathscr{H}$, defined on compact metric space $\mathscr{X}$, with associated kernel $k(\cdot,\cdot)$. Then $MMD(\mathscr{F}, p, q) = 0$ if and only if $p = q$.*

Intuitively, we can expect $MMD[\mathscr{F}, X, Y]$ to be small if $p = q$, and the quantity to be large if distributions are far apart. This criterion can also be used as a metric for anomaly detection, as described in [16]. Given a training dataset D and a query point x, we can compute the following:

$$\begin{aligned}\mathrm{MMD}(\mathscr{F}, x, D) &= sup_{f\in\mathscr{F}}\Big(f(x) - \frac{1}{n}\sum_{x'\in D_i} f(x'))\Big)\\ &= \Big(k(x,x) - \frac{2}{n}\sum_{x'\in D_i} k(x,x') + \frac{1}{n^2}\sum_{x',x''\in D_i} k(x',x'')\Big)^{\frac{1}{2}} \qquad (1)\end{aligned}$$

and define $\mathrm{maxMMD}(D) = \max_{x\in D}\mathrm{MMD}(\mathscr{F}, x, D)$. As illustrated in [16], we report x as an anomaly for dataset D if $\mathrm{MMD}(x, D) > \mathrm{maxMMD}(D)$.

Figure 1b shows the result of anomaly detection using the MMD criterion in one of our domains, in which the robot needs to steer to the goal from a given initial position and a forward velocity for the robot.

Algorithm 3 TOILEx(x_0, Π, α)

```
for $t = 0$ to $H$ do
  $\pi$ = SelectLocalTrajGenerator($\Pi$, $x_t$, $\alpha$)
  execute $\pi(x_t, \alpha)$
  $x_{t+1}$ = ObserveState
end for
```

3.3 Trajectory Optimization

In trajectory optimization, the goal is to produce a locally optimal open-loop trajectory that minimizes a cost function along this trajectory, for a given initial condition x_0. The problem of finding an optimal trajectory can be formulated generically as:

$$\begin{aligned} u^*(\cdot) = \underset{u(\cdot)}{\operatorname{argmin}}\ J(x_0; \alpha) &= \underset{u(\cdot)}{\operatorname{argmin}} \int_{t=0}^{T} g(x_t, u_t)\, dt \\ s.t.\ \ \dot{x}_t = f_\alpha(x_t, u_t) &\quad \forall\, t \quad \textbf{and} \quad x_T = x_g \end{aligned} \tag{2}$$

where, x_t and u_t respectively represent the state and the control input of the system at time t, $g(x_t, u_t)$ is the cost function, $\dot{x}_t = f_\alpha(x_t, u_t)$ governs the dynamics of the system, x_0 is the initial state and x_g is the goal state.

There are multiple way of solving nonlinear programs of this form [18], any of which would be appropriate for use with TOIL.

3.4 Online Execution

Algorithm 3 illustrates the use of a learned Π in an on-line control situation. We assume that, at execution time, the parameter α is observable. At each time point, we find the local controller that is appropriate for the current state and α, execute the u that it generates, and then observe the next state. Assuming we have K local controllers, each of which is trained with H data points, the worst-case time complexity for computing a control for a given state and model is then $O(HK)$. This can be achieved by storing the Gram matrix of the dataset (i.e. the third part of Eq. 1) in a database, in which case the computation of the second part of Eq. 1 becomes the dominant term.

4 Partially Observable Dynamics

In more realistic situations, the exact value of α will not be observable online. Instead, we will be able to make observations o, which allow computation of a posterior distribution $\Pr(\alpha \mid o)$. The TOIL approach can be generalized directly to this setting, but rather than selecting the u_t to minimize $J(x_0, \alpha)$ we minimize it in expectation, hoping to obtain a trajectory that “hedges its bets” and performs reasonably well in expectation over system dynamics f_α where $\alpha \sim \Pr(\alpha \mid o)$.

In practice, we use a sampled approximation of the expected value; in particular, we assume that we have N αs drawn from $P(\alpha \mid o)$, and we formulate the constrained optimization problem as

$$u^*(\cdot) = \underset{u(\cdot)}{\operatorname{argmin}} \ \mathbb{E}_{\alpha \sim P(\alpha|o)}\big[J(x_0; \alpha)\big] \approx \underset{u(\cdot)}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \int_{t=0}^{T} g(x_t^i, u_t)\, dt$$
$$s.t. \quad \dot{x}_t^i = f_{\alpha_i}(x_t^i, u_t) \quad \forall\, t, i \quad \textbf{and} \quad x_T^i = x_g \quad \forall\, i \qquad (3)$$

Note that there are different state variables x_t^i for each possible dynamics α_i, allowing the trajectories to be different, but that there is a single sequence of control variables u_t.

The only additional change to the TOIL algorithm is that, instead of conditioning on α in the supervised learning and selection of local trajectory generators, we condition on the observation o.

5 Experiments

We evaluate our framework on two domains: aircraft path finding and robot manipulation. In all of our experiments, we use random forests [19] as our supervised learner. Specifically, we use the TreeBagger class implemented in MATLAB for the aircraft task, and RandomForestRegressor class implemented in scikit-learn [20] for the manipulation task. We used a Gaussian kernel for the MMD metric in both tasks.

To evaluate TOIL, we compute three different measures: success rate, trajectory generation time, and training time. The success rate is the percentage of the time the trajectory generated by TOIL satisfied the constraints of the environment and reached the goal. Trajectory generation time shows how much TOIL decreases the online computation burden and training time measures the off-line computation time required to learn Π for a new domain.

We compared TOIL to three different benchmarks: (1) calling a standard trajectory optimization procedure (solving Eq. 2 using snopt [21]) online in each new task instance, (2) using the initial training trajectories as input to a random forest supervised learning algorithm; and (3) DAgger, which calls trajectory optimization at every time step.

5.1 Airplane Control

This task is to cause an airplane traveling at a constant speed in the plane to avoid obstacles and reach a goal location by controlling its angular acceleration. Figure 1a shows an instance of this task. System states s_t and controls u_t are defined to be:

$$s_t = \left[x_t, y_t, \theta_t, \dot{\theta}_t\right]^T, \quad u_t = \ddot{\theta}_t$$

where (x, y) is the location of the airplane in the 2D plane, θ is the heading angle, and $\dot{\theta}$ and $\ddot{\theta}$ are the angular velocity and acceleration, respectively. The dynamics of the system is given by:

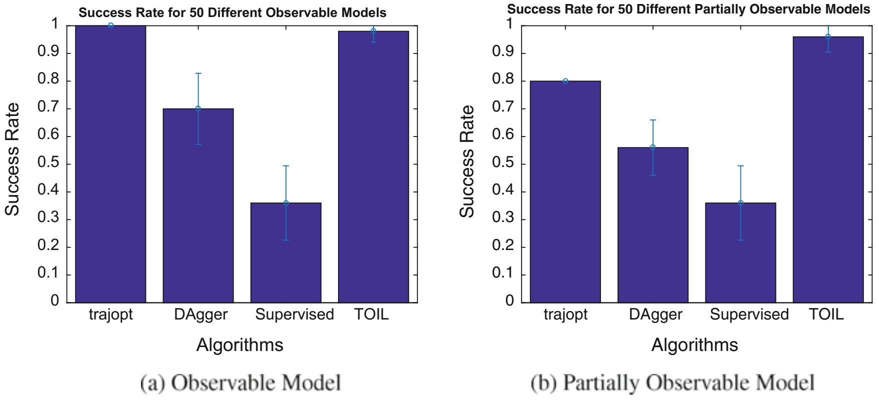
$$f(x_t, u_t) = \left[\dot{x}_t, \dot{y}_t, \dot{\theta}_t, \ddot{\theta}_t\right]^T = \left[-v \cdot sin(\theta_t), v \cdot cos(\theta_t), \ddot{\theta}_t, u_t\right]^T$$

where v denotes the constant speed of the airplane. The objective function is integrated cost $g(s_t, u_t) = u_t^2 - distToNearestObstacle(s_t)$.

We consider a trajectory to be a "success" if it does not collide with any obstacles, and arrives at the goal.

Observable Case The aspect of the dynamics that is variable, corresponding to α in the algorithms, is the speed of the aircraft, v; it is correctly observed by the system at the execution time. For training, we sampled 30 different α values from $P(\alpha) = \text{Uniform}(5, 30)$, and then generated training trajectories by solving Eq. 2. For testing, we sampled 50 different α values from $P(\alpha)$.

Figure 2a shows the success rates of the different algorithms. Trajectory optimization always returns a trajectory that is able to arrive at the goal without a collision. DAgger frequently failed to find feasible trajectories, mainly because it has not sampled training trajectories in the relevant parts of the state space. Simple supervised learning was even less successful due to its inability to sample any extra



(a) Observable Model

(b) Partially Observable Model

Fig. 2 Success rates for airplane domain

Table 1 Observable airplane domain trajectory generation and training times

Algorithm	Trajectory generation time (s)	Training time (min)	Number of traj opt calls
TOIL	2.73	64.53	32
DAgger	2.10	375.01	186
Supervised	1.57	60.50	30
Traj opt	121.31	0	0

trajectories at training time. TOIL, in comparison, is able to generate trajectories whose performance is almost on par with the trajectory optimizer.

Table 1 shows the online trajectory computation time required by each algorithm for 21 knot points. All of the learning-based methods significantly reduce the online computation time compared to the online trajectory optimization. In terms of training time, we can see that DAgger makes a very large number of calls to the trajectory optimizer, collecting training data at inappropriate regions of state space. In contrast, TOIL is able to ask only when necessary, achieving much faster training time. Overall, then, TOIL produces very good trajectories with reasonable time requirements for both training and testing.

Partially Observable Case For the partially observable case, the observation o is the mass of the aircraft, m. We assume a Gaussian distribution, $P(\alpha \mid o = m) = \mathcal{N}(\frac{1}{m}, 1)$, where m is the mass of the aircraft. During training we pick 30 different m values, and for each m we sample 5 different α values from $P(\alpha \mid o = m)$ and solve Eq. 3 to produce 30 training trajectories, each of which is intended to be robust to varying α values.

For testing, we sampled 50 different new m values, and for each, sampled 5 α values from $P(\alpha \mid o = m)$. In this phase, the robot only sees m, but not α or $P(\alpha|o)$. For online trajectory optimization, we computed a trajectory using those 5 α samples by solving Eq. 3. Then, for all learning algorithms, we report the result of evaluating the trajectory they produce on one of the five sampled α values.

Figure 2b shows the success rate of different techniques in these problems. In contrast to the observable case, the robust trajectory optimization cannot always find trajectories that succeed. This is because the non-linear optimizer needs to find a trajectory that is feasible for all 5 sampled α values, which makes the optimization problem much more difficult.

TOIL performs much better than trajectory optimization, which sometimes gets stuck in terrible local optima, such as one that collides with obstacles. TOIL is more robust because the MMD criterion is able to pick appropriate local trajectory generators depending on the initial state. In this task, most of the training trajectories have heading angles facing forward and travel through the middle of the field due to the placement of obstacles. Therefore, it is unlikely for a local trajectory generator whose training data includes traveling towards obstacles to be selected, because x_0 is located at the middle of field, facing forward. This is well illustrated in Fig. 1c.

Table 2 Partially observable airplane trajectory generation and training times

Algorithm	Trajectory generation time (s)	Training time (min)	Number of traj opt calls
TOIL	8.40	273	37
DAgger	7.84	1720.30	233
Supervised	1.75	221.50	30
Traj opt	443.12	0	0

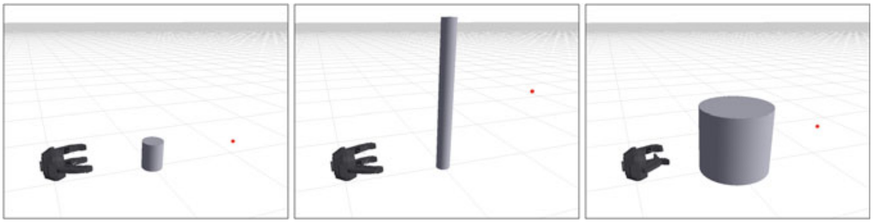


Fig. 3 Examples of the cylinders used in testing. The robotic hand is at its initial position, and the *red* dot indicates the goal location for the center of the cylinder

Here, we can see that the local trajectory generator trained with data that collides with obstacles by traveling to right is never selected during execution.

Now we consider the trajectory generation time and training time, as shown in Table 2. As the generation times show, the learning methods reduce the online computation time significantly compared to online optimization. TOIL's training time was again significantly smaller than DAgger's, and comparable with that of supervised learning.

5.2 *Manipulation Control*

In this domain, the task is to move a cylindrical object with a multi-fingered robot hand from an initial position to a goal position. A state of the robot is an element of an 84 dimensional space which consists of: position and orientation of the palm and the cylinder, as well as poses of the other 8 links relative to palm, q; associated velocities, $\dot{q}$, and accelerations $\ddot{q}$. We make use of an augmented position controller whose inputs are desired poses and an amount of time that should be taken to achieve them: $u_t = (q_{t+1}, dt_{t+1})$. The aspect of the dynamics that varies is $\alpha = (C_x, C_y, C_z)$, the center of mass of the cylinder, and in the partially observable case, the observation $o = (r, l)$ is the radius and length of the cylinder. We declare a trajectory to be successful if it does not violate the dynamics constraints and moves the cylinder to a desired goal pose. Figure 3 shows some example cylinders used in the testing phase.

The nonlinear program for trajectory optimization has the following components: decision variables $\{q_t, \dot{q}_t, \ddot{q}_t, F_t^{(1)}, F_t^{(2)}, F_t^{(3)}\}_{t=0}^{T}$ where $F^{(i)}$ is the force exerted by i^{th} finger at a contact point; objective function $g(x_t, u_t) = \alpha \dot{q}_t^2 + \beta \ddot{q}_t^2 + \gamma \sum_{i=1}^{3} {F_t^{(i)}}^2$ where $\alpha, \beta, \gamma \in \mathbb{R}$. are weights on each component; constraints between x_t, u_t and x_{t+1} that enforce the physics of the world; constraints on final and initial states; finger-tip contact constraints that require the robot to contact the object with its finger tips only; friction cone constraints between robot and cylinder, and cylinder and the surface; complementarity constraints of the form $F^{(f)} \cdot d(q_t) = 0, \quad \forall f, q_t$, where, $F^{(f)}$ denotes the force being exerted by finger f, q_t denotes the location of the hand at time t, and $d(q_t)$ denotes the distance from the object to the finger f at time t. Our implementation of the physics constraints embodies several approximations to real world physics. However, this still represents a challenging test for the learning methods.

Intuitively, given the objective function and the constraints, the optimal behavior is to simply push the object to the goal, because pushing requires the robot to exert less total force and move along a shorter trajectory than lifting the object. However, when there is uncertainty about the center of mass, pushing the object may be risky: if the height at which it pushes is too far above or below the COM, the cylinder may tip over, and so picking the object up may be preferable, in expectation. We find that when the system dynamics are observable, TOIL selects appropriate pushing trajectories, but when they are only partially observable, TOIL makes more robust choices; an example is illustrated in Fig. 4 and a few more examples are shown in the video[2]

Observable Case In the observable case, we directly observe α. For training, we sample 40 different α's from $P(\alpha = (C_x, C_y, C_z))$, which is defined as a joint uniform distribution with its range defined by the length and radius of the cylinder. For testing, we sample 50 different models from the same distribution.

Figure 5a shows the success rate of the same set of algorithms as for the airplane domain. As the figure shows, even trajectory optimization sometimes fails to satisfy the constraints within the given time limit for optimization, because the problem is quite large. While TOIL again performed just slightly worse than trajectory optimization, DAgger and supervised learning performed relatively poorly. Table 3 shows the training and trajectory computation times. The learning approaches are much more efficient at generating trajectories online than the optimizer is; Again, DAgger makes extra calls to the trajectory optimizer, while the supervised learner makes too few.

Partially Observable Case In a more realistic scenario, the robot only gets to observe length and radius of the cylinder, but not the exact center of mass. For training, we sampled 40 different observations, and sampled two different α's from the conditional distribution $P(\alpha = [C_x, C_y, C_z] | \mathbf{o} = [r, l])$, which is defined as a joint normal distribution centered at the center of the cylinder (i.e. $\mu_\alpha = (r_x, r_y, l/2)$) and variance defined as half of radius for (x, y) direction, and length in z direction. For testing, we sampled 50 different observations and tested the algorithms on one

[2] https://www.youtube.com/watch?v=r9o0pUIXV6w.

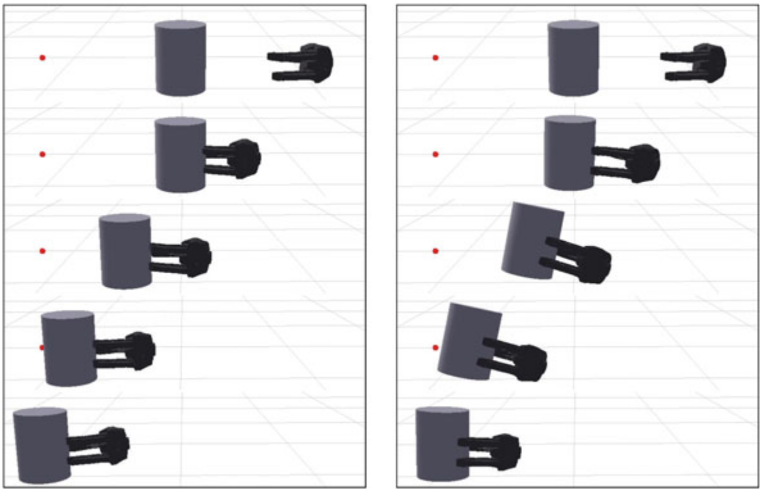
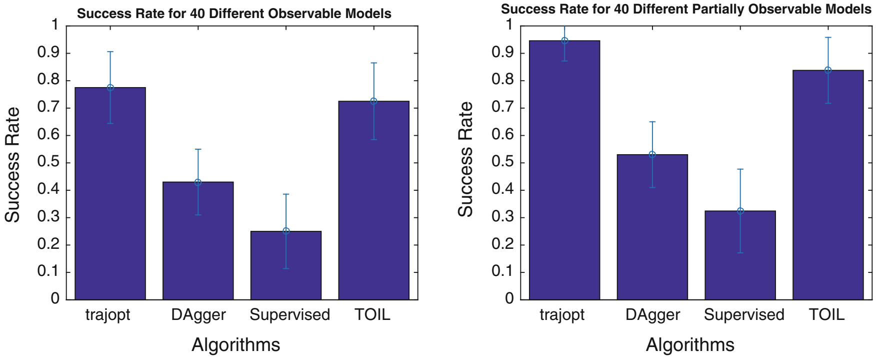


Fig. 4 Trajectories for the observable (*left*) and partially observable case (*right*). For the observable case, the robot simply pushes the object to the goal. For the partially observable case, the robot lifts the object to to the goal, as to minimize the risk of tipping the cylinder over



(a) Observable Model (b) Partially Observable Model

Fig. 5 Success rates for manipulation domain

Table 3 Trajectory computation times and training times for various algorithms

Algorithm	Trajectory generation time (s)	Training time (mins)	Number of traj opt calls
TOIL	1.10	1012	44
DAgger	1.04	1320	60
Supervised	0.45	880	40
Traj Opt	1302	0	0

Table 4 Trajectory computation times and training times of various algorithms

Algorithm	Trajectory generation time (s)	Training time (mins)	Number of traj opt calls
TOIL	1.16	1978	46
DAgger	1.04	2580	60
Supervised	0.45	1720	40
Traj Opt	2628	0	0

of the α's sampled from the same distribution. The robot gets to see only $\mathbf{o}$, but not the conditional distribution or α.

Figure 5b shows the success rate of the different algorithms. The pattern of performance is similar to the observable case. All the algorithms performed somewhat better than in the observable case, presumably because the trajectories found are less sensitive to variations in the dynamics. Table 4 shows the training and trajectory generation times. For this case, the learning algorithms are even more efficient relative to trajectory optimization, because the optimization problem is so difficult. As before, DAgger gathered much more data, while TOIL collected just enough to perform almost as well as the trajectory optimizer.

6 Conclusion

We proposed TOIL, an algorithm that learns an online trajectory generator that can generalize over varying and uncertain dynamics. When the dynamics is certain, our generator is able to generalize across various model parameters. If it is partially observable, then it is able to generalize across different observations. It is shown, in two simulated domains, to find solutions that are nearly as good as, and sometimes better than, those obtained by calling the trajectory optimizer on line. The online execution time is dramatically decreased, and the off-line training time is reasonable.

A significant concern about TOIL, as well as other supervised learning based algorithms for trajectory generation [7–9], is that the resulting controller has no guarantee of stability. In contrast, controllers synthesized from a set of local stabilizing controllers, such as LQRs, can guarantee that the controller would stabilize to the goal state [5]. Investigating the stability guarantees of supervised learning based trajectory generators would be an interesting research avenue for the future.

Acknowledgements This work was supported in part by the NSF (grant 1420927). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also gratefully acknowledge support from the ONR (grant N00014-14-1-0486), from the AFOSR (grant FA23861014135), and from the ARO (grant W911NF1410433).

References

1. Betts, J.T.: Survey of numerical methods for trajectory optimization. In: Journal of Guidance, Control, and Dynamics (1998)
2. Ross, S., Gordon, G.J., Bagnell, J.A.: A reduction of imitation learning and structured prediction to no-regret online learning. In: International Conference on Artificial Intelligence and Statistics (2011)
3. Kim, B., Pineau, J.: Maximum mean discrepancy imitation learning. In: Robotics: Science and Systems (2013)
4. Atkeson, C.: Using local trajectory optimizers to speed up global optimization in dynamic programming. In: Neural Information Processing Systems (1994)
5. Tedrake, R.: LQR-trees: feedback motion planning via sums of squares verification. In: International Journal of Robotics Research (2010)
6. Atkeson, C., Liu, C.: Trajectory-based dynamic programming, In: Modeling, Simulation, and Optimization of Bipedal Walking (2013)
7. Levine, S., Koltun, V.: Guided policy search. In: International Conference on Machine Learning (2013)
8. Levine, S., Koltun, V.: Learning complex neural network policies with trajectory optimization. In: International Conference on Machine Learning (2014)
9. Mordatch, I., Todorov, E.: Combining the benefits of function approximation and trajectory optimization. In: Robotics: Science and Systems (2014)
10. Argall, B., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. In: Robotics and Autonomous Systems (2009)
11. Bagnell, J.A.: An invitation to imitation. In Tech Report CMU-RI-TR-15-08, Robotics Institute, Carnegie Mellon University (2015)
12. Abbeel, P., Coates, A., Ng. A.Y.: Autonomous helicopter aerobatics through apprenticeship learning. In: International Journal of Robotics Research (2010)
13. Ross, S., Melik-Barkhudarov, N., Shankar, K S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning monocular reactive UAV control in cluttered natural environments. In International Conference on Robotics and Automation (2013)
14. Berg, J., Miller, S., Duckworth, D., Hu, H., Wan, A., Fu, X., Goldberg, K., Abbeel, P.: Superhuman performance of surgical tasks by robots using iterative Learning from human-guided demonstrations. In: International Conference on Robotics and Automation (2010)
15. Gretton, A., Borgwardt, K., Rasch, M., Schlkopf, B., Smola, A.: A kernel method for the two sample problem. In: Neural Information Processing Systems (2007)
16. Cristianini, N., Shawe-Taylor, J.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
17. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. In: ACM Computing Surveys (2009)
18. Betts, J.: SIAM Advances in Design and Control. Practical methods for optimal control using nonlinear programming. Society for Industrial and Applied Mathematics, Philadelphia (2001)
19. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. In: Journal of Machine Learning Research (2011)
21. Gill, P.E., Murray, W., Saunders, M.A.: Snopt: an sqp algorithm for large-scale constrained optimization. In: SIAM Journal on Optimization (2002)
22. Levine, S., Wagener, N., Abbeel, P.: Learning contact-rich manipulator skills with guided policy search. In: International Conference on Automation and Control (2015)
23. Marchese, A.D., Tedrake, R., Rus, D.: Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator. In: International Conference on Automation and Control (2015)
24. Dai, H., Valenzuela, A., Tedrake, R.: Whole-body motion planning with centroidal dynamics and full kinematics. In: International Conference on Humanoid Robots (2014)

25. Posa, M., Cantu, C., Tedrake, R.: A direct method for trajectory optimization of rigid bodies through contact. In: International Journal of Robotics Research (2014)
26. Stryk, O.V., Bulirsch, R.: Direct and indirect methods for trajectory optimization. Ann. Op. Res. **37**, 357–373 (1992)
27. Boggs, P.T., Tolle, J.W.: Sequential quadratic programming. Acta Numerica **4**, 1–51 (1995)
28. Tedrake, R.: Drake: a planning, control, and analysis toolbox for nonlinear dynamical systems (2014). http://drake.mit.edu
29. Daume, H., Langford, J., Marcu, D.: Search-based structured prediction. In: Machine Learning Journal (2009)
30. Perkins, T.J., Barto, A.G.: Lyapunov design for safe reinforcement learning. J. Mach. Learn. Res. **3**, 803–832 (2002)

Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations

Peter Englert and Marc Toussaint

1 Introduction

Most tasks in real world scenarios require contacts with the environment. For example, the task of opening a door requires contact between the robot gripper and the door handle. In this paper, we address learning from demonstration for the case of manipulation that incorporates contacts. Specifically, we want to extract from demonstrations how to represent and execute manipulations in such a way that the robot can perform such tasks in a robust and general manner.

Cost functions are a powerful representation for robot skills, since they are able to encode task knowledge in a very abstract way. This property allows them to reach high generalization to a wide range of problem configurations. However, designing cost functions by hand can be hard since the right features have to be chosen and combined with each other. Therefore, inverse optimal control, also known as inverse reinforcement learning [18], automates the design of cost functions by extracting the important task spaces and cost parameters from demonstrations. Many successful applications in different areas have demonstrated the capabilities of this idea, including the learning of quadruped locomotion [8], helicopter acrobatics [1] and simulated car driving [10].

There are two parts necessary for applying learning from demonstration with IOC: (1) The inverse optimization method for extracting the cost function from demonstrations; (2) The motion optimization method that creates motions by minimizing such cost functions. Both parts are coupled by the cost function, which is the output of the first and input of the second part, see Fig. 1. Usually IOC algorithms try to find a cost function such that the output of the motion optimization method is similar to

This work was supported by the EU-ICT Project 3rdHand 610878 and the DFG priority program Autonomous Learning 1527.

P. Englert (✉) · M. Toussaint
Machine Learning & Robotics Lab, Universität Stuttgart, Stuttgart, Germany
e-mail: peter.englert@ipvs.uni-stuttgart.de

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_4

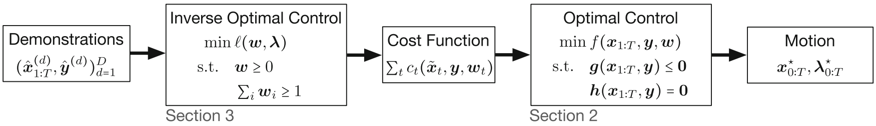

Demonstrations
$(\hat{\boldsymbol{x}}^{(d)}_{1:T}, \hat{\boldsymbol{y}}^{(d)})^{D}_{d=1}$
Inverse Optimal Control
$\min \ell(\boldsymbol{w}, \boldsymbol{\lambda})$
s.t. $\boldsymbol{w} \geq 0$
$\sum_i \boldsymbol{w}_i \geq 1$
Section 3
Cost Function
$\sum_t c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t)$
Optimal Control
$\min f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w})$
s.t. $\boldsymbol{g}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \leq \mathbf{0}$
$\boldsymbol{h}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) = \mathbf{0}$
Section 2
Motion
$\boldsymbol{x}^{\star}_{0:T}, \boldsymbol{\lambda}^{\star}_{0:T}$

Fig. 1 Concept of skill learning with inverse optimal control, where the cost function plays the central role of encoding the demonstrated behavior. In this paper, we present our formulation of learning a cost function for a constrained trajectory optimization problem

the input demonstrations of the inverse problem. Therefore, the cost function is used as a compact representation that encodes the demonstrated behavior.

Our approach finds a cost function, including the identification of relevant task spaces, such that the demonstrations fulfill the KKT conditions of an underlying constrained optimization problem with this cost function. Thereby we integrate constraints into the IOC method, which allows us to learn from object manipulation demonstrations that naturally involve contact constraints. Motion generation for such cost functions (point 2 above) is a non-linear constrained program, which we solve using an augmented Lagrangian method. However, for typical cost function parameterizations, the IOC problem of inferring the cost function parameters (point 1 above) becomes a quadratic program, which can be solved very efficiently.

The structure of the paper is as follows. We would like to defer the discussion of related work to after we have introduced our method, in Sect. 4. First, in Sect. 2, we introduce some background on constrained trajectory optimization, which represents the counterpart to the IOC approach. Afterwards, we develop our IOC algorithm in Sect. 3 by deriving a cost function based on KKT conditions. In Sect. 5 we evaluate our approach on simulated and real robot experiments.

The main contribution of this paper is the introduction of an IOC method for constrained motions with equality and inequality constraints that is based on the KKT conditions. This method allows to efficiently extract task spaces and parameters from demonstrations.

2 Constrained Trajectory Optimization

We define a trajectory $\boldsymbol{x}_{0:T}$ as a sequence of $T + 1$ robot configurations $\boldsymbol{x}_t \in \mathbb{R}^n$. The goal of trajectory optimization is to find a trajectory $\boldsymbol{x}^{\star}_{1:T}$, given an initial configuration $\boldsymbol{x}_0$, that minimizes a certain objective function

$$f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) = \sum_{t=1}^{T} c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t) \ . \tag{1}$$

This defines the objective as a sum over cost terms $c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t)$, where each cost term depends on a k-order tuple of consecutive states $\tilde{\boldsymbol{x}}_t = (\boldsymbol{x}_{t-k}, \dots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$, containing the current and k previous robot configurations [22]. This allows us to specify costs on the level of positions, velocities or accelerations (for $k = 2$) in configuration space as well as any task spaces. In addition to the robot configuration state $\tilde{\boldsymbol{x}}_t$, we use external parameters of the environment $\boldsymbol{y}$ to contain information that are important for planning the motion (parameters of the environment's configuration, e.g. object positions). These $\boldsymbol{y}$ usually vary between different problem instances, which is used to generalize the skill to different environment configurations.

We typically assume that the cost terms in Eq. (1) are a weighted sum of squared features,

$$c_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}, \boldsymbol{w}_t) = \boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})^\top \mathrm{diag}(\boldsymbol{w}_t) \boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) \ , \tag{2}$$

where $\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ are the features and $\boldsymbol{w}_t$ is the weighting vector at time t. A simple example for a feature is the robot's endeffector position at the end of the motion T relative to the position of a cup. In this example the feature $\boldsymbol{\phi}_T(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ would compute the difference between the forward kinematics mapping and cup position (given by $\boldsymbol{y}$). More complex tasks define body orientations or relative positions between robot and an object. Transition costs are a special type of features, which could be squared torques, squared accelerations or a combination of those, or velocities or accelerations in any task space.

In addition to the task costs we also consider inequality and equality constraints

$$\forall_t \qquad \boldsymbol{g}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) \leq \boldsymbol{0}, \qquad \boldsymbol{h}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y}) = \boldsymbol{0} \tag{3}$$

which are analogous to features $\boldsymbol{\phi}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ and can refer to arbitrary task spaces. An example for an inequality constraint is the distance to an obstacle, which should not be below a certain threshold. In this example $\boldsymbol{g}_t(\tilde{\boldsymbol{x}}_t, \boldsymbol{y})$ would be the smallest difference between the distance of the robot body to the obstacle and the allowed threshold. The equality constraints are in our approach mostly used to represent persistent contacts with the environment (e.g., $\boldsymbol{h}_t$ describes the distance between hand and object that should be *exactly* 0). The motivation for using equality constraints for contacts, instead of using cost terms in the objective function as in Eq. (2), is the fact that minimizing costs does not guarantee that they will become 0, which is essential for establishing a contact.

For better readability we transform Eqs. (1) and (3) into vector notation by introducing the vectors $\boldsymbol{w}$, $\boldsymbol{\Phi}$, $\boldsymbol{g}$ and $\boldsymbol{h}$ that concatenate all elements over time. This allows us to write the objective function of Eq. (1) as

$$f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) = \Phi(\boldsymbol{x}_{1:T}, \boldsymbol{y})^\top \mathrm{diag}(\boldsymbol{w}) \ \Phi(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \tag{4}$$

and the overall optimization problem as

$$\begin{aligned} \boldsymbol{x}^{\star}_{1:T} = \operatorname*{argmin}_{\boldsymbol{x}_{1:T}} \ & f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) \\ \text{s.t.} \ & \boldsymbol{g}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \leq \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) = \boldsymbol{0} \end{aligned} \tag{5}$$

We solve such problems using the augmented Lagrangian method [13]. Therefore, additionally to the solution $\boldsymbol{x}^{\star}_{1:T}$ we also get the Lagrange parameters $\boldsymbol{\lambda}^{\star}_{1:T}$, which provide information on when the constraints are active during the motion. This knowledge can be used to make the control of interactions with the environment more robust [23]. We use a Gauss–Newton optimization method to solve the unconstrained Lagrangian problem in the inner loop of augmented Lagrangian. For this problem, the gradient is

$$\nabla_{\boldsymbol{x}_{1:T}} f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) = 2\boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y})^{\top} \mathrm{diag}(\boldsymbol{w}) \boldsymbol{\Phi}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \tag{6}$$

and the Hessian is approximated as in Gauss–Newton as

$$\nabla^2_{\boldsymbol{x}_{1:T}} f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) \approx 2\boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y})^{\top} \mathrm{diag}(\boldsymbol{w}) \boldsymbol{J}(\boldsymbol{x}_{1:T}, \boldsymbol{y}), \tag{7}$$

where $\boldsymbol{J} = \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{x}}$ is the Jacobian of the features. Using a gradient based trajectory optimization method restricts the class of possible features $\boldsymbol{\Phi}$ to functions that are continuous with respect to $\boldsymbol{x}$. However, we will show in the experimental section that this restriction still allows to represent complex behavior like opening a door or sliding a box on a table.

3 Inverse KKT Motion Optimization

We now present an approach to the inverse problem for the constrained trajectory optimization formulation introduced in the previous section. To this end we learn the weight vector $\boldsymbol{w}$ in Eq. (5) from demonstrations. We assume that D demonstrations of a task are provided with the robot body (e.g., through teleoperation or kinesthetic teaching) and are given in the form $(\hat{\boldsymbol{x}}^{(d)}_{1:T}, \hat{\boldsymbol{y}}^{(d)})_{d=1}^{D}$, where $\hat{\boldsymbol{x}}^{(d)}_{1:T}$ is the demonstrated trajectory and $\hat{\boldsymbol{y}}^{(d)}$ is the environment configuration (e.g., object position).

Our IOC objective is derived from the Lagrange function of the problem in Eq. (5)

$$L(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = f(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{w}) + \boldsymbol{\lambda}^{\top} \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \\ \boldsymbol{h}(\boldsymbol{x}_{1:T}, \boldsymbol{y}) \end{bmatrix} \tag{8}$$

and the Karush–Kuhn–Tucker (KKT) conditions. The first KKT condition says that for an optimal solution $\boldsymbol{x}^{\star}_{1:T}$ the condition $\nabla_{\boldsymbol{x}_{1:T}} L(\boldsymbol{x}^{\star}_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = \boldsymbol{0}$ has to be fulfilled. With Eq. (6) this leads to

$$2\boldsymbol{J}(\boldsymbol{x}_{1:T},\boldsymbol{y})^\top \mathrm{diag}(\boldsymbol{w})\boldsymbol{\Phi}(\boldsymbol{x}_{1:T},\boldsymbol{y}) + \boldsymbol{\lambda}^\top \boldsymbol{J}_c(\boldsymbol{x}_{1:T},\boldsymbol{y}) = \boldsymbol{0} \tag{9}$$

where the matrix $\boldsymbol{J}_c$ is the Jacobian of all constraints. We assume that the demonstrations are optimal and should fulfill this conditions. Therefore, the IOC problem can be viewed as searching for a parameter $\boldsymbol{w}$ such that this condition is fulfilled for all the demonstrations.

We express this idea in terms of the loss function

$$\ell(\boldsymbol{w},\boldsymbol{\lambda}) = \sum_{d=1}^{D} \ell^{(d)}(\boldsymbol{w},\boldsymbol{\lambda}^{(d)}) \tag{10}$$

$$\ell^{(d)}(\boldsymbol{w},\boldsymbol{\lambda}^{(d)}) = \left(\nabla_{\boldsymbol{x}_{1:T}} L(\hat{\boldsymbol{x}}^{(d)}_{1:T}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{\lambda}^{(d)}, \boldsymbol{w})\right)^2 , \tag{11}$$

where we sum over D demonstrations of the scalar product of the first KKT condition. In Eq. (10), d enumerates the demonstrations and $\boldsymbol{\lambda}^{(d)}$ is the dual to the demonstration $\hat{\boldsymbol{x}}^{(d)}_{1:T}$ under the problem defined by $\boldsymbol{w}$. Note that the dual demonstrations are initially unknown and, of course, depend on the underlying cost function f. More precisely, $\boldsymbol{\lambda}^{(d)} = \boldsymbol{\lambda}^{(d)}(\hat{\boldsymbol{x}}^{(d)}_{1:T}, \hat{\boldsymbol{y}}^{(d)}, \boldsymbol{w})$ is a function of the primal demonstration, the environment configuration of that demonstration, and the underlying parameters $\boldsymbol{w}$. And $\ell^{(d)}(\boldsymbol{w},\boldsymbol{\lambda}^{(d)}(\boldsymbol{w})) = \ell^{(d)}(\boldsymbol{w})$ becomes a function of the parameters only (we think of $\hat{\boldsymbol{x}}^{(d)}_{1:T}$ and $\hat{\boldsymbol{y}}^{(d)}$ as given, fixed quantities, as in Eqs. 10 and 11).

Given that we want to minimize $\ell^{(d)}(\boldsymbol{w})$ we can substitute $\boldsymbol{\lambda}^{(d)}(\boldsymbol{w})$ for each demonstration by choosing the dual solution that analytically minimizes $\ell^{(d)}(\boldsymbol{w})$ *subject to* the KKT's complementarity condition

$$\frac{\partial}{\partial \boldsymbol{\lambda}^{(d)}} \ell^{(d)}(\boldsymbol{w},\boldsymbol{\lambda}^{(d)}) = \boldsymbol{0} \tag{12}$$

$$\Rightarrow \quad \boldsymbol{\lambda}^{(d)}(\boldsymbol{w}) = -(\tilde{\boldsymbol{J}}_c \tilde{\boldsymbol{J}}_c{}^\top)^{-1} \tilde{\boldsymbol{J}}_c \boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{\Phi})\boldsymbol{w} \; . \tag{13}$$

Note that here the matrix $\tilde{\boldsymbol{J}}_c$ is a subset of the full Jacobian of the constraints $\boldsymbol{J}_c$ that contains only the active constraints during the demonstration, which we can evaluate as $\boldsymbol{g}$ and $\boldsymbol{h}$ are independent of $\boldsymbol{w}$. This ensures that (13) is the minimizer subject to the complementarity condition. The number of active constraint at each time point has a limit. This limit would be exceeded if more degrees of freedom of the system are constrained than there are available.

By inserting Eq. (13) into Eq. (11) we get

$$\ell^{(d)}(\boldsymbol{w}) = 4\boldsymbol{w}^\top \underbrace{\mathrm{diag}(\boldsymbol{\Phi})\boldsymbol{J}\left(\boldsymbol{I} - \tilde{\boldsymbol{J}}_c{}^\top (\tilde{\boldsymbol{J}}_c \tilde{\boldsymbol{J}}_c{}^\top)^{-1} \tilde{\boldsymbol{J}}_c\right)\boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{\Phi})}_{\boldsymbol{\Lambda}^{(d)}}\boldsymbol{w} \tag{14}$$

which is the IOC cost per demonstration. Adding up the loss per demonstration and plugging this into Eq. (10) we get a total inverse KKT loss of

$$\ell(\boldsymbol{w}) = \boldsymbol{w}^\top \Lambda \boldsymbol{w} \qquad \text{with} \quad \Lambda = 4 \sum_{d=1}^{D} \Lambda^{(d)}. \tag{15}$$

The resulting optimization problem is

$$\min_{\boldsymbol{w}} \ \boldsymbol{w}^\top \Lambda \boldsymbol{w} \quad \text{s.t.} \quad \boldsymbol{w} \geq \boldsymbol{0} \tag{16}$$

Note that we constrain the parameters $\boldsymbol{w}$ to be positive. This reflects that we want squared cost features to only positively contribute to the overall cost in Eq. (4).

However, the above formulation may lead to the singular solution $\boldsymbol{w} = \boldsymbol{0}$ where zero costs are assigned to all demonstrations, trivially fulfilling the KKT conditions. This calls for a regularization of the problem. In principle there are two ways to regularize the problem to enforce a non-singular solution: First, we can impose positive-definiteness of Eq. (4) at the demonstrations (cf. [10]). Second, as the absolute scaling of Eq. (4) is arbitrary we may additionally add the constraint

$$\begin{aligned} &\min_{\boldsymbol{w}} \ \boldsymbol{w}^\top \Lambda \boldsymbol{w} \\ &\text{s.t.} \quad \boldsymbol{w} \geq \boldsymbol{0}\,, \quad \sum_i \boldsymbol{w}_i \geq 1 \end{aligned} \tag{17}$$

to our problem formulation (16). We choose the latter option in our experiments.

Equation (17) is a (convex) quadratic program (QP), for which there exist efficient solvers. The gradient $\boldsymbol{w}^\top \Lambda$ and Hessian Λ are very structured and sparse, which we exploit in our implementations.

In practice we usually use parametrizations on $\boldsymbol{w}$. This is useful since in the extreme case, when for each time step a different parameter is used, this leads to a very high dimensional parameter space (e.g., 10 tasks and 300 time steps lead to 3000 parameter). This space can be reduced by using the same weight parameter over all time steps or to activate a task only at some time points. The simplest variant is to use a linear parametrization $\boldsymbol{w}(\boldsymbol{\theta}) = A\boldsymbol{\theta}$, where $\boldsymbol{\theta}$ are the parameters that the IOC method learns. This parametrization allows a flexible assignment of one parameter to multiple task costs. Further linear parametrizations are radial basis function or B-spline basis functions over time t to more compactly describe smoothly varying cost parameters. For such linear parametrization the problem in Eq. (17) remains a QP that can be solved very efficiently.

Another option we will consider in the evaluations is to use a nonlinear mapping $\boldsymbol{w}(\boldsymbol{\theta}) = \mathscr{A}(\boldsymbol{\theta})$ to more compactly represent all parameters. For instance, the parameters $\boldsymbol{w}$ can be of a Gaussian shape (as a function of t), where the mean and variance of the Gaussian is described by $\boldsymbol{\theta}$. Such a parametrization would allow us to learn directly the time point when costs are active. In such a case, the problem is not convex anymore. We address such problems using a general non-linear programming method (again, augmented Lagrangian) and multiple restarts are required with different initializations of the parameter.

Our approach also works in the unconstrained case. In this case the constraint term vanishes in Eq. (9) and the remaining part is the optimality condition of unconstrained optimization, which says that the gradient of the cost function should be equal to zero.

4 Related Work

In the recent years, there has been extensive research on imitation learning and inverse optimal control. In the following section, we will focus on the approaches and methods that are most related to our work of learning cost functions for manipulation tasks. For a broader overview on IOC approaches, we refer the reader to the survey paper of Zhifei and Joo [24] and for an overview on general imitation learning we recommend Argall et al. [3].

4.1 Max-Entropy and Lagrangian-Based IOC Approaches

The work of Levine and Koltun [10] is perhaps the closest to our approach. They use a probabilistic formulation of inverse optimal control that approximates the maximum entropy model of Ziebart et al. [25]. In our framework of trajectory optimization (cf. Sect. 2) this translates to

$$\min_{\boldsymbol{w}} \nabla_{\boldsymbol{x}} f^\top (\nabla^2_{\boldsymbol{x}} f)^{-1} \nabla_{\boldsymbol{x}} f - \log |\nabla^2_{\boldsymbol{x}} f|. \tag{18}$$

The first term of this equation is similar to our loss in Eq. (10), where the objective is to get small gradients. Additionally, they use the inverse Hessian as a weighting of the gradient. The second term ensures the positive definiteness of the Hessian and also acts as a regularizer on the weights. The learning procedure is performed by maximizing the log-likelihood of the approximated reward function. Instead of enforcing a fully probabilistic formulation, we focus on finite-horizon *constrained* motion optimization formulation with the benefit that it can handle constraints and leads to a fast QP formulation. Further, our formulation also targets at efficiently extracting the relevant task spaces.

Puydupin-Jamin et al. [16] introduced an approach to IOC that also handles linear constraints. It learns the weight parameter $\boldsymbol{w}$ and Lagrange parameter $\boldsymbol{\lambda}$ by solving a least-squares optimization problem

$$\min_{\boldsymbol{w},\boldsymbol{\lambda}} \left(\begin{bmatrix} 2\boldsymbol{J}^\top \mathrm{diag}(\boldsymbol{\Phi}) \; \boldsymbol{J}_c^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} + \boldsymbol{J}^{/w} \right)^2 \tag{19}$$

where ${}^{/w}$ denotes the part in the cost function that is not weighted with $\boldsymbol{w}$. The method only addresses equality constraints (no complementarity condition for $\boldsymbol{\lambda}$). Our main concern with this formulation is that there are no constraints that ensure that the weight parameter $\boldsymbol{w}$ do not become 0 or negative. If $\boldsymbol{J}^{/w}$ is zero, as in our case, the solution is identially zero $(\boldsymbol{w}, \boldsymbol{\lambda})$. Starting with the KKT condition, they derive a linear residual function that they optimize analytically as the unconstrained least squares. In the experimental section they consider human locomotion with a unicycle model, where they learn one weight parameter of torques and multiple constraints that define the dynamics of the unicycle model and the initial and target position. The idea of using KKT conditions is similar to our approach. However, our formulation allows for inequality constraints and leads to a QP with boundary constraints that ensures that the resulting parameters are feasible. Instead of optimizing for $\boldsymbol{\lambda}$, we eliminate $\boldsymbol{\lambda}$ from the inverse KKT optimization using Eq. (13).

The work of Albrecht et al. [2] learns cost functions for human reaching motions from demonstrations that are a linear combination of different transition types (e.g., jerk, torque). They transformed a bilevel optimization problem, similar to [11], into a constrained optimization problem of the form

$$\min_{\boldsymbol{x}_{1:T}, \boldsymbol{w}, \boldsymbol{\lambda}} \left(\phi^{\text{pos}}(\boldsymbol{x}_T) - \phi^{\text{pos}}(\hat{\boldsymbol{x}}_T^{(d)})\right)^2 \tag{20}$$

$$\text{s.t.} \qquad \nabla_{\boldsymbol{x}_{1:T}} L(\boldsymbol{x}_{1:T}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{w}) = \boldsymbol{0} \tag{21}$$

$$\boldsymbol{h}(\boldsymbol{x}_{1:T}) = 0 \qquad \sum_i w_i = 1 \qquad \boldsymbol{w} \geq 0 \tag{22}$$

The objective is the squared distance between optimal and demonstrated final hand position. They optimize this objective for the trajectory $\boldsymbol{x}_{1:T}$, the parameter $\boldsymbol{w}$ and the Lagrange parameter $\boldsymbol{\lambda}$ with the constraints that the KKT conditions of the trajectory $\boldsymbol{x}_{1:T}$ are fulfilled. To apply this approach demonstrations are first preprocessed by extracting a characteristic movement with dynamic time warping and a clustering step. Their results show that a combination of different transition costs represent human arm movements best and that they are able to generalize to new hand positions. The advantage of their approach is that they do not only get the parameter weights $\boldsymbol{w}$, but also an optimal trajectory $\boldsymbol{x}^{\star}_{1:T}$ out of the inverse problem in Eqs. (20)–(22). The use of the KKT conditions differs from our approach in two ways. First, they use the KKT conditions in the constrained part of the formulation in Eq. (21), whereas we use them directly as scalar product in the cost function. Second, they use them on the optimization variables $\boldsymbol{x}_{1:T}$, whereas we use them on the demonstrations $\hat{\boldsymbol{x}}^{(d)}$ (see Eq. (10)). Instead of minimizing a function directly of the final endeffector position and only learning weights of transition costs, we present a more general solution to imitation learning that can learn transition and task costs in arbitrary feature spaces. Our approach also handles multiple demonstrations directly without preprocessing them to a characteristic movement.

4.2 Black-Box Inverse Optimal Control

Black-box optimization approaches are another category of methods for IOC. There, usually an optimization procedure with two layers is used, where in the outer loop black box optimization methods are used to find suitable parameter of the inner motion problem. For this usually no gradients of the outer loop cost function are required.

Mombaur et al. [11] use such a two-layered approach, where they use in the outer loop a derivative free trust region optimization technique and in the inner loop a direct multiple shooting technique. The fitness function of their outer loop is the squared distance between inner loop solution and demonstrations. They apply it on a human locomotion task, where they record demonstration of human locomotion and learn a cost function that they transfer to a humanoid robot. Rückert et al. [17] uses a similar idea to learn movements. They use covariance matrix adaptation [5] in the outer loop to learn policy parameters of a planned movement primitive represented as a cost function. Such methods usually have high computational costs for higher-dimensional spaces since the black box optimizer needs many evaluations. One also needs to find a cost function for the outer loop that leads to reasonable behavior.

Kalakrishnan et al. [7] introduce an inverse formulation of the path integral reinforcement learning method PI^2 [21] to learn objective functions for manipulation tasks. The cost function consists of a control cost and a general state dependent cost term at each time step. They maximize the trajectory likelihood of demonstrations $p(\hat{\boldsymbol{x}}_{1:T}|\boldsymbol{w})$ for all demonstrations by creating sampled trajectories around the demonstrations. Further, they L1 regularize $\boldsymbol{w}$ to only select a subset of the weights. The method is evaluated on grasping tasks.

4.3 Task Space Extraction

Jetchev and Toussaint [6] discover task relevant features by training a specific kind of value function, assuming that demonstrations can be modelled as down-hill walks of this function. Similar to our approach, the function is modelled as linear in several potential task spaces, allowing to extract the one most consistent with demonstrations. In Muhlig et al. [12] they automatically select relevant task spaces from demonstrations. Therefore, the demonstrations are mapped on a set of predefined task spaces, which is then searched for the task spaces that best represent the movement. In contrast to these methods, our approach more rigorously extracts task dimensions in the inverse KKT motion optimization framework, including motions that involve contacts.

4.4 Model-Free Imitation Learning

Another approach is the widely used framework of direct imitation learning with movement primitives [14, 15, 19]. They belong to a more direct approach of imitation learning that does not try to estimate the cost function of the demonstration. Instead they represent the demonstrations in a parametrized form that is used to generalize to new situations (e.g., changing duration of motion, adapting the target). Many extensions with different parametrization exist that try to generalize to more complex scenarios [4, 20]. They are very efficient to learn from demonstrations and have been used for manipulation tasks (e.g., pushing a box [9]).

The major difference of such kind of approaches to our method is that they do not need an internal model of the environment, which is sometimes difficult to obtain. However, if such a model is available it can be used to learn a cost function that provide better generalization abilities than movement primitives. This is the case since cost functions are a more abstract representation of task knowledge. Examples of such generalization abilities are demonstrated in Sect. 5 with a box sliding task where we generalized to different box positions and with the door opening task where we generalized to different door angles.

5 Experiments

In the following experimental evaluations, we demonstrate the learning properties and the practical applicability of our approach and compare it to an alternative method in terms of accuracy and learning speed.

For applying an IOC method a set of potential features $\boldsymbol{\Phi}$ has to be provided as input. For the following experiments we implemented a simple feature generator to produce a set of potential cost function features in a task independent manner. The used feature types are:

- **Transition features**: Represent the smoothness of the motion (e.g., sum of squared acceleration or torques)
- **Position features**: Represent a body position relative to another body.
- **Orientation features**: Represent orientation of a body relative to another body.

A body is either a part of the robot or belongs to the environment. In the following experiments the time points are either learned with RBF parametrization or they are heuristically extracted from points of interest of the demonstrations (e.g., zero velocity, contact release). We demonstrate in the following experiments that by combining such simple feature types at different time steps into a cost function allows to represent complex behavior.

First, we present on a simple task the ability to reestimate weight functions from optimal demonstrations with different weight parametrizations. Afterwards, we present more complex tasks like sliding a box and opening a door with a real PR2.

5.1 Different Weight Parametrizations in a Benchmark Scenario

The goal of our work is to learn cost functions for finite horizon optimal control problems, including when and how long the costs should be active. In this experiment we test our approach on a simple benchmark scenario. Therefore, we create synthetic demonstrations by optimizing the forward problem with a known ground truth parameter set $\boldsymbol{w}^{\mathrm{GT}}$ and test if it is possible to reestimate these parameters from the demonstrations. We create three demonstrations with 50 time steps, where we define that in the time steps 25–30 of these demonstrations the robot endeffector is close to a target position. For this experiments we use a simple robot arm with 7 degree of freedom and the target is a sphere object. We compare the three parametrizations

- **Direct parametrization**: A different parameter is used at each time step (i.e., $\boldsymbol{w} = \boldsymbol{\theta}$) which results in $\boldsymbol{\theta} \in \mathbb{R}^{50}$.
- **Radial basis function**: The basis functions are equally distributed over the time horizon. We use 30 Gaussian basis functions with standard deviation 0.8. This results in $\boldsymbol{\theta} \in \mathbb{R}^{30}$.
- **Nonlinear Gaussian**: A single unnormalized Gaussian weight profile where we have $\boldsymbol{\theta} \in \mathbb{R}^3$ with the weight as linear parameter and the nonlinear parameters are directly the mean and standard deviation. In this case the mean directly corresponds to the time where the activation is highest.

The demonstrations are used as input to our inverse KKT method (see Sect. 3) and the weights are initialized randomly. A comparison of the learned parameters and the ground truth parameter is shown in Fig. 2. The green line represents the ground truth knowledge used for creating the demonstrations. The black dots show the learned parameters of the direct parametrization. The red line shows the learned Gaussian activation and the blue line shows the RBF network. As it can be seen all parametrization detect the right activation region between the time steps 25–30 and approximate the ground truth profile. The Gaussian and RBF parametrization also give some weight to the region outside the actual cost region, which is reasonable since in the demonstrations the robot is still close to the target position. After learning with these parametrizations, we conclude that the linear RBF network are most suited to learn time profiles of cost functions. The main reason for this is the linearity of the para-

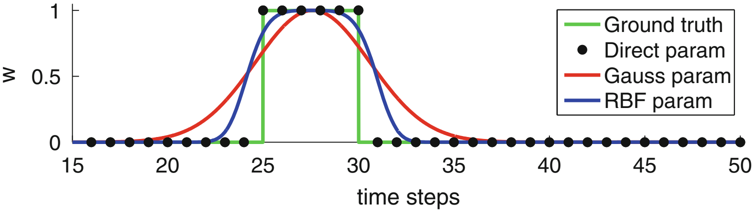


Fig. 2 Learned time profiles of different weight parameterizations. For more details see Sect. 5.1

metrization that makes the inverse KKT problem convex and the versatility of the RBF network to take on more complex forms. Directly learning the time with the nonlinear Gaussian-shaped parametrization was more difficult and required multiple restarts with different initialization. This demonstrates that the framework of constrained trajectory optimization and its counterpart inverse KKT works quite well for reestimating cost functions of optimal demonstrations.

5.2 Sliding a Box on a Table

In this experiment we use our approach to learn a cost function for sliding a box on a table. This task is depicted in Fig. 3. The goal is to move the blue box on the table to the green marked target position and orientation. The robot consist of a fixed base and a hand with 2 fingers. In total the robot has 10 degrees of freedom. Additionally to these degree of freedom we model the box as part of the configuration state, which adds 3 more degrees of freedom (2 translational + 1 rotational). The final box position and orientation is provided as input to our approach and part of the external parameters $\boldsymbol{y}$. We used three synthetic demonstrations of the task and created a set of features with the approach described above that led to $\theta \in \mathbb{R}^{537}$ parameters. The relevant features extracted from our algorithm are

- **transition**: Squared acceleration at each time step in joint space
- **posBox**: Relative position between the box and the target.
- **vecBox**: Relative orientation between the box and the target.
- **posFinger1/2**: Relative position between the robots fingertips and the box.
- **posHand**: Relative position between robot hand and box.
- **vecHand**: Relative orientation between robot hand and box.

The contacts between the fingers and the box during the sliding are modeled with equality constraints. They ensure that during the sliding the contact is maintained. For achieving realistic motions, we use an inequality constraint that restrict the movement direction during contact into the direction in which the contact is applied. This ensures that no unrealistic motions like sliding backwards or sidewards are created. For clarity we would like to note that we are not doing a physical simulation of the sliding

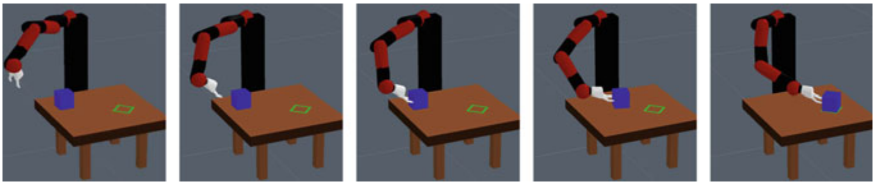


Fig. 3 These images show the box sliding motion of Sect. 5.2 where the goal of the task is to slide the *blue* box on the table to the *green* target region

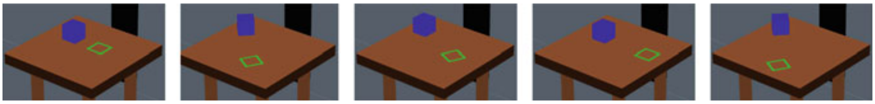


Fig. 4 Each image shows a different instance of the box sliding task. We were able to generalize to different initial box states (*blue* box) and to different final box targets (*green* area)

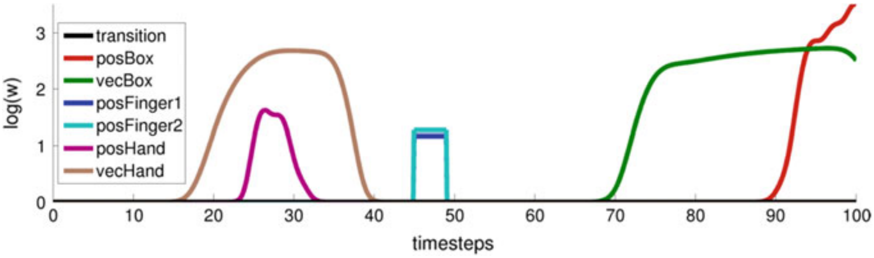


Fig. 5 The resulting parameters $\boldsymbol{w}$ of the extracted relevant features plotted over time. task is depicted in this slideshow

```
Black box IOC:
  repeat
     Resample parameters $\{w^{(n)}\}_{n=1}^{N}$ with CMA
     for all $w^{(n)}$ do
        Optimize cost function with parameter $w^{(n)}$
        Compute fitness $f^{(n)} = \sum_d (x^{(n)} - \hat{x}^{(d)})^2$
     end for
     Update CMA distribution with fitness values
  until
```

Method	$(x^{(n)} - \hat{x})^2$	comp. time
inverse KKT	0.00021	49.29 sec
black box IOC	0.00542	7116.74 sec

Fig. 6 On the *left* side is the black box IOC algorithm we used for comparison in Sect. 5.2. On the *right* side are the results of the evaluation that show that our method is superior in terms of squared error between the trajectories and computation time

behavior in these experiments. Our goal was more to learn a policy that executes a geometric realistic trajectory from an initial to a final box position. Figure 3 shows one of the resulting motion after learning. We were able to generalize to a wide range of different start and goal position of the box (see Figs. 4 and 5). Videos of the resulting motions can be found in the supplementary material.

We compare our method to a black-box optimization approach similar to [11, 17]. We implemented this approach with the black-box method Covariance Matrix Adaptation (CMA) [5] in the outer loop and our constrained trajectory optimization method (see Sect. 2) in the inner loop. The resulting algorithm is described in Fig. 6. As fitness function for CMA we used the squared distance between the current solution $\boldsymbol{x}^{(n)}$ and the demonstrations $\hat{\boldsymbol{x}}^{(d)}$. We compare this method with our inverse KKT approach by computing the error between the solution and demonstrations and the computational time, which are shown in the table in Fig. 6. The black-box method took around 4900 iterations of the outer loop of the above algorithm until it converged

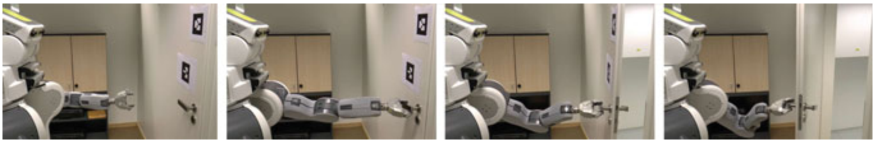


Fig. 7 The resulting motion after learning the door opening task is depicted in this slideshow. See Sect. 5.3 for more details

to a solution. This comparison shows that using structure and optimality conditions of the solution can enormously improve the learning speed. Further difficulties with black box methods is that they cannot naturally deal with constraints (in our case $\boldsymbol{w} > 0$) and that the initialization is non-trivial.

5.3 Opening a Door with a PR2

In this experiment we apply the introduced inverse KKT approach from Sect. 3 on a skill with the goal to open a door with a real PR2 robot. The problem setup is visualized in Fig. 7. We use a model of the door for our planning approach and track the door angle with AR marker. We use the left arm of the robot that consists of 7 rotational joints and also include the door angle as configuration state into $\boldsymbol{x}$. This allows us to define cost functions directly on the door angle. The gripper is fixed during the whole motion. For our IOC algorithm we recorded 2 demonstrations of opening the door from different initial positions with kinesthetic teaching. The motions also include the unlocking of the door by turning the handle first. During the demonstrations we also recorded the door position with the attached markers. We created a feature set similar to the box sliding motion from the previous experiment. Our inverse KKT algorithm extracted the features:

- Relative position & orientation between gripper and handle before and after unlocking the handle.
- Endeffector orientation during the whole opening motion.
- Position of the final door state.

We use equality constraints, similar to the box sliding experiment to keep the contact between endeffector and door. Furthermore, we use inequality constraints to avoid contacts with the rest of the robot body. A resulting motion of optimizing the constrained trajectory optimization problem with the learned parameter $\boldsymbol{w}^{\star}$ is visualized in Fig. 7. We are able to robustly generate motions with these parameters that generalize to different initial positions and different target door angles (see Fig. 8). Videos of all these motions can be found in the supplementary material.

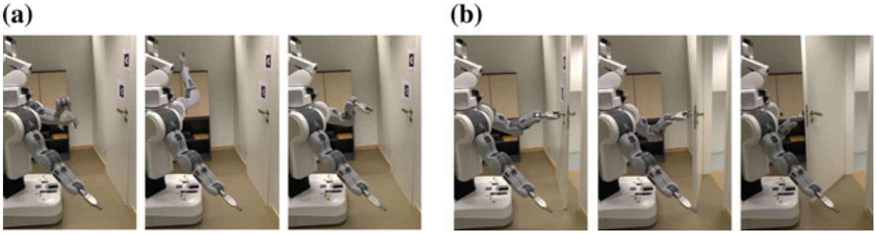


Fig. 8 These images show the generalization abilities of our approach. The pictures in (**a**) show different initial positions of the robot and the pictures in (**b**) show different final door angle positions. After learning the weight parameter $\boldsymbol{w}^{\star}$ with inverse KKT it was possible to generalize to all these instances of the door opening task

6 Conclusion

In this paper we introduced inverse KKT motion optimization, an inverse optimal control method for learning cost functions for constrained motion optimization problems. Our formulation is focused on finite horizon optimal control problems for tasks that include contact with the environment. The resulting method is based on the KKT conditions that the demonstrations should fulfill. For a typical linear parameterization of cost functions this leads to a convex problem; in the general case it is implemented as a 2nd order optimization problem, which leads to a fast convergence rate. We demonstrated the method in a real robot experiment of opening a door that involved contact with the environment. In our future research we plan to further automate and simplify the skill acquisition process. Thereby, one goal is to extend the proposed method to be able to handle demonstrations that are not recorded on the robot body. Another goal is to further improve the skill with reinforcement learning.

References

1. Abbeel, P., Coates, A., Ng, A.Y.: Autonomous helicopter aerobatics through apprenticeship learning. Int. J. Robot. Res. **29**(13), 1608–1639 (2010)
2. Albrecht, S., Ramirez-Amaro, K., Ruiz-Ugalde, F., Weikersdorfer, D., Leibold, M., Ulbrich, M., Beetz, M.: Imitating human reaching motions using physically inspired optimization principles. In: Proceedings of HUMANOIDS (2011)
3. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009)
4. Calinon, S., Alizadeh, T., Caldwell, D.G.: On improving the extrapolation capability of task-parameterized movement models. In: Proceedings of IROS (2013)
5. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
6. Jetchev, N., Toussaint, M.: TRIC: task space retrieval using inverse optimal control. Auton. Robot. **37**(2), 169–189 (2014)
7. Kalakrishnan, M., Pastor, P., Righetti, L., Schaal, S.: Learning objective functions for manipulation. In: Proceedings of ICRA (2013)

8. Kolter, J.Z., Abbeel, P., and Ng, A.Y. Hierarchical apprenticeship learning with application to quadruped locomotion. In: NIPS (2008)
9. Kroemer, O., van Hoof, H., Neumann, G., Peters, J.: Learning to predict phases of manipulation tasks as hidden states. In: Proceedings of ICRA (2014)
10. Levine, S., Koltun, V.: Continuous inverse optimal control with locally optimal examples. In: Proceedings of ICML (2012)
11. Mombaur, K., Truong, A., Laumond, J.-P.: From human to humanoid locomotionan inverse optimal control approach. Auton. Robot. **28**(3), 369–383 (2010)
12. Muhlig, M., Gienger, M., Steil, J.J., Goerick, C.: Automatic selection of task spaces for imitation learning. In: Proceedings of IROS (2009)
13. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)
14. Paraschos, A., Daniel, C., Peters, J., Neumann, G.: Probabilistic Movement Primitives. In: NIPS (2013)
15. Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., Schaal, S.: Skill learning and task outcome prediction for manipulation. In: Proceedings of ICRA (2011)
16. Puydupin-Jamin, A.-S., Johnson, M., Bretl, T.: A convex approach to inverse optimal control and its application to modeling human locomotion. In: Proceedings of ICRA (2012)
17. Rückert, E.A., Neumann, G., Toussaint, M., Maass, W.: Learned graphical models for probabilistic planning provide a new class of movement primitives. Front. Comput. Neurosci. **6**, 1–20 (2013)
18. Russell, S.: Learning agents for uncertain environments. In: Proceedings of the Conference on Computational Learning Theory (1998)
19. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philos. Trans. R. Soc. Lond. **358**, 537–547 (2003)
20. Stulp, F., Raiola, G., Hoarau, A., Ivaldi, S., Sigaud, O.: Learning compact parameterized skills with a single regression. In: Proceedings of HUMANOIDS (2013)
21. Theodorou, E., Buchli, J., Schaal, S.: A generalized path integral control approach to reinforcement learning. J. Mach. Learn. Res. **11**, 3137–3181 (2010)
22. Toussaint, M.:Newton methods for k-order markov constrained motion problems (2014). arXiv:1407.0414 [cs.RO]
23. Toussaint, M., Ratliff, N., Bohg, J., Righetti, L., Englert, P., Schaal, S.: Dual execution of optimized contact interaction trajectories. In: Proceedings of IROS (2014)
24. Zhifei, S., Joo, E.M.: A survey of inverse reinforcement learning techniques. Int. J. Intell. Comput. Cybern. **5**(3), 293–311 (2012)
25. Ziebart, B.D., Maas, A., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (2008)

Autonomously Acquiring Instance-Based Object Models from Experience

John Oberlin and Stefanie Tellex

1 Introduction

Robotics will assist us at childcare, help us cook, and provide service to doctors, nurses, and patients in hospitals. Many of these tasks require a robot to robustly perceive and manipulate objects in its environment, yet robust object manipulation remains a challenging problem. Transparent or reflective surfaces that are not visible in IR or RGB make it difficult to infer grasp points [23], while emergent physical dynamics cause objects to slip out of the robot's gripper; for example, a heavy object might slip to the ground during transport unless the robot grabs it close to the center of mass. Instance-based approaches that focus on specific objects can have higher accuracy but usually require training by a human operator, which is time consuming and can be difficult for a non-expert to perform [18, 19, 31]. Existing approaches for autonomously learning 3D object models often rely on expensive iterative closest point-based methods to localize objects, which are susceptible to local minima and take time to converge [16].

To address this problem, we take an instance-based approach, exploiting the robot's ability to collect its own training data. Although this approach does not generalize to novel objects, it enables experience with the object to directly improve the robot's performance during future interactions, analogous to how mapping an environment improves a robot's ability later to localize itself. After this data collection process is complete, the robot can quickly and reliably manipulate the objects. Our first contribution is an approach that enables a robot to achieve the high accuracy of instance-based methods by autonomously acquiring training data on a per object basis. Our grasping and perception pipeline uses standard computer vision techniques to perform data collection, feature extraction, and training. It uses active visual servoing for localization, and only uses depth information at scan time. Because our

J. Oberlin (✉) · S. Tellex
Brown University, Providence, RI, USA
e-mail: oberlin@cs.brown.edu

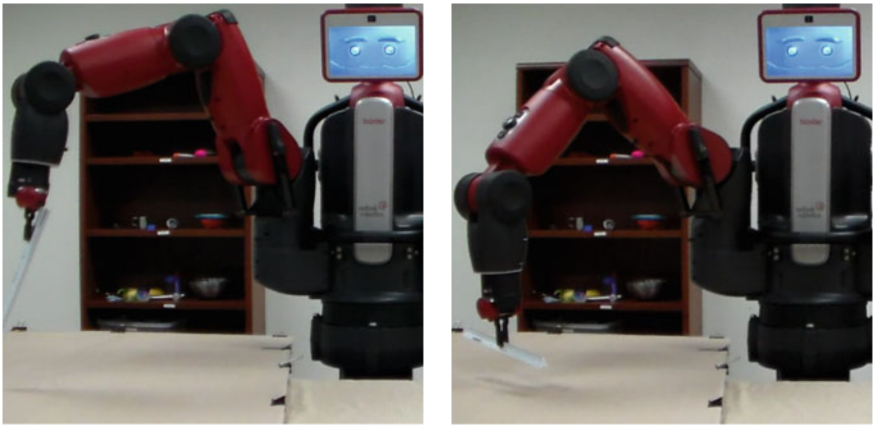
© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_5

camera can move with seven degrees of freedom, the robot collects large quantities of view-based training data, so that straightforward object detection approaches perform with high accuracy. This framework enables a Baxter robot to detect, classify, and manipulate many objects.

However, limitations in sensing and complex physical dynamics cause problems for some objects. Our second contribution addresses these limitations by enabling a robot to learn about an object through exploration and adapt its grasping model accordingly. We frame the problem of model adaptation as identifying the best arm for an N-armed bandit problem [41] where the robot aims to minimize simple regret after a finite exploration period [3]. Existing algorithms for best arm identification require pulling all the arms as an initialization step [1, 5, 26]; in the case of identifying grasp points, where each grasp takes more than 15 s and there are more than 1000 potential arms, this is a prohibitive expense. To avoid pulling all the arms, we present a new algorithm, Prior Confidence Bound, based on Hoeffding races [27]. In our approach, the robot pulls arms in an order determined by a prior, which allows it to try the most promising arms first. It can then autonomously decide when to stop by bounding the confidence in the result. Figure 1 shows the robot's performance before and after training on a ruler; after training it grasps the object in the center, improving the success rate.

Our evaluation demonstrates that our scanning approach enables a Baxter robot with no additional sensing to detect, localize, and pick up a variety of household objects. Further, our adaptation step improves the overall pick success rate from 55 to 75% on our test set of 30 household objects, shown in Fig. 6 (Fig. 2).


(a) Before learning, the ruler slips.

(b) After learning, the robot picks it up.

Fig. 1 Before learning, the robot grasps the ruler near the end, and it twists out of the gripper and falls onto the table; after learning, the robot successfully grasps near the ruler's center of mass

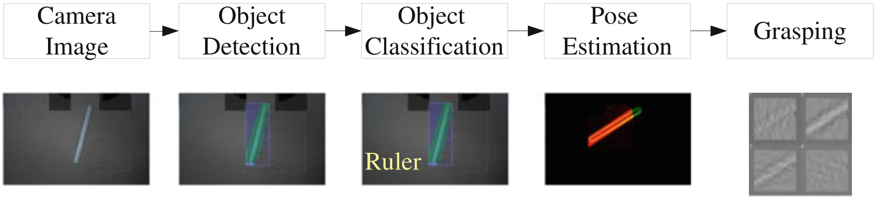


Fig. 2 Results at each phase of the grasping pipeline

2 Grasping System

Our object detection and pose estimation pipeline uses conventional computer vision algorithms in a simple software architecture to achieve a frame rate of about 2 Hz for object detection and pose estimation. Object classes consist of specific object instances rather than general object categories. Using instance recognition means we cannot reliably detect categories, such as "mugs," but the system will be much better able to detect, localize, and grasp the specific instances, e.g. particular mugs, for which it does have models.

Our detection pipeline runs on stock Baxter with one additional computer. The pipeline starts with video from the robot's wrist cameras, proposes a small number of candidate object bounding boxes in each frame, and classifies each candidate bounding box as belonging to a previously encountered object class. When the robot moves to attempt a pick, it uses detected bounding boxes and visual servoing to move the arm to a position approximately above the target object. Next, it uses image gradients to servo the arm to a known position and orientation above the object. Because we can know the gripper's position relative to the object, we can reliably collect statistics about the success rate of grasps at specific points on the object.

2.1 Object Detection

The goal of the object detection component is to extract bounding boxes for objects in the environment from a relatively uniform background. The robot uses object detection to identify regions of interest for further processing. The input of the object detection component is an image, I; the output is a set of candidate bounding boxes, B. Our object detection approach uses a modified Canny algorithm which terminates before the usual non-maximal suppression step [4]. We start by converting I to a YCbCr opponent color representation. Then we apply 5×5 Sobel derivative filters [39] to each of the three channels and keep the square gradient magnitude. We take a convex combination of the three channels, where Cb and Cr and weighted the same and more heavily than Y because Y contains more information about shadows and specular information, which adds noise. Finally we downsample, apply the two Canny thresholds, and find connected components. We generate a candidate bounding box for each remaining component by taking the smallest box which contains the component. We throw out boxes which do not contain enough visual data to classify. If a box is contained entirely within another, we discard it.

2.2 Object Classification

The object classification module takes as input a bounding box, B, and outputs a label for that object, c, based on the robot's memory. This label is used to identify the object and look up other information about the object for grasping further down the pipeline. For each object c we wish to classify, we gather a set of example crops E_c which are candidate bounding boxes (derived as above) which contain c. We extract dense SIFT features [21] from all boxes of all classes and use k-means to extract a visual vocabulary of SIFT features [40]. We then construct a Bag of Words feature vector for each image and augment it with a histogram of colors which appear in that image. The augmented feature vector is incorporated into a k-nearest-neighbors model which we use to classify objects at inference [40]. We use kNN because our automated training process allows us to acquire as much high-quality data as necessary to make the model work well, and kNN supports direct matching to this large dataset.

2.3 Pose Estimation

For pose estimation, we require a crop of the image gradient of the object at a specific, known pose. As during the bounding box proposal step, we approximate the gradient using 5×5 Sobel derivative filters [39], but we use a different convex combination of the channels which focuses even less on the Y channel. Camera noise in the color channels is significant. To cope with the noise, we marginalize the gradient estimate over several frames taken from the same location, providing a much cleaner signal which matches more robustly. To estimate pose, we rotate our training image and find the closest match to the image currently recorded from the camera, as detected and localized via the pipeline in Sects. 2.1 and 2.2. Once the pose is determined, we have enough information to attempt any realizable grasp, but our system focuses on crane grasps.

Lighting changes between scan and pick time can make it difficult to perform image matching. In order to match our template image with the crop observed at pick time, we remove the mean from the two images and L^2 normalize them. Removing the mean provides invariance to bias, and normalizing introduces invariance to scaling. These both help to provide compensation for inadequacies in the lighting.

2.4 Grasping

During grasp point identification, we use a model of the gripper to perform inference over a depth map of the object. The grasp model scores each potential grasp according to a linear model of the gripper in order to estimate grasp success. A default algorithm picks the highest-scoring grasp point using hand designed linear filters, but frequently this point is not actually a good grasp, because the object might slip out of the robot's gripper or part of the object may not be visible in IR. The input to this module is the 3D pose of the object, and the output is a grasp point (x, y, θ); at this point we employ only crane grasps rather than full 3D grasping, where θ is the angle which the gripper assumes for the grasp. This approach is not a state-of-the-art but is simple to implement and works well for many objects in practice. In Sect. 4, we describe how we can improve grasp proposals from experience, which can in principle use any state-of-the-art grasp proposal system as a prior.

3 Autonomously Acquiring Object Models

An object model in our framework consists of the following elements, which the robot autonomously acquires:

- cropped object templates (roughly 200), $t^1 \dots t^K$
- depth map, D, which consists of a point cloud, $(x, y, z, r, g, b)^{i,j}$.
- cropped gradient templates at different heights, $t_0 \dots t^M$

The robot collects gradient images by servoing to the center of the extracted bounding box for the object, described in Sect. 2.1, and then recording a gradient image at several different heights. It records each image for several frames to average away noise from the camera. Gradient images for the ruler appear in Fig. 3c.

Next, it acquires a depth image. Normally, this image could be acquired from an RGB-D sensor such as the Kinect. However, in order to make our approach run on a stock Baxter robot with no additional sensing, we acquire a depth scan using Baxter's IR sensor, turning the arm into a seven degree of freedom, one-pixel depth sensor. After acquiring visual and IR models for the object at different poses of the arm, we acquire view-based object models for detection and classification by moving the camera around the object, extracting bounding boxes from the images, and storing the resulting crops. Figure 3a shows RGB images automatically collected for one object in our dataset.

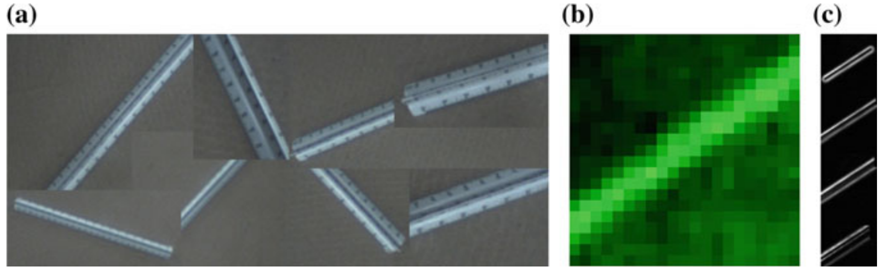


Fig. 3 Autonomously acquired object model. **a** Cropped RGB images, **b** Depth map, **c** Aerial gradient images at four different heights

4 Bandit-Based Model Adaptation

The formalization we contribute treats grasp learning as an N-armed bandit problem. Formally, the agent is given an N-armed bandit, where each arm pays out 1 with probability μ_i and 0 otherwise. The agent's goal is to identify a good arm (with payout $\geq k$) with probability c (e.g., 95% confidence that this arm is good) as quickly as possible. As soon as it has done this, it should terminate. The agent is also given a prior π on the arms so that it may make informed decisions about which grasps to explore.

4.1 Algorithm

Our algorithm, Prior Confidence Bound, iteratively chooses the arm with the highest observed (or prior) success rate but whose probability of being below k is less than a threshold. It then tries that arm, records the results, and updates its estimate of the probability of success, μ_i. If it is sufficiently certain that the arm's payout is either very good or very bad, it terminates; otherwise, it continues pulling the arm to collect more information. Pseudo-code appears in Algorithm 1. Our algorithm takes as input π, an estimate of the payout of each arm, as well as δ_{accept} and δ_{reject}, parameters controlling how certain it must be to accept or reject an arm. We need to estimate the probability that the true payout probability, μ_i, is greater than the threshold, c, given the observed number of successes and failures:

$$\Pr(\mu_i > k|S, F) \tag{1}$$

We can compute this probability using the law of total probability:

$$\Pr(\mu_i > k|S, F) = 1 - \int_0^k \Pr(\mu_i = \mu|S, F)d\mu \tag{2}$$

```
PriorConfidenceBound($\pi$, $k$, $\delta_{accept}$, $\delta_{reject}$, $maxTries$)
Initialize $S_0 \dots S_n$ to 0
Initialize $F_0 \dots F_n$ to 0
$totalTries \leftarrow 0$
while $true$ do
    $totalTries \leftarrow totalTries + 1$
    Set $M_0 \dots M_n$ to $\frac{S_0}{S_0+F_0} \cdots \frac{S_n}{S_n+F_n}$
    $j \leftarrow bestValidArm$;    // set j to the arm with $p_{below} < \delta_{reject}$ that
    has the highest marginal value
    $r \leftarrow sample(arm_j)$
    if $r = 1$ then
        $S_j \leftarrow S_j + 1$
    else
        $F_j \leftarrow F_j + 1$
    end
    $p_{below} \leftarrow \int_0^k \Pr(\mu_j = \mu | S_j, F_j) d\mu$
    $p_{above} \leftarrow \int_k^1 \Pr(\mu_j = \mu | S_j, F_j) d\mu$
    $p_{threshold} \leftarrow \int_{k-\varepsilon}^{k+\varepsilon} \Pr(\mu_j = \mu | S_j, F_j) d\mu$
    if $p_{above} \geq \delta_{accept}$ then
        return $j$ ;                                   // accept this arm
    else if $p_{threshold} \geq \delta_{accept}$ then
        return $j$ ;                                   // accept this arm
    else if $totalTries \geq maxTries$ then
        return $maxI$ ;  // return the arm with the best marginal value
        out of those that were tried
    else
        pass ;                                          // keep trying
    end
end
```

Algorithm 1: Prior Confidence Bound for Best Arm Identification

We assume a beta distribution on μ:

$$= \int_k^1 \mu^S (1-\mu)^F d\mu \tag{3}$$

This integral is the CDF of the beta distribution, and is called the regularized incomplete beta function [32].

The prior controls both the order that arms are explored and when the algorithm moves on to the next arm. If the prior is optimistic (i.e., overestimates μ_i), the algorithm will more quickly move on to the next arm if it encounters failures, because its empirical estimate of μ_i will be lower than the estimate from the prior of the next arm to pull. If the prior is pessimistic, the algorithm will be more likely to continue pulling an arm even if it encounters failure. By using a prior that incorporates probability estimates, it enables our algorithm to exploit information from the underlying grasp proposal system and make more informed decisions about when to move on.

4.2 Simulation

We simulate our algorithm by creating a sequence of 50 bandits, where each arm i pays out at a rate uniformly sampled between 0 and 1. For algorithms that incorporate prior knowledge, we sample a vector of estimates for each μ_i from a beta distribution with $\alpha = \beta = 1 + e * \mu_i$ where e controls the entropy of the sampling distribution.

To compare to a well-known baseline, we assess the performance of Thompson Sampling [41] in the fixed budget setting, although this algorithm minimizes total regret, including regret during training, rather than simple regret. Second, we compare to a Uniform baseline that pulls every arm equally until the budget is exceeded. This baseline corresponds to the initialization step in UCB or the confidence bound algorithms in Chen et al. [5]. The state-of-the-art CLUCB algorithm from Chen et al. [5] would not have enough pulls to finish this initialization step in our setting. Finally, we show the performance of three versions of Prior Confidence Bound, one with an uninformed prior ($e = 0$, corresponding to Hoeffding races [27]), one quite noisy with $e = 1$(but still informative), the other less noisy $e = 5$).

We run each experiment for 100 trials, and report 95% confidence intervals around the algorithm's simple regret. For Thompson Sampling and Uniform, which always use all trials in their budget, we report performance at each budget level; for Prior Confidence Bound, we report the mean number of trials the algorithm took before halting, also at 95% confidence intervals.

Results appear in Fig. 4. Thompson Sampling always uses all trials in its budget and improves performance as larger budgets are available. The Uniform method fails

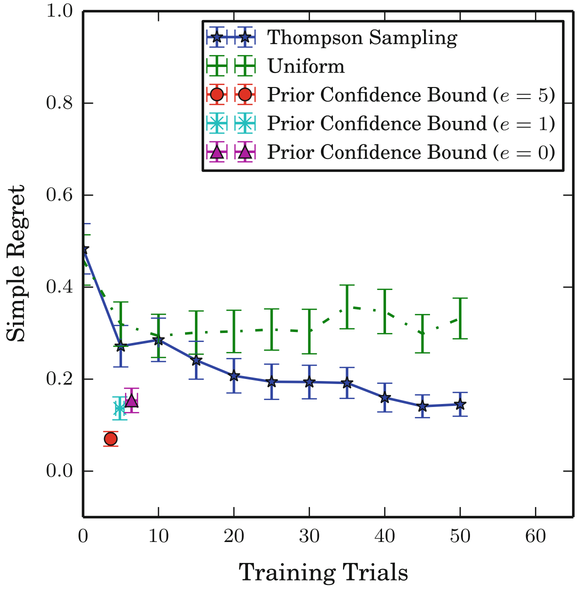


Fig. 4 Results comparing our approach to various baselines in simulation

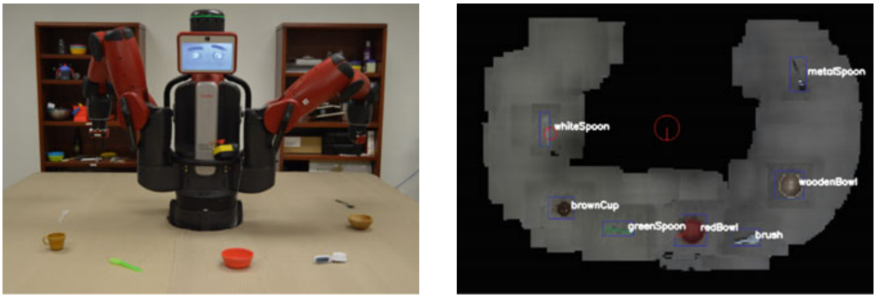
to find the optimal arm because there is not enough information when pulling each arm once. All variants of Prior Confidence Bound outperform these baselines, but as more prior information is incorporated, regret decreases. Even with a completely uninformed prior, bounding the confidence and decided when to stop improves performance over Thompson sampling or a uniform baseline, but the approach realizes significant further improvement with more prior knowledge.

5 Evaluation

The aim of our evaluation is to assess the ability of the system to acquire visual models of objects which are effective for grasping and object detection. We implemented our approach on a Baxter robot; a video showing our training and grasping pipeline is available at https://www.youtube.com/watch?v=xfH0B3g782Y.

5.1 Mapping

Mapping assesses the ability of our robot to accurately localize and label objects in a tabletop scene. The robot maps the scene by maintaining a data structure with an entry for each cell in its work space, at approximately 1 cm resolution, and recording the last time that cell was observed by the camera. It samples a new cell uniformly from the set of oldest cells, moves to that location, then runs the detection step. If it sees an object, it servos to that object, then adds the object's bounding box and class label to the map. By running object classification directly over the object, we obtain high-accuracy recognition rates, since the robot sees the object from a consistent pose. Figure 5 shows the map created in this way for a tabletop scene. We compute



(a) Tabletop scene with objects

(b) Map created for the scene by scanning with both arms.

Fig. 5 The robot actively scans the table and maps its environment using learned models (Descriptive object labels are provided by hand)

colors for each cell by taking the average of camera pixel colors at that cell, given the current table height.

5.2 Pick and Place

The robot acquired visual and RGB-D models for 30 objects using our autonomous learning system. The objects used in our evaluation appear in Fig. 6. We manually verified that the scans were accurate, and set the following parameters: height above the object for the IR scan (to approximately 2cm); this height could be acquired automatically by doing a first coarse IR scan following by a second IR scan 2 cm above the tallest height, but we set it manually to save time. Additionally we set the height of the arm for the initial servo to acquire the object. After acquiring visual and IR models for the object at different poses of the arm, the robot performed the bandit-based adaptation step using Algorithm 1. The algorithm requires a scoring of candidate grasps, π, which we provided using the linear filter described in Sect. 2.4. In principle, we could use any state-of-the-art system for proposing grasps in the prior (e.g., [9, 34, 36]); if the proposed grasp is successful, the algorithm will quickly terminate. Otherwise it will continue trying to pick until it finds a successful grasp.

After the robot detects an initially successful grab, it shakes the object vigorously to ensure that it would not fall out during transport. After releasing the object and moving away, the robot checks to make sure the object is not stuck in its gripper. If the object falls out during shaking or does not release properly, the grasp is recorded as a failure. If the object is stuck, the robot pauses and requests assistance before proceeding.

Most objects have more than one pose in which they can stand upright on the table. If the robot knocks over an object, the model taken in the reference pose is no longer meaningful. Thus, during training, we monitored the object and returned

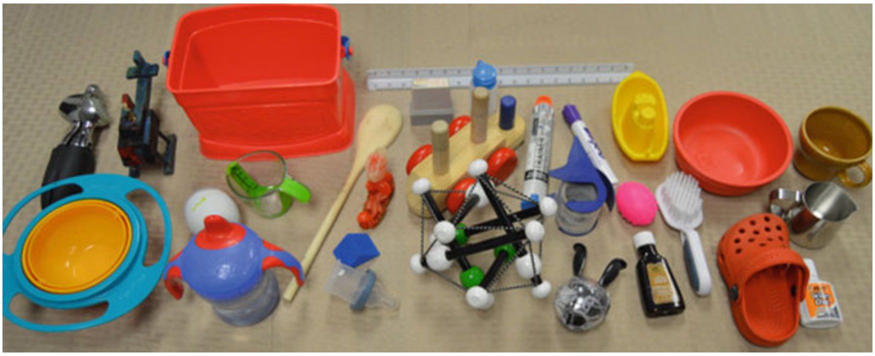
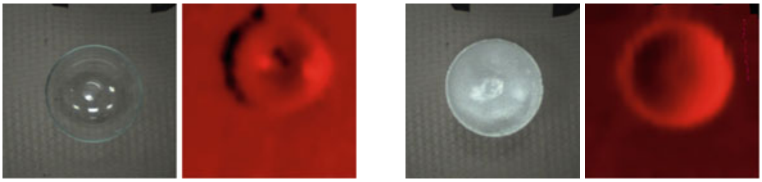


Fig. 6 The objects used in our evaluation, sorted from worst performing (*left*) to best performing (*right*)



(a) The depth map for the glass bowl without a contrast agent applied is noisy and leads to poor grasp hypothesis.

(b) When a contrast agent is applied, the depth map is clean and grasp performance improves. In our method, the contrast agent is only needed at scan time, not at pick time.

Fig. 7 Depth map for a transparent object with and without a contrast agent

it to the reference pose whenever the robot knocked it over. In the future, we aim to incorporate multiple components in the models which will allow the robot to cope with objects whose pose can change during training.

5.2.1 Contrast Agents

Many objects are challenging to grasp for our approach because they are transparent or dark in IR; grasping objects in spite of these issues is a challenging problem that remains an active area of research [11, 28]. As in Lysenkov et al. [23], which used paint to construct models of objects, apply a contrast agent to make the object visible in IR. Because our approach only uses depth information at scan time, and not at grasping time, and is heavily instance-based, it enables us to we apply a temporary coating for the IR scan and then remove it for learning visual models and later interaction with the object. (We found that hair spray coated with a layer of flour gives good performance, but can be easily removed.) Figure 7 shows an object and IR scan with and without a contrast agent. This demonstrates the advantage of our view-based and instance-based approach, which uses IR only during the single, initial scan, and not at inference time; once a good scan is obtained, grasps can be proposed at any time in the future and high-quality grasps can be learned through training. For the glass bowl in Fig. 7, pick accuracy improved from 0/10 without a contrast agent to 8/10 with a contrast agent applied during the IR scan only and removed before the pick trials.

5.2.2 Bandit-Based Adaptation

We evaluate our bandit-based adaptation step by allowing the robot to try grasps on the object until it either halts or reaches a maximum of 50 grasp attempts. Our algorithm used an accept threshold of 0.7, reject confidence of 0.95 and epsilon of 0.2. These parameters result in a policy that rejects a grasp after one failed try, and accepts if the first three picks are successful. Different observations of success and failure will cause the algorithm to try the grasp more to determine the true probability of success.

We report the performance of the robot at picking using the learned height for servoing, but without grasp learning, then the number of trials used for grasp learning by our algorithm, and finally the performance at picking using the learned grasp location and orientation. These results appear in Table 1.

Some objects significantly improved performance. Objects that improved typically had some feature that prevented our grasping model from working. For example, the triangular block failed with the prior grasp because the gripper slid over the sloped edges and pinched the block out of its grippers. The robot tried grasps until it found one that targeted the sides that were parallel to the grippers, resulting in a flush grasp, significantly improving accuracy. For the round salt shaker, the robot first attempted to grab the round plastic dome, but the gripper is not wide enough for this grasp. It tried grasps until it found one on the handle that worked reliably.

Objects such as the round salt shaker and the bottle top are on the edge of tractability for thorough policies such as Thompson sampling. Prior Confidence Bound, on the other hand, rejects arms quickly so as to make these two objects train in relatively short order while bringing even more difficult objects such as the sippy cup and big syringe into the realm of possibility. It would have taken substantially more time and picks for Thompson sampling to reject the long list of bad grasps on the sippy cup before finding the good ones.

The garlic press is a geometrically simple object but quite heavy compared to the others. The robot found a few grasps which might have been good for a lighter object, but it frequently shook the press out of its grippers when confirming grasp quality. The big syringe has some good grasps which are detected well by the prior, but due to its poor contrast and transparent tip, orientation servoing was imprecise and the robot was unable to learn well due to poor signal. What improvement did occur was due to finding a grasp which consistently deformed the bulb into a grippable shape regardless of the perceived orientation of the syringe. We observed similar problems with the clear pitcher and icosahedron.

Objects that failed to improve fall into several categories. For some, performance was already high, so there was not much room to move or a reasonable grasp was accepted quickly without waiting to find a better one. A common failure mode for poorly performing objects was failure to accurately determine the position and orientation through visual servoing. If the grasp map cannot be localized accurately, significant noise is introduced because the map does not correspond to the same physical location on the object at each trial. For example, there is only about a 5 mm difference between the width of the dragon and the width of the gripper; objects such as these would benefit from additional servo iterations to increase localization precision. If we double the number of iterations during fine grained servoing we can more reliably pick it, but this would either introduce another parameter in the system (iterations) or excessively slow down other objects which are more tolerant to error.

Table 1 Results from the robotic evaluation of Prior Confidence Bound, sorted by pick success rate. All objects either maintained or improved performance after learning except for one: Dragon

Low-performing objects			
	Before learning	During learning	After learning
Garlic press	0/10	8/50	2/10
Helicopter	2/10	8/39	3/10
Gyro bowl	0/10	5/15	3/10
Big syringe	1/10	13/50	4/10
Sippy cup	0/10	6/50	4/10
Clear pitcher	4/10	3/4	4/10
Red bucket	5/10	3/3	5/10
Wooden spoon	7/10	3/3	7/10
Dragon	8/10	5/6	7/10
Triangle block	0/10	3/13	7/10
Bottle top	0/10	5/17	7/10
Ruler	6/10	5/12	7/10
High-Performing Objects			
Epipen	8/10	4/5	8/10
Icosahedron	7/10	7/21	8/10
Stamp	8/10	3/3	8/10
Blue salt shaker	6/10	5/10	8/10
Wooden train	4/10	11/24	8/10
Packing tape	9/10	3/3	9/10
Purple marker	9/10	3/3	9/10
Round salt shaker	1/10	4/16	9/10
Toy egg	8/10	4/5	9/10
Yellow boat	9/10	5/6	9/10
Vanilla	5/10	4/5	9/10
Brush	10/10	3/3	10/10
Red bowl	10/10	3/3	10/10
Shoe	10/10	3/3	10/10
Whiteout	10/10	3/3	10/10
Metal pitcher	6/10	7/12	10/10
Mug	3/10	3/4	10/10
Syringe	9/10	6/9	10/10

6 Related Work

Pick-and-place has been studied since the early days of robotics [2, 22]. Initial systems relied on models of object pose and end effector pose being provided to the algorithm, and simply planned a motion for the arm to grasp. Modern approaches use object recognition systems to estimate pose and object type, then libraries of grasps either annotated or learned from data [8, 29, 36]. These approaches attempt to create systems that can grasp arbitrary objects based on learned visual features or known 3D configuration.

Collecting training sets is an expensive process and is not accessible to the average user in a non-robotics setting. If the system does not work for the user's particular application, there is no easy way for it to adapt or relearn. Our approach enables the robot to autonomously acquire more information to increase robustness at detecting and manipulating the specific object that is important to the user at the current moment. Other approaches that focus on object discovery and manipulation fail to combine a camera that moves with an end to end system that learns to recognize objects and improves grasp success rates through experience [6, 15, 24, 37].

We formalize the problem as an N-armed bandit [41] where the robot aims to perform best arm identification [1, 5], or alternatively, to minimize simple regret after a finite exploration period [3]. Audibert and Bubeck [1] explored best arm identification in a fixed budget setting; however a fixed budget approach does not match our problem, because we would like the robot to stop sampling as soon as it has improved performance above a threshold. We take a fixed confidence approach as in Chen et al. [5], but their fixed confidence algorithm begins by pulling each arm once, a prohibitively expensive operation on our robot. Instead our algorithm estimates confidence that one arm is better than another, following Hoeffding races [27] but operating in a confidence threshold setting that incorporates prior information. By incorporating prior information, our approach achieves good performance without being required to pull all the arms. Kaufmann et al. [12] describe Bayesian upper confidence bounds for bandit problems but do not use simple regret, with a training period followed by an evaluation period. Additionally these approaches do not provide a stopping criterion, to decide when to move to the next object.

By formalizing grasp identification as a bandit problem, we are able to leverage existing strategies for inferring the best arm. Our system brings together key techniques in autonomous data collection and online learning for persistent robotic systems to establish a baseline grasping system which we show to be useful and extensible. Nguyen and Kemp [30] learn to manipulate objects such as a light switch or drawer with a similar self-training approach. Our work autonomously learns visual models to detect, pick, and place previously unencountered rigid objects by actively selecting the best grasp point with a bandit based system, rather than acquiring models for the manipulation of articulated objects. We rely on the fixed structure of objects rather than learning how to deal with structure that can change during manipulation.

Hudson et al. [10] used active perception to create a grasping system capable of carrying out a variety of complex tasks. Using feedback is critical for good performance, but the model cannot adapt itself to new objects. Existing general purpose grasp algorithms achieve fairly good performance on novel objects but leave appreciable gaps which could be closed by using our system to learn from experience [9, 20, 25, 33, 34]. Kroemer et al. [17] also use reinforcement learning to choose where to grasp novel objects, operating in continuous state spaces. However their approach does not incorporate prior knowledge and requires forty or more trials to learn a good grasp; in contrast, because our approach incorporates prior knowledge, we often obtain improvement after trying only a few grasps.

Collet et al. [6] describe an approach for lifelong robotic object discovery, which infers object candidates from the robot's perceptual data. This system does not learn grasping models and does not actively acquire more data to recognize, localize, and grasp the object with high reliability. It could be used as a first-pass to our system, after which the robot uses an active method to acquire additional data enabling it to grasp the object. Some approaches integrate SLAM and moving object tracking to estimate object poses over time but have not been extended to manipulation [7, 35, 38, 42]. Crowd-sourced and web robotics have created large databases of objects and grasps using human supervision on the web [13, 14]. These approaches outperform automatically inferred grasps but still require humans in the loop. Our approach can incorporate human annotations in the form of the prior: if the annotated grasps work well, then the robot will quickly converge and stop sampling; if they are poor grasps, our approach will find better ones.

7 Conclusion

The contribution of this paper is a system for automatically acquiring instance-based models of objects using a Baxter robot. Using our approach, the robot scans objects and collects RGB and depth information, which it then uses to perform detection, classification, and grasping. We demonstrate that the robot can improve grasping performance through active exploration by formalizing the grasping problem as best arm identification on an N-armed bandit. This approach significantly improves the robot's success rate at grasping specific objects as it practices picking them.

A limitation of our system is the requirement that a target object be in a canonical upright position with respect to the table, leaving only one degree of freedom to describe its orientation and two for its position. In our evaluation, if the salt shaker fell down, we reset it to a randomized upright position. With this paradigm, if we want to be able to handle the salt shaker whether it is upright or on its side, we must train two models and use logic outside the system to identify the models as the same object. Our next goal is to automatically explore these object modes and acquire classification, localization, grasping, and transition models for them over a long period of time. This improvement will enable any Baxter robot to automatically scan objects for long periods of time.

A second limitation is that by taking an instance-based approach, knowledge obtained from interacting with one object does not generalize to another object. Our approach runs on a stock Baxter robot and does not require any additional sensing. We aim to release our software so that anyone with a Baxter can train models using our approach and automatically share their models through a common database. This approach will enable us to scale up and distribute the scanning effort, so that a very large corpus of instance-based models can be automatically collected. As more and more models are collected, containing RGB image crops, point clouds, and logs of grasp success rates at different geometries, this data set will provide a unique opportunity to train new category-based models for general detection and grasping, supplying well-annotated data of multiple views of many instances of individual objects.

Our system focuses on manipulation of small objects; however, objects in the environment have more affordances than just manipulation: bottles can be opened; light switches can be flipped; buttons can be pushed, and doors can be unlocked. We aim to expand our approach to instance-based semantic mapping of large-scale environments, so that the robot can interactively learn about features of its environment such as drawers, door knobs, and light switches. By taking an instance-based approach, the robot can automatically create robust detectors for object features and fix up any problems through interaction with the environment. This approach will create a true semantic map of the environment, including affordances of objects.

References

1. Audibert, J.-Y., Bubeck, S: Best arm identification in multi-armed bandits. In: COLT-23th Conference on Learning Theory-2010, pp. 13–p (2010)
2. Brooks, R.A.: Planning collision-free motions for pick-and-place operations. Int. J. Robot. Res. **2**(4), 19–44 (1983)
3. Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in multi-armed bandits problems. In: Algorithmic Learning Theory, pp. 23–37. Springer (2009)
4. Canny, John: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 679–698 (1986)
5. Chen, Shouyuan, Lin, Tian, King, Irwin: Michael R Lyu, and Wei Chen, pp. 379–387. Combinatorial pure exploration of multi-armed bandits. In, Advances in Neural Information Processing Systems (2014)
6. Collet, Alvaro, Xiong, Bo, Gurau, Corina, Hebert, Martial, Srinivasa, Siddhartha S.: Herbdisc: towards lifelong robotic object discovery. Int. J. Robot. Res. (2014). doi:10.1177/0278364914546030
7. Gallagher, G., Srinivasa, S.S., Andrew Bagnell, J., Ferguson, D.: Gatmo: a generalized approach to tracking movable objects. In: IEEE International Conference on Robotics and Automation, 2009. ICRA'09, pp. 2043–2048. IEEE (2009)
8. Goldfeder, C., Ciocarlie, M., Dang, H., Allen, P.K.: The columbia grasp database. In: IEEE International Conference on Robotics and Automation, 2009. ICRA'09, pp. 1710–1716. IEEE (2009)
9. Gori, I., Pattacini, U., Tikhanoff, V., and Giorgio Metta. Three-finger precision grasp on incomplete 3d point clouds. In: 2014 IEEE international conference on robotics and automation (ICRA), pp. 5366–5373. IEEE (2014)
10. Hudson, N., Howard, T., Ma, J., Jain, A., Bajracharya, M., Myint, S., Kuo, C., Matthies, L., Backes, P, Hebert, P et al.: End-to-end dexterous manipulation with deliberate interactive

estimation. In: 2012 IEEE international conference on robotics and automation (ICRA), pp. 2371–2378. IEEE (2012)

11. Ivo Ihrke, Kiriakos N Kutulakos, Hendrik PA Lensch, Marcus Magnor, and Wolfgang Heidrich. State of the art in transparent and specular object reconstruction. In *EUROGRAPHICS 2008 STAR–STATE OF THE ART REPORT*. Citeseer, 2008
12. Kaufmann, E., Cappé, O., Garivier, A.: On bayesian upper confidence bounds for bandit problems. In: International Conference on Artificial Intelligence and Statistics, pp. 592–600 (2012)
13. Kent, D., Chernova, S.: Construction of an object manipulation database from grasp demonstrations. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), pp. 3347–3352. IEEE (2014)
14. Kent, D., Behrooz, M., Chernova, S.: Crowdsourcing the construction of a 3d object recognition database for robotic grasping. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 4526–4531. IEEE (2014)
15. Kraft, D., Detry, R., Pugeault, N., Baseski, E., Guerin, F., Piater, J.H., Kruger, N.: Development of object and grasping knowledge by robot exploration. IEEE Trans. Auton. Mental Dev. **2**(4), 368–383 (2010)
16. Krainin, Michael, Henry, Peter, Ren, Xiaofeng, Fox, Dieter: Manipulator and object tracking for in-hand 3D object modeling. Int. J. Robot. Res. **30**(11), 1311–1327 (2011)
17. Kroemer, O.B., Detry, R., Piater, J., Peters, J.: Combining active learning and reactive control for robot grasping. Robot. Auton. Syst. **58**(9), 1105–1116 (2010)
18. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1817–1824. IEEE (2011)
19. Lai, K., Bo, L., Ren, X., Fox, D.: A scalable tree-based approach for joint object and pose recognition. In: Twenty-Fifth Conference on Artificial Intelligence (AAAI) (2011)
20. Le, Q.V., Kamm, D., Kara, A.F., Ng, A.Y.: Learning to grasp objects with multiple contact points. In: 2010 IEEE International Conference on Robotics and Automation (ICRA). pp. 5062–5069. IEEE (2010)
21. Lowe, D.G.: Object recognition from local scale-invariant features. In: The proceedings of the seventh IEEE international conference on Computer vision, vol. 2, pp. 1150–1157. IEEE (1999)
22. Lozano-Pérez, Tomás, Jones, Joseph L., Mazer, Emmanuel, O'Donnell, Patrick A.: Task-level planning of pick-and-place robot motions. IEEE Comput. **22**(3), 21–29 (1989)
23. Lysenkov, I., Eruhimov, V., Bradski, G.: Recognition and pose estimation of rigid transparent objects with a kinect sensor. Robotics. p. 273 (2013)
24. Lyubova, N., Filliat, D., Ivaldi, S.: Improving object learning through manipulation and robot self-identification. In: 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1365–1370. IEEE (2013)
25. Maldonado, A., Klank, U., Beetz, M.: Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In: IROS, pp. 2586–2591 (2010)
26. Mannor, S., Tsitsiklis, J.N.: The sample complexity of exploration in the multi-armed bandit problem. J. Mach. Learn. Res. **5**, 623–648 (2004)
27. Maron, O., Moore, A.W.: Hoeffding races: Accelerating model selection search for classification and function approximation. Robotics Institute, pp. 263 (1993)
28. Mériaudeau, Fabrice, Rantoson, Rindra, Fofi, David, Stolz, Christophe: Review and comparison of non-conventional imaging systems for three-dimensional digitization of transparent objects. J. Electron. Imaging **21**(2), 021105 (2012)
29. Morales, A., Chinellato, E., Fagg, A.H., Pasqual del Pobil, A.: Experimental prediction of the performance of grasp tasks from visual features. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003), vol. 4, pp. 3423–3428. IEEE (2003)
30. Nguyen, H., Kemp, C.C.: Autonomously learning to visually detect where manipulation will succeed. Auton. Robots **36**(1–2), 137–152 (2014)
31. Object Recognition Kitchen. http://wg-perception.github.io/object_recognition_core/ (2014)

32. Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W. (eds.): NIST Handbook of Mathematical Functions. Cambridge University Press, New York, NY (2010)
33. Rao, D., Le, Q.V., Phoka, T., Quigley, M., Sudsang, A., Ng, A.Y.: Grasping novel objects with depth segmentation. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2578–2585. IEEE (2010)
34. Sahbani, A., El-Khoury, S.: A hybrid approach for grasping 3d objects. In: IEEE/RSJ International Conference on Intelligent Robots and System, IROS 2009, pp. 272–1277. IEEE (2009)
35. Salas-Moreno, R.F., Newcombe, R.A., Strasdat, H., Kelly, P.H.J., Davison, A.J.: Slam++: simultaneous localisation and mapping at the level of objects. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1352–1359. IEEE (2013)
36. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. Int. J. Robot. Res. 27(2), 157–173 (2008)
37. Schiebener, D., Schill, J., Asfour, T: Discovery, segmentation and reactive grasping of unknown objects. In: Humanoids, pp. 71–77 (2012)
38. Selvatici, A.H.P., Costa, A.H.R.: Object-based visual slam: How object identity informs geometry (2008)
39. Sobel, I.: An isotropic 3x3x3 volume gradient operator. Technical report, Hewlett-Packard Laboratories (1995)
40. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer (2010)
41. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, pp. 285–294 (1933)
42. Wang, Chieh-Chih, Thorpe, Charles, Thrun, Sebastian, Hebert, Martial, Durrant-Whyte, Hugh: Simultaneous localization, mapping and moving object tracking. Int. J. Robot. Res. **26**(9), 889–916 (2007)

Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning

Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel and Ken Goldberg

1 Introduction

Recorded demonstrations of robot-assisted minimally invasive surgery (RMIS) have been used for surgical skill assessment [7], development of finite state machines for automation [13, 25], learning from demonstration (LfD) [29], and calibration [22]. Intuitive Surgical's *da Vinci* robot facilitated over 570, 000 procedures in 2014 [11]. There are proposals to record all of Intuitive's RMIS procedures similar to flight data recorders ("black boxes") in airplanes [12], which could lead to a proliferation of data. While these large datasets have the potential to facilitate learning and autonomy; the length and variability of surgical trajectories pose a unique challenge. Each surgical trajectory may represent minutes of multi-modal observations, may contain loops (failures and repetitions until achieving the desired result), and even

S. Krishnan (✉) · S. Patil · P. Abbeel · K. Goldberg
EECS, UC Berkeley, Berkeley, CA, USA
e-mail: sanjaykrishnan@berkeley.edu

S. Patil
e-mail: sachinpatil@berkeley.edu

P. Abbeel
e-mail: pabbeel@berkeley.edu

K. Goldberg
e-mail: goldberg@berkeley.edu

A. Garg (✉) · K. Goldberg
IEOR, UC Berkeley, Berkeley, CA, USA
e-mail: animesh.garg@berkeley.edu

C. Lea · G. Hager
Computer Science Department, The Johns Hopkins University, Baltimore, MD, USA
e-mail: clea1@jhu.edu

G. Hager
e-mail: hager@cs.jhu.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_6

identical procedures can vary due to differences in the environment. In this setting, typical techniques for establishing spatial and temporal correspondence that employ continuous deformations can be unreliable (e.g., Dynamic Time Warping [14] and spline-based registration [31]).

Segmentation of a task into sub-tasks can be valuable since individual segments are less complex, less variable, and allow for easier detection and rejection of outliers. Trajectory segmentation in robotics is an extensively studied problem [4, 5, 16, 20, 21, 26, 30]. However, prior work in robotic surgery focuses on the *supervised* problem setting, either requiring manual segmentation of example trajectories or using a set of pre-defined primitive motions called "surgemes" [21, 30, 36]. Manual labelling requires specifying consistent segmentation criteria and applying these criteria to across demonstrations, which can be time-consuming and unreliable. Similarly, it can be challenging to manually construct a dictionary of primitives at the correct level of abstraction.

Outside of surgery, there have been several proposals for *unsupervised* segmentation [5, 16, 20, 26], where the criteria are learned from data without a pre-defined dictionary. The salient feature of these approaches is a clustering or local regression model to identify locally similar states. Inherently, the success of unsupervised approaches is dependent on how well the demonstrations match the assumptions of the model (i.e., the definition of "similar"). In surgery, the tissue and environment may vary greatly between demonstrations making it difficult to directly compare different trajectories. Our insight is that while the trajectories may be very different, there can be a common latent structure in the demonstrations that can be learned from the data. Segmentation can be performed with respect to these latent parameters leading to robust segmentation criteria.

Transition State Clustering (TSC) combines hybrid dynamical system theory with Bayesian statistics to learn such a structure. We model demonstrations as repeated realizations of an unknown noisy switched linear dynamical system [8]. TSC identifies changes in local linearity in each demonstration, and leans a model to infer regions of the state-space at which switching events occur. These regions are generated from a hierarchical nonparametric Bayesian model, where the number of regions are determined by a Dirichlet Process and the shape of the regions are determined by a mixture of multivariate Gaussian random variables. A series of merging and pruning steps (controlled by user-specified parameters δ and ρ respectively) remove outlier transition states.

We also explore how to use the video data that accompanies kinematic data in surgical demonstration recordings. In this work, we explore improving segmentation through hand-engineered visual features. We manually label the video stream with two features: a binary variable identifying object grasp events and a scalar variable indicating surface penetration depth. We evaluate results with and without these visual features (Sect. 5.4). In future work, we will explore automated methods to construct featurized representations of the video data.

2 Related Work and Background

Motion Primitives and Skill Learning: Motion primitives are segments that discretize the *action*-space of a robot, and can facilitate faster convergence in LfD [10, 23, 27]. On the other hand, TSC discretizes the *state*-space, which can be interpreted as segmenting a task and not a trajectory. Much of the initial work in motion primitives considered manually identified segments, but recently, Niekum et al. [26] proposed learning the set of primitives from demonstrations using the Beta-Process Autoregressive Hidden Markov Model (BP-AR-HMM). Calinon et al. [2] also build on a large corpus of literature of unsupervised skill segmentation including the task-parameterized movement model [6], and GMMs for segmentation [5].

The ideas in Niekum et al. inspire the results presented in this work, namely, the use of Bayesian non-parametric models for segmentation and switched linear models. Unlike Niekum et al. and our work, Calinon et al. do not employ Bayesian non-parametrics or multimodal data. In Niekum et al. transition events are only dependent on the current dynamical regime, and in TSC they also depend on the current state (as illustrated in Fig. 1 with a dashed line). In this paper, we extend this line of work with non-parametric clustering on a GMM based model, and account for specific challenges such as looping and inconsistency in surgical demonstrations.

Handling Temporal Inconsistency: The most common model for handling demonstrations that have varying temporal characteristics is Dynamic Time Warping (DTW). However, DTW is a greedy dynamic programming approach which assumes that trajectories are largely the same up-to some smooth temporal deformations. When there are significant variations due to looping or spurious states, this model can give unreliable results [14], as shown by our results.

Another common model for modeling temporal inconsistencies is the Finite State Markov Chain model with Gaussian Mixture Emissions (GMM+HMM) [1, 3, 15, 34]. These models, impose a probabilistic grammar on the segment transitions and can be learned with an EM algorithm. However, they can be sensitive to hyper-parameters such as the number of segments and the amount of data [32]. The problem of robustness in GMM+HMM (or closely related variants) has been addressed using down-weighting transient states [17] and sparsification [9]. In TSC, we explore whether it is sufficient to know *transition states* without having to fully parametrize a Markov Chain for accurate segmentation. In Fig. 1, we compare the graphical models of GMM+HMM, and TSC. The TSC model applies Dirichlet Process priors to automatically set the number of hidden states (regimes).

The TSC algorithm finds spatially and temporally similar transition states across demonstrations, and it does not have to model correlations between switching events–in essence, using the current state as a sufficient statistic for switching behavior. On the other hand, the typical GMM+HMM model learns a full $k \times k$ transition matrix. Consequently, we empirically find that the TSC model is robust to noise and temporal variation, especially for a small number of demonstrations.

Surgical Task Recognition: Surgical robotics has largely studied the problem of supervised segmentation using either segmented examples or a pre-defined dictionary of motions (similar to motion primitives). For example, given manually segmented videos, Zappella et al. [36] use features from both the videos and kinematic data to classify surgical motions. Simiarly, Quellec et al. [28] use manually segmented examples as training for segmentation and recognition of surgical tasks based on archived cataract surgery videos. The dictionary-based approaches are done with a domain-specific set of motion primitives for surgery called "surgemes". A number of works (e.g., [19, 21, 33, 35]), use the surgemes to bootstrap learning segmentation.

3 Problem Setup and Model

The TSC model is summarized by the hierarchical graphical model in the previous section (Fig. 1). Here, we formalize each of the levels of the hierarchy and describe the assumptions in this work.

Dynamical System Model: Let $\mathscr{D} = \{d_i\}$ be the set of demonstrations where each d_i is a trajectory $\mathbf{x}(t)$ of fully observed robot states and each state is a vector in $\mathbb{R}^d$. We model each demonstration as a switched linear dynamical system. There is a finite set of $d \times d$ matrices $\{A_1, \ldots, A_k\}$, and an i.i.d zero-mean additive Gaussian Markovian noise process $W(t)$ which accounts for noise in the dynamical model:

$$\mathbf{x}(t+1) = A_i \mathbf{x}(t) + W(t) : A_i \in \{A_1, \ldots, A_k\}$$

Transitions between regimes are instantaneous where each time t is associated with exactly one dynamical system matrix $1, \ldots, k$.

Transition States and Times: Transition states are defined as the last states before a dynamical regime transition in *each* demonstration. Each demonstration d_i follows a

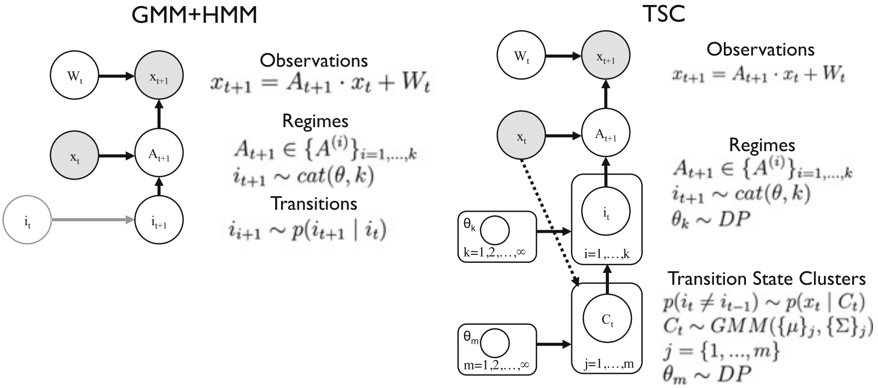

GMM+HMM
W_t
x_t+1
x_t
A_t+1
i_t
i_t+1
Observations
$x_{t+1} = A_{t+1} \cdot x_t + W_t$
Regimes
$A_{t+1} \in \{A^{(i)}\}_{i=1,\ldots,k}$
$i_{t+1} \sim cat(\theta, k)$
Transitions
$i_{i+1} \sim p(i_{t+1} \mid i_t)$
TSC
W_t
x_t+1
x_t
A_t+1
i_t
i=1,...,k
θ_k
k=1,2,...,∞
C_t
j=1,...,m
θ_m
m=1,2,...,∞
Observations
$x_{t+1} = A_{t+1} \cdot x_t + W_t$
Regimes
$A_{t+1} \in \{A^{(i)}\}_{i=1,\ldots,k}$
$i_{t+1} \sim cat(\theta, k)$
$\theta_k \sim DP$
Transition State Clusters
$p(i_t \neq i_{t-1}) \sim p(x_t \mid C_t)$
$C_t \sim GMM(\{\mu\}_j, \{\Sigma\}_j)$
$j = \{1, \ldots, m\}$
$\theta_m \sim DP$

Fig. 1 **a** A finite-state Hidden Markov Chain with Gaussian Mixture Emissions (GMM + HMM), and **b** TSC model. TSC uses Dirichlet Process Priors and the concept of transition states to learn a robust segmentation switched linear dynamical system model, therefore there is a time series of regimes $A(t)$ associated with each demonstration. Consequently, there will be times t at which $A(t) \neq A(t+1)$.

We model the occurrence of these events as a stochastic process conditioned on the current state. Switching events are governed by a latent function of the current state $S : \mathscr{X} \mapsto \{0, 1\}$, and we have noisy observations of switching events $\widehat{S}(x(t)) = S(x(t) + Q(t))$, where $Q(t)$ is a i.i.d noise process. Thus, across all demonstrations, the observed switching events induce a probability density $f(x)$ over the state space $\mathscr{X}$. In this paper, we focus on the problem where $f(x)$ is a Mixture of Gaussian densities.

Transition State Clusters: Across all demonstrations, we are interested in aggregating nearby (spatially and temporally) transition states together. The goal of *transition state clustering* is to find a mixture model for f that approximately recovers the true latent function S. Consequently, a transition state cluster is defined as a clustering of the set of transition states across all demonstrations; partitioning these transition states into m non-overlapping similar groups:

$$\mathscr{C} = \{C_1, C_2, \ldots, C_m\}$$

Every U_i can be represented as a sequence of integers indicating that transition states assignment to one of the transition state clusters $U_i = [1, 2, 4, 2]$.

Consistency: We assume, demonstrations are *consistent*, meaning there exists a non-empty sequence of transition states $\mathscr{U}^*$ such that the partial order defined by the elements in the sequence (i.e., s_1 happens before s_2 and s_3) is satisfied by every U_i. For example,

$$U_1 = [1, 3, 4], U_2 = [1, 1, 2, 4], \mathscr{U}^* = [1, 4]$$

A counter example,

$$U_1 = [1, 3, 4], U_2 = [2, 5], \mathscr{U}^* \text{ no solution}$$

Intuitively, this condition states that there have to be a consistent ordering of actions over all demonstrations up to some additional regimes (e.g., spurious actions).

Loops: Loops are common in surgical demonstrations. For example, a surgeon may attempt to insert a needle 2–3 times. When demonstrations have varying amounts of retrials it is challenging. In this work, we assume that these loops are modeled as repeated transitions between transition state clusters, which is justified in our experimental datasets, for example,

$$U_1 = [1, 3, 4], U_2 = [1, 3, 1, 3, 1, 3, 4], \mathscr{U}^* = [1, 3, 4]$$

Our algorithm will *compact* these loops together into a single transition.

Minimal Solution: Given a consistent set of demonstrations, that have additional regimes and loops, the goal of the algorithm is to find a *minimal solution*, $\mathscr{U}^*$ that is loop-free and respects the partial order of transitions in all demonstrations.

Given a set of demonstrations $\mathscr{D}$, *the Transition State Clustering problem is to find a set of transition state clusters* $\mathscr{C}$ *such that they represent a minimal parametrization of the demonstrations.*

Multi-modal TSC : This model can similarly be extended to states derived from sensing. Suppose at every time t, there is a feature vector $z(t)$. Then the augmented state of both the robot spatial state and the features denoted is:

$$\mathbf{x}(t) = \begin{pmatrix} x(t) \\ z(t) \end{pmatrix}$$

In our experiments, we worked the da Vinci surgical robot with two 7-DOF arms, each with 2 finger grippers. Consider the following feature representation which we used in our experiments:

1. *Gripper grasp*. Indicator that is 1 if there is an object between the gripper, 0 otherwise.
2. *Surface Penetration*. In surgical tasks, we often have a tissue phantom. This feature describes whether the robot (or something the robot is holding like a needle) has penetrated the surface. We use an estimate of the truncated penetration depth to encode this feature. If there is no penetration, the value is 0, otherwise the value of penetration is the robot's kinematic position in the direction orthogonal to the tissue phantom.

4 Transition State Clustering

In this section, we describe the hierarchical clustering process of TSC. This algorithm is a greedy approach to learning the parameters in the graphical model in Fig. 1. We decompose the hierarchical model into stages and fit parameters to the generative model at each stage. The full algorithm is described in Algorithm 1.

4.1 Background: Bayesian Statistics

One challenge with mixture models is hyper-parameter selection, such as the number of clusters. Recent results in Bayesian statistics can mitigate some of these problems. The basic recipe is to define a generative model, and then use Expectation Maximization to fit the parameters of the model to observed data. The generative model that we will use is called a mixture model, which defines a probability distribution that is a composite of multiple distributions.

One flexible class of mixture models are Gaussian Mixture Models (GMM), which are described generatively as follows. We first sample some c from a categorical distribution, one that takes on values from (1…K), with probabilities ϕ, where ϕ is a K dimensional simplex:

$$c \sim cat(K, \phi)$$

Then, given the event $\{c = i\}$, we specify a multivariate Gaussian distribution:

$$x_i \sim N(\mu_i, \Sigma_i)$$

The insight is that a stochastic process called the Dirichlet Process (DP) defines a distribution over discrete distributions, and thus instead we can draw samples of $cat(K, \phi)$ to find the most likely choice of K via EM. The result is the following model:

$$(K, \phi) \sim DP(H, \alpha) \qquad c \sim cat(K, \phi) \qquad X \sim N(\mu_i, \Sigma_i) \tag{1}$$

After fitting the model, every observed sample of $x \sim X$ will have a probability of being generated from a mixture component $P(x \mid c = i)$. Every observation x will have a most likely generating component. It is worth noting that each cluster defines an ellipsoidal region in the feature space of x, because of the Gaussian noise model $N(\mu_i, \Sigma_i)$.

We denote this entire clustering method in the remainder of this work as DP-GMM. We use the same model at multiple levels of the hierarchical clustering and we will describe the feature space at each level. We use a MATLAB software package to solve this problem using a variational EM algorithm [18].

4.2 Transition States Identification

The first step is to identify a set of transition states for each demonstration in $\mathscr{D}$. To do this, we have to fit a switched dynamic system model to the trajectories. Suppose there was only one regime, then this would be a linear regression problem:

$$\arg\min_A \|AX_t - X_{t+1}\|$$

where X_t and X_{t+1} are matrices where each column vector is corresponding $x(t)$ and $x(t+1)$. Moldovan et al. [24] showed that fitting a jointly Gaussian model to $n(t) = \binom{\mathbf{x}(t+1)}{\mathbf{x}(t)}$ is equivalent to Bayesian Linear Regression.

Therefore, to fit a switched linear dynamical system model, we can fit a Mixture of Gaussians (GMM) model to $n(t)$ via DP-GMM. Each cluster learned signifies a different regime, and co-linear states are in the same cluster. To find transition states, we move along a trajectory from $t = 1, \ldots, t_f$, and find states at which $n(t)$ is in a different cluster than $n(t+1)$. These points mark a transition between clusters (i.e., transition regimes).

4.3 Transition State Pruning

We consider the problem of outlier transitions, ones that appear only in a few demonstrations. Each of these regimes will have constituent vectors where each $n(t)$ belongs to a demonstration d_i. Transition states that mark transitions to or from regimes whose constituent vectors come from fewer than a fraction ρ demonstrations are *pruned.* ρ should be set based on the expected rarity of outliers. In our experiments, we set the parameter ρ to 80% and show the results with and without this step.

4.4 Transition State Compaction

Once we have transition states for each demonstration, and have applied pruning, the next step is to remove transition states that correspond to looping actions, which are prevalent in surgical demonstrations. We model this behavior as consecutive linear regimes repeating, i.e., transition from i to j and then a repeated i to j. We apply this step after pruning to take advantage of the removal of outlier regimes during the looping process. These repeated transitions can be compacted together to make a single transition.

The key question is how to differentiate between repetitions that are part of the demonstration and ones that correspond to looping actions–the sequence might contain repetitions not due to looping. To differentiate this, as a heuristic, we threshold the L2 distance between consecutive segments with repeated transitions. If the L2 distance is low, we know that the consecutive segments are happening in a similar location as well. In our datasets, this is a good indication of looping behavior. If the L2 distance is larger, then repetition between dynamical regimes might be happening but the location is changing.

Algorithm 1: The Transition State Clustering Algorithm

```
Input: $\mathscr{D}$, $\rho$ pruning parameter, and $\delta$ compaction parameter.
$n(t) = \binom{\mathbf{x}(t+1)}{\mathbf{x}(t)}$.
Cluster the vectors $n(t)$ using DP-GMM assigning each state to its most likely cluster.
*Transition states* are times when $n(t)$ is in a different cluster than $n(t+1)$.
Remove states that transition to and from clusters with less than a fraction of $p$ demonstrations.
Remove consecutive transition states when the L2 distance between these transitions is less than $\delta$.
Cluster the remaining transition states in the state space $\mathbf{x}(t+1)$ using DP-GMM.
Within each state-space cluster, sub-cluster the transition states temporally.
Output: A set $\mathscr{M}$ of clusters of transition states and the associated with each cluster a time interval of
    transition times.
```

For each demonstration, we define a segment $\mathbf{s}^{(j)}[t]$ of states between each transition states. The challenge is that $\mathbf{s}^{(j)}[t]$ and $\mathbf{s}^{(j+1)}[t]$ may have a different number of observations and may be at different time scales. To address this challenge, we apply Dynamic Time Warping (DTW). Since segments are locally similar up-to small time variations, DTW can find a most-likely time alignment of the two segments.

Let $\mathbf{s}^{(j+1)}[t^*]$ be a time aligned (w.r.t to $\mathbf{s}^{(j)}$) version of $\mathbf{s}^{(j+1)}$. Then, after alignment, we define the L_2 metric between the two segments:

$$d(j, j+1) = \frac{1}{T} \sum_{t=0}^{T} (\mathbf{s}^{(j)}[i] - \mathbf{s}^{(j+1)}[i^*])^2$$

when $d \leq \delta$, we compact two consecutive segments. δ is chosen empirically and a larger δ leads to a sparser distribution of transition states, and smaller δ leads to more transition states. For our needle passing and suturing experiments, we set δ to correspond to the distance between two suture/needle insertion points–thus, differentiating between repetitions at the same point versus at others.

However, since we are removing points from a time-series this requires us to adjust the time scale. Thus, from every following observation, we shift the time stamp back by the length of the compacted segments.

4.5 State-Space Clustering

After compaction, there are numerous transition states at different locations in the state-space. If we model the states at transition states as drawn from a GMM model:

$$x(t) \sim N(\mu_i, \Sigma_i)$$

Then, we can apply the DP-GMM again to cluster the state vectors at the transition states. Each cluster defines an ellipsoidal region of the state-space space.

4.6 Time Clustering

Without temporal localization, the transitions may be ambiguous. For example, in circle cutting, the robot may pass over a point twice in the same task. The challenge is that we cannot naively use time as another feature, since it is unclear what metric to use to compare distance between $\binom{\mathbf{x}(t)}{t}$. However a second level of clustering by time within each state-space cluster can overcome this issue. Within a state cluster, if we model the times which change points occur as drawn from a GMM $t \sim N(\mu_i, \sigma_i)$, and then we can apply DP-GMM to the set of times. We cluster time second because we observe that the surgical demonstrations are more consistent spatially than temporally. This groups together events that happen at similar times during the demonstrations. The result is clusters of states and times. Thus, a transition states m_k is defined as tuple of an ellipsoidal region of the state-space and a time interval.

5 Results

5.1 *Experiment 1. Synthetic Example of 2-Segment Trajectory*

In our first experiment, we segment noisy observations from a two regime linear dynamical system. Figure 2 illustrates examples from this system under the different

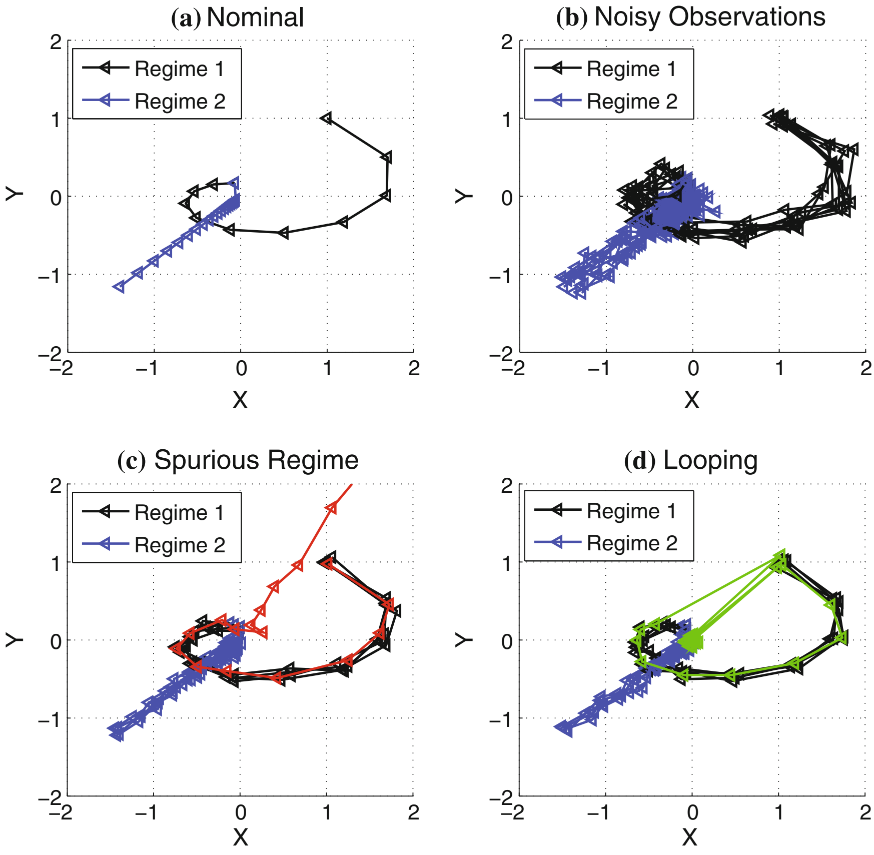


Fig. 2 **a** Observations from a dynamical system with two regimes, **b** Observations corrupted with Gaussian Noise, **c** Observations corrupted with a spurious inserted regime (*red*), and **d** Observations corrupted with an inserted loop(*green*)

types of corruption. Since there is a known a ground truth of two segments, we measure the precision (average fraction of observations in each segment that are from the same regime) and recall (average fraction of observations from each regime segmented together) in recovering these two segments. We can jointly consider precision and recall with the *F1 Score* which is the harmonic mean of the two. We compare three techniques against TSC: K-Means (only spatial), GMM+T (using time as a feature in a GMM), GMM+HMM (using an HMM to model the grammar). For the GMM techniques, we have to select the number of segments, and we experiment with $k = 1, 2, 3$ (i.e., a slightly sub-optimal parameter choice compared to $k = 2$). In this example, for TSC, we set the two user-specified parameters to $\delta = 0$ (merge all repeated transitions), and $\rho = 80\%$ (prune all regimes representing less than 80% of the demonstrations).

First, we generate 100 noisy observations (additive zero mean Gaussian noise) from the system without loops or spurious states–effectively only measuring the DP-GMM versus the alternatives. Figure 3a shows the F1-score as a function of the noise in the observations. Initially, for an appropriate parameter choice $k = 2$ both of the GMM-based methods perform well and at low noise levels the DP-GMM used by our work mirrors this performance. However, if the parameter is set to be $k = 3$, we see that the performance significantly degrades. $k = 1$ corresponds to a single segment which has a F1 score of 0.4 on all figures. The DP-GMM mitigates this sensitivity to the choice of parameter by automatically setting the value. Furthermore, as the noise increases, the 80% pruning of DP-GMM mitigates the effect of outliers leading to improved accuracy.

In Fig. 3b, we look at the accuracy of each technique as a function of the number of demonstrations. GMM+HMM has more parameters to learn and therefore requires more data. GMM+T converges the fastest, TSC requires slightly more data, and the GMM+HMM requires the most. In Fig. 3c, we corrupt the observations with spurious dynamical regimes. These are random transition matrices which replace one of the two dynamical regimes. We vary the rate at which we randomly corrupt the data, and measure the performance of the different segmentation techniques as a function of this rate. Due to the pruning, TSC gives the most accurate segmentation. The Dirichlet process groups the random transitions in different clusters and the small clusters are pruned out. On the other hand, the pure GMM techniques are less accurate since they are looking for exactly two regimes.

In Fig. 3d, introduce corruption due to loops and compare the different techniques. A loop is a step that returns to the start of the regime randomly, and we vary this random rate. For an accurately chosen parameter $k = 2$, for the GMM−HMM, it gives the most accurate segmentation. However, when this parameter is set poorly $k = 3$, the accuracy is significantly reduced. On the other hand, using time as a GMM feature (GMM+T) does not work since it does not know how to group loops into the same regime.

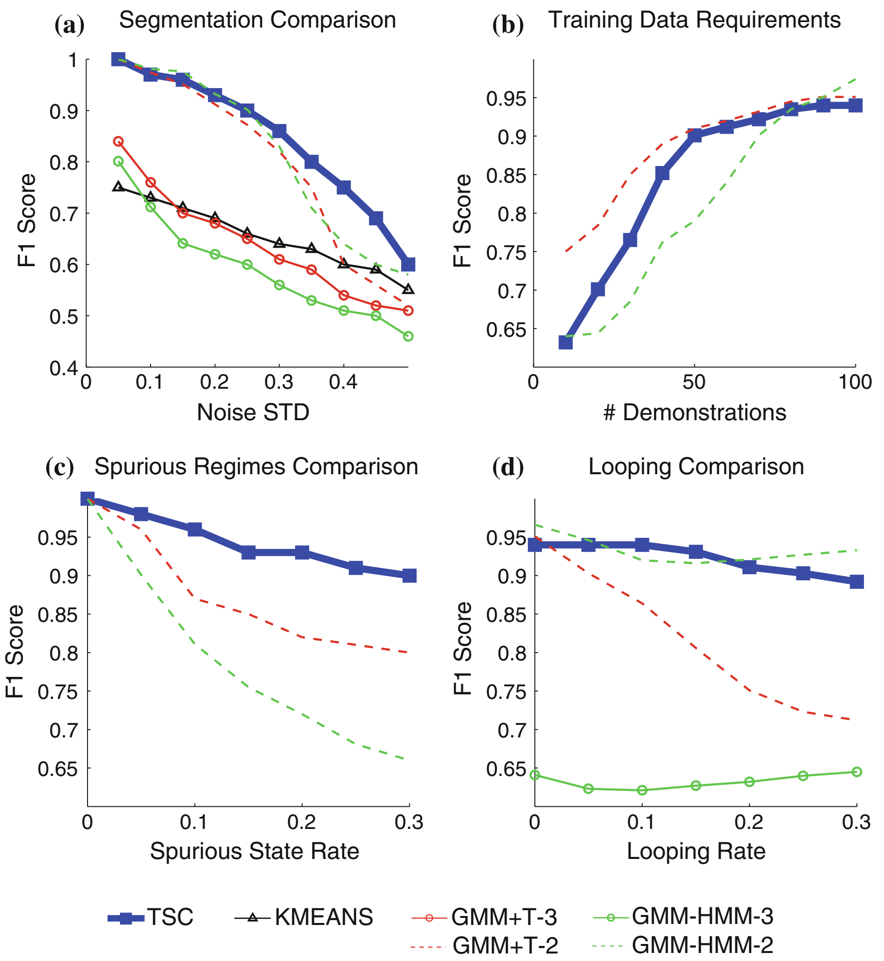


Fig. 3 Accuracy as a function of noise: TSC, K-Means, GMM+T (GMM with time), GMM+HMM (GMM with HMM). **a** The Dirichlet Process used in TSC reduces sensitivity to parameter choice and is comparable to GMM techniques using the optimal parameter choice, **b** HMM based methods need more training data as they have to learn transitions, **c** TSC clusters are robust to spurious regimes, and **d** TSC clusters are robust to loops–without having to know the regimes in advance

5.2 Surgical Experiments: Evaluation Tasks

We describe the three tasks used in our evaluation, and show manually segmented versions in Fig. 4. This will serve as ground truth when qualitatively evaluating our segmentation on real data.

Circle Cutting: In this task, we have a 5 cm diameter circle drawn on a piece of gauze. The first step is to cut a notch into the circle. The second step is to cut clockwise. Next, the robot transitions to the other side cutting counter clockwise. Finally, the

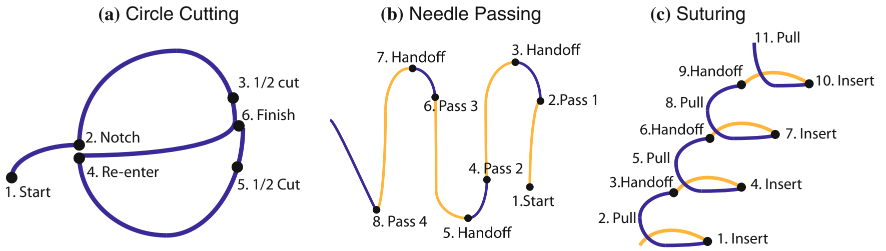


Fig. 4 Hand annotations of the three tasks: **a** circle cutting, **b** needle passing, and **c** suturing. *Right* arm actions are listed in *dark blue* and *left* arm actions are listed in *yellow*

robot finishes the cut at the meeting point of the two incisions. As the left arm's only action is maintain the gauze in tension, we exclude it from the analysis. In Fig. 4a, we mark 6 manually identified transitions points for this task from [25]: (1) start, (2) notch, (3) finish 1st cut, (4) cross-over, (5) finish 2nd cut, and (6) connect the two cuts. For the circle cutting task, we collected 10 demonstrations by non-experts familiar with operating the da Vinci Research Kit (dVRK).

We apply our method to the JIGSAWS dataset [7] consisting of surgical activity for human motion modeling. The dataset was captured using the da Vinci Surgical System from eight surgeons with different levels of skill performing five repetitions each of Needle Passing and Suturing.

Needle Passing: We applied our framework to 28 demonstrations of the needle passing task. The robot passes a needle through a loop using its right arm, then its left arm to pull the needle through the loop. Then, the robot hands the needle off from the left arm to the right arm. This is repeated four times as illustrated with a manual segmentation in Fig. 4b.

Suturing: Next, we explored 39 examples of a 4 throw suturing task (Fig. 4c). Using the right arm, the first step is to penetrate one of the points on right side. The next step is to force the needle through the phantom to the other side. Using the left arm, the robot pulls the needle out of the phantom, and then hands it off to the right arm for the next point.

5.3 Experiment 2. Pruning and Compaction

In Fig. 5, we highlight the benefit of pruning and compaction using the Suturing task as exemplar. First, we show the transition states without applying the compaction step to remove looping transition states (Fig. 5a). We find that there are many more transition states at the "insert" step of the task. Compaction removes the segments that correspond to a loop of the insertions. Next, we show the all of the clusters found by DP-GMM. The centroids of these clusters are marked in Fig. 5b. Many of these clusters are small containing only a few transition states. This is why we created the heuristic to prune clusters that do not have transition states from at least 80% of the demonstrations. In all, 11 clusters are pruned by this rule.

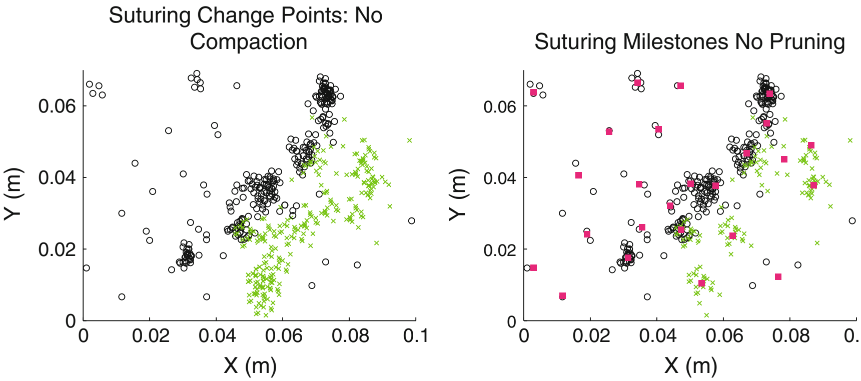


Fig. 5 We first show the transition states without compaction (in *black* and *green*), and then show the clusters without pruning (in *red*). Compaction sparsifies the transition states and pruning significantly reduces the number of clusters

5.4 Experiment 3. Can Vision Help?

In the next experiment, we evaluate TSC in a featurized state space that incorporates states derived from vision (Described in Sect. 5.1). We illustrate the transition states in Fig. 6 with and without visual features on the circle cutting task. At each point where the model transitions, we mark the end-effector (x, y, z) location. In particular, we show a region (red box) to highlight the benefits of these features. During the cross-over phase of the task, the robot has to re-enter the notch point and adjust to cut the other half of the circle. When only using the end-effector position, the locations where this transition happens is unreliable as operators may approach the entry from slightly different angles. On the other hand, the use of a gripper contact binary feature clusters the transition states around the point at which the gripper is in position and

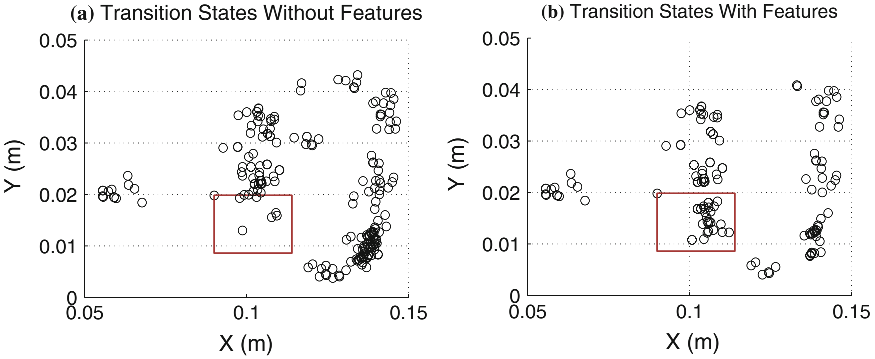


Fig. 6 **a** We show the transition states without visual features, **b** and with visual features. Marked in the *red* box is a set of transitions that cannot always be detected from kinematics alone

ready to begin cutting again. In the subsequent experiments, we use the same two visual features.

5.5 Experiment 4. TSC Evaluation

Circle Cutting: Figure 7a shows the transition states obtained from our algorithm. And Fig. 7b shows the TSC clusters learned (numbered by time interval midpoint). The algorithm found 8 clusters, one of which was pruned out using our $\rho = 80\%$ threshold rule.

The remaining 7 clusters correspond well to the manually identified transition points. It is worth noting that there is one extra cluster (marked $2'$), that does not correspond to a transition in the manual segmentation. At $2'$, the operator finishes a notch and begins to cut. While at a logical level notching and cutting are both penetration actions, they correspond to two different linear transition regimes due to the positioning of the end-effector. Thus, TSC separates them into different clusters even though a human annotator may not do so.

Needle Passing: In Fig. 8a, we plot the transition states in (x, y, z) end-effector space for both arms. We find that these transition states correspond well to the logical segments of the task (Fig. 4b). These demonstrations are noisier than the circle cutting demonstrations and there are more outliers. The subsequent clustering finds 9 (2 pruned). Next, Fig. 8b–c illustrate the TSC clusters. We find that again TSC learns a small parametrization for the task structure with the clusters corresponding well to the manual segments. However, in this case, the noise does lead to a spurious cluster (4 marked in green). One possible explanation is that the two middle loops are in close proximity and demonstrations contain many adjustments to avoid colliding with the loop and the other arm while passing the needle through leading to numerous transition states in that location.

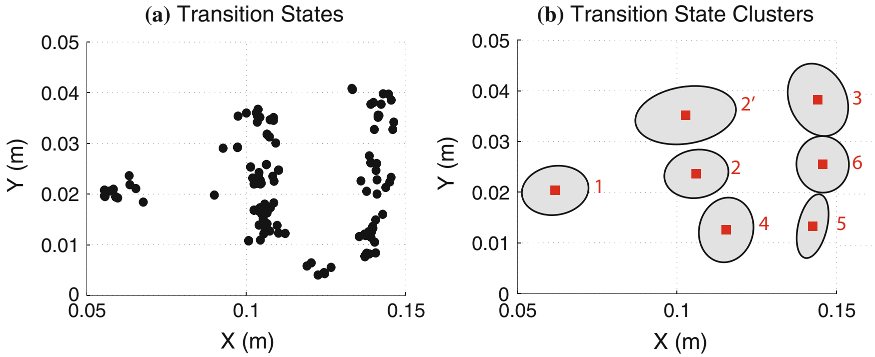


Fig. 7 **a** The transition states for the circle cutting task are marked in *black*. **b** The TSC clusters, which are clusters of the transition states, are illustrated with their 75% confidence ellipsoid

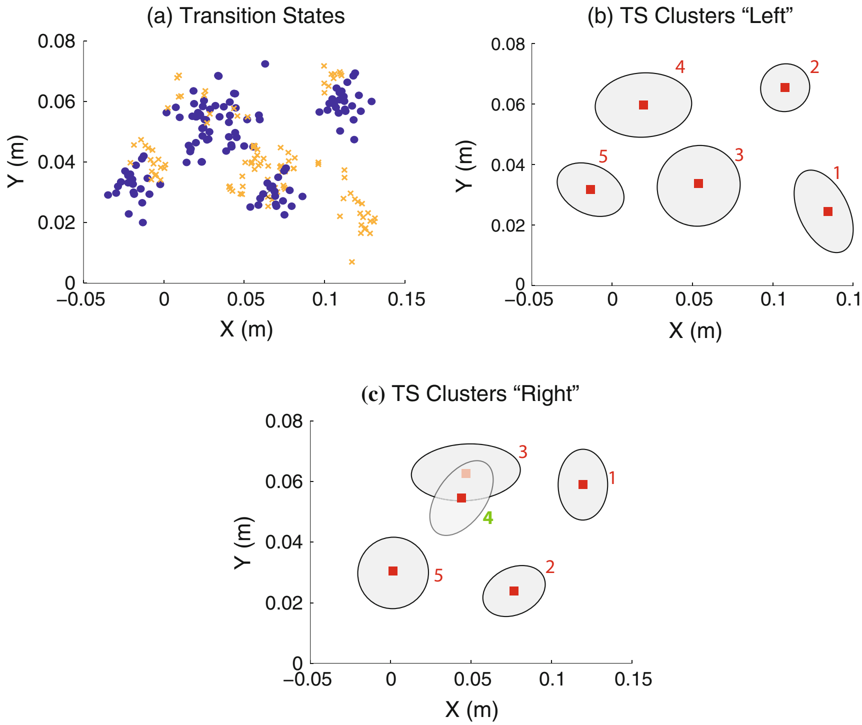


Fig. 8 **a** The transition states for the task are marked in *orange* (*left* arm) and *blue* (*right* arm). **b–c** The TSC clusters, which are clusters of the transition states, are illustrated with their 75% confidence ellipsoid for both arms

Suturing: In Fig. 9, we show the transition states and clusters for the suturing task. As before, we mark the left arm in orange and the right arm in blue. This task was far more challenging than the previous tasks as the demonstrations were inconsistent. These inconsistencies were in the way the suture is pulled after insertion (some pull to the left, some to the right, etc.), leading to transition states all over the state space. Furthermore, there were numerous demonstrations with looping behaviors for the left arm. In fact, the DP-GMM method gives us 23 clusters, 11 of which represent less than 80% of the demonstrations and thus are pruned (we illustrate the effect of the pruning in the next section). In the early stages of the task, the clusters clearly correspond to the manually segmented transitions. As the task progresses, we see that some of the later clusters do not.

5.6 *Experiment 5. Comparison to "Surgemes"*

Surgical demonstrations have an established set of primitives called surgemes, and we evaluate if segments discovered by our approach correspond to surgemes. In Table 1,

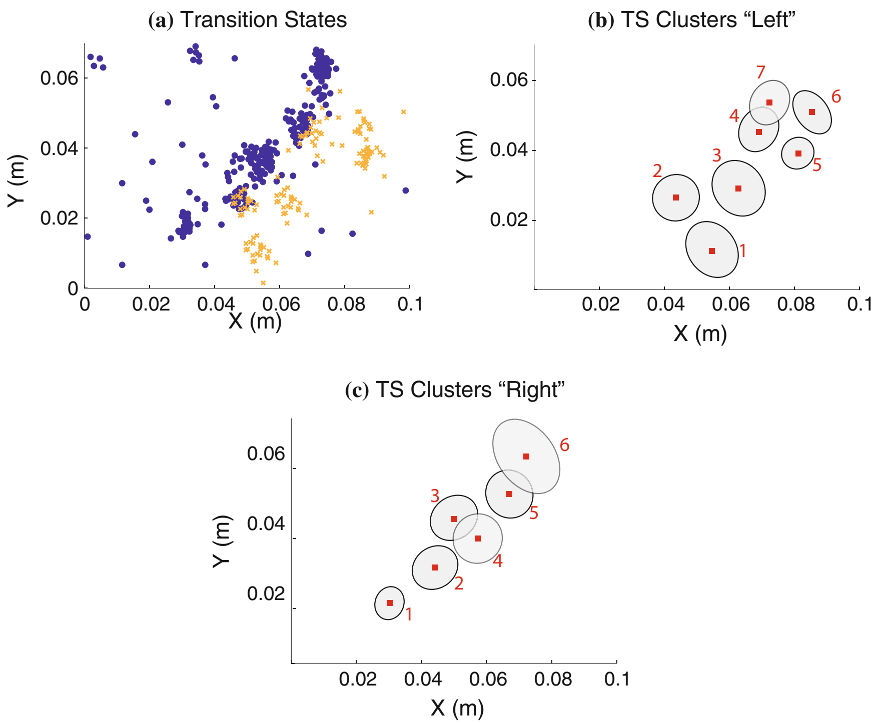


Fig. 9 **a** The transition states for the task are marked in *orange* (*left* arm) and *blue* (*right* arm). **b**–**c** The clusters, which are clusters of the transition states, are illustrated with their 75% confidence ellipsoid for both arms

Table 1 83 and 73% of transition clusters for needle passing and suturing respectively contained exactly one surgeme transition

	No. of surgeme segments	No. of segments + C/P	No. of TSC	TSC-Surgeme (%)	Surgeme-TSC (%)
Needle passing	19.3 ± 3.2	14.4 ± 2.57	11	83	74
Suturing	20.3 ± 3.5	15.9 ± 3.11	13	73	66

we compare the number of TSC segments for needle passing and suturing to the number of annotated surgeme segments. A key difference between our segmentation and number of annotated surgemes is our compaction and pruning steps. To account for this, we first select a set of surgemes that are expressed in most demonstrations (i.e., simulating pruning), and we also apply a compaction step to the surgeme segments. In case of consecutive appearances of these surgemes, we only keep the 1 instance of each for compaction. We explore two metrics: **TSC-Surgeme** the fraction of TSC clusters with only one surgeme switch (averaged over all demonstrations), and **Surgeme-TSC** the fraction of surgeme switches that fall inside exactly one TSC clusters.

6 Conclusion and Future Work

We presented Transition State Clustering (TSC), which leverages hybrid dynamical system theory and Bayesian statistics to robustly learn segmentation criteria. To learn these clusters, TSC uses a hierarchical Dirichlet Process Gaussian Mixture Model (DP-GMM) with a series of merging and pruning steps. Our results on a synthetic example suggest that the hierarchical clusters are more robust to looping and noise, which are prevalent in surgical data. We further applied our algorithm to three surgical datasets and found that the transition state clusters correspond well to hand annotations and transitions w.r.t motions from a pre-defined surgical motion vocabulary called surgemes.

There are a number of important open-questions for future work. First, we believe that the growing maturity of Convolutional Neural Networks can facilitate transition state clustering directly from raw data (e.g., pixels), as opposed to the features studied in this work, and is a promising avenue for future work. Next, we are also particularly interested in closing-the-loop and using segmentation to facilitate optimal control or reinforcement learning. Finally, we are also interested in relaxing the consistency and normality assumptions in our parameter inference algorithm.

Acknowledgements This research was supported in part by a seed grant from the UC Berkeley Center for Information Technology in the Interest of Society (CITRIS), by the U.S. National Science Foundation under Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems. This work has been supported in part by funding from Google and Cisco. We also thank Florian Pokorny, Jeff Mahler, and Michael Laskey.

References

1. Asfour, T., Gyarfas, F., Azad, P., Dillmann, R.: Imitation learning of dual-arm manipulation tasks in humanoid robots. In: 2006 6th IEEE-RAS International Conference on Humanoid Robots, pp. 40–47 (2006)
2. Calinon, S.: Skills learning in robots by interaction with users and environment. In: 2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 161–162. IEEE (2014)
3. Calinon, S., Billard, A.: Stochastic gesture production and recognition model for a humanoid robot. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems 2004, (IROS 2004), vol. 3, pp. 2769–2774 (2004)
4. Calinon, S., Halluin, F.D., Caldwell, D.G., Billard, A.G.: Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework. In: 9th IEEE-RAS International Conference on Humanoid Robots, 2009, Humanoids 2009, pp. 582–588. IEEE (2009)
5. Calinon, S., D'halluin, F., Sauser, E.L., Caldwell, D.G., Billard, A.G.: Learning and reproduction of gestures by imitation. IEEE Robot. Autom. Mag. **17**(2), 44–54 (2010)
6. Calinon, S., Bruno, D., Caldwell, D.G.: A task-parameterized probabilistic model with minimal intervention control. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 3339–3344 (2014)
7. Gao, Y., Vedula, S., Reiley, C., Ahmidi, N., Varadarajan, B., Lin, H., Tao, L., Zappella, L., Bejar, B., Yuh, D., Chen, C., Vidal, R., Khudanpur, S., Hager, G.: The jhu-isi gesture and skill

assessment dataset (jigsaws): a surgical activity working set for human motion modeling. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI) (2014)

8. Goebel, R., Sanfelice, R.G., Teel, A.: Hybrid dynamical systems. IEEE Control Syst. **29**(2), 28–93 (2009)
9. Grollman, D.H., Jenkins, O.C.: Incremental learning of subtasks from unsegmented demonstration. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 261–266. IEEE (2010)
10. Ijspeert, A., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Neural Information Processing Systems (NIPS), pp. 1523–1530 (2002)
11. Intuitive Surgical: Annual report (2014). http://investor.intuitivesurgical.com/phoenix.zhtml?c=122359&p=irol-IRHome
12. Johns Hopkins: Surgical robot precision. http://eng.jhu.edu/wse/magazine-winter-14/print/surgical-precision
13. Kehoe, B., Kahn, G., Mahler, J., Kim, J., Lee, A., Lee, A., Nakagawa, K., Patil, S., Boyd, W., Abbeel, P., Goldberg, K.: Autonomous multilateral debridement with the raven surgical robot. In: International Conference on Robotics and Automation (ICRA) (2014)
14. Keogh, E.J., Pazzani, M.J.: Derivative dynamic time warping. SIAM
15. Kruger, V., Herzog, D., Baby, S., Ude, A., Kragic, D.: Learning actions from observations. IEEE Robot. Autom. Mag. **17**(2), 30–43 (2010)
16. Krüger, V., Tikhanoff, V., Natale, L., Sandini, G.: Imitation learning of non-linear point-to-point robot motions using dirichlet processes. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 2029–2034. IEEE (2012)
17. Kulić, D., Nakamura, Y.: Scaffolding on-line segmentation of full body human motion patterns. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008, pp. 2860–2866. IEEE (2008)
18. Kurihara, K., Welling, M., Vlassis, N.A.: Accelerated variational dirichlet process mixtures. In: Advances in Neural Information Processing Systems, pp. 761–768 (2006)
19. Lea, C., Hager, G.D., Vidal, R.: An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks. In: WACV (2015)
20. Lee, S.H., Suh, I.H., Calinon, S., Johansson, R.: Autonomous framework for segmenting robot trajectories of manipulation task. Auton. Robots **38**(2), 107–141 (2014)
21. Lin, H., Shafran, I., Murphy, T., Okamura, A., Yuh, D., Hager, G.: Automatic detection and segmentation of robot-assisted surgical motions. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 802–810. Springer (2005)
22. Mahler, J., Krishnan, S., Laskey, M., Sen, S., Murali, A., Kehoe, B., Patil, S., Wang, J., Franklin, M., Abbeel, P.K.G.: Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression. In: International Conference on Automated Sciences and Engineering (CASE), pp. 532–539 (2014)
23. Manschitz, S., Kober, J., Gienger, M., Peters, J.: Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. Robot. Auton. Syst. **74**(5), 97–107 (2015)
24. Moldovan, T., Levine, S., Jordan, M., Abbeel, P.: Optimism-driven exploration for nonlinear systems. In: International Conference on Robotics and Automation (ICRA) (2015)
25. Murali, A., Sen, S., Kehoe, B., Garg, A., McFarland, S., Patil, S., Boyd, W., Lim, S., Abbeel, P., Goldberg, K.: Learning by observation for surgical subtasks: multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms. In: International Conference on Robotics and Automation (ICRA) (2015)
26. Niekum, S., Osentoski, S., Konidaris, G., Barto, A.: Learning and generalization of complex tasks from unstructured demonstrations. In: International Conference on Intelligent Robots and Systems (IROS), pp. 5239–5246. IEEE (2012)
27. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: International Conference on Robotics and Automation (ICRA), pp. 763–768. IEEE (2009)
28. Quellec, G., Lamard, M., Cochener, B., Cazuguel, G.: Real-time segmentation and recognition of surgical tasks in cataract surgery videos. IEEE Trans. Med. Imag. **33**(12), 2352–2360 (2014)

29. Reiley, C.E., Plaku, E., Hager, G.D.: Motion generation of robotic surgical tasks: learning from expert demonstrations. In: 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 967–970. IEEE (2010)
30. Rosen, J., Brown, J.D., Chang, L., Sinanan, M.N., Hannaford, B.: Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model. IEEE Trans. Biomed. Eng. **53**(3), 399–413 (2006)
31. Schulman, J., Ho, J., Lee, C., Abbeel, P.: Learning from demonstrations through the use of non-rigid registration
32. Tang, H., Hasegawa-Johnson, M., Huang, T.S.: Toward robust learning of the gaussian mixture state emission densities for hidden markov models. In: 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 5242–5245. IEEE (2010)
33. Tao, L., Zappella, L., Hager, G.D., Vidal, R.: Surgical gesture segmentation and recognition. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013, pp. 339–346. Springer (2013)
34. Vakanski, A., Mantegh, I., Irish, A., Janabi-Sharifi, F.: Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. IEEE Trans. Syst. Man Cybern. Part B Cybern. **42**(4), 1039–1052 (2012)
35. Varadarajan, B., Reiley, C., Lin, H., Khudanpur, S., Hager, G.: Data-derived models for segmentation with application to surgical assessment and training. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 426–434. Springer (2009)
36. Zappella, L., Bejar, B., Hager, G., Vidal, R.: Surgical gesture classification from video and kinematic data. Med. Image Analysis **17**(7), 732–745 (2013)

Robot Learning with Task-Parameterized Generative Models

Sylvain Calinon

1 Introduction

Robots are provided with an increasing number of sensors and actuators. This trend introduces original challenges in machine learning, where the sample size is often bounded by the cost of data acquisition, thus requiring models capable of handling wide-ranging data. Namely, models that can start learning from a small number of demonstrations, while still being able to continue learning when more data become available.

Robot learning from demonstration is one such field, which aims at providing end-users with intuitive interfaces to transfer new skills to robots. The challenges in robot learning can often be reinterpreted as designing appropriate domain-specific priors that can supply the required generalization capability from small training sets. The position adopted in this paper is that: (1) generative models are well suited for robot learning from demonstration because they can treat recognition, classification, prediction and synthesis within the same framework; and (2) an efficient and versatile prior is to consider that the task parameters describing the current situation (body and workspace configuration encountered by the robot) can be represented as affine transformations (including frames of reference, coordinate systems or projections).

By providing such structure to the skill generation problem, the role of the experimenter is to provide the robot with a set of candidate frames (list of coordinate systems) that could potentially be relevant for the task. This paper will show that structuring the affine transformations in such way has a simple interpretation, that it can be easily implemented, and that it remains valid for a wide range of skills that a robot can experience.

This work was in part supported by the DexROV Project through the EC Horizon 2020 programme (Grant #635491).

S. Calinon (✉)
Idiap Research Institute, Martigny, Switzerland
e-mail: sylvain.calinon@idiap.ch

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_7

The task-parameterized Gaussian mixture model (TP-GMM) was presented in [8, 10, 11] for the special case of frames of reference representing rotations and translations in Cartesian space. The current paper discusses the potentials of the approach and introduces several routes for further investigation, aiming at applying the proposed technique to a wider range of affine transformations (directly exploiting the considered application domain), including constraints in both configuration and operational spaces, as well as priority constraints. It also shows that the proposed method can be applied to different probabilistic encoding strategies, including subspace clustering approaches enabling the consideration of high dimension feature spaces. Examples are provided in simulations and on a real robot (transfer of manipulation skills to the Baxter bimanual robot). Accompanying source codes are available at http://www.idiap.ch/software/pbdlib/.

2 Adaptive Models of Movements

Task-parameterized models of movements/behaviors refer to representations that can adapt to a set of task parameters describing for example the current context, situation, state of the environment or state of the robot configuration. The *task parameters* can for example refer to the variables collected by the system to describe the position of objects in the environment. The *task parameters* can be fixed during an execution trial or they can vary while the motion is executed. The *model parameters* refer to the variables learned by the system, namely, that are stored in memory (the internal representation of the movement). During reproduction, a new set of *task parameters* (describing the present situation) is used to generate a new movement (e.g., adaptation to new position of objects).

Several denominations have been introduced in the literature to describe these models, such as *task-parameterized* [11, 40] (the denomination that will be used here), *parametric* [26, 29, 49] or *stylistic* [7]. In these models, the encoding of skills usually serve several purposes, including classification, prediction, synthesis and online adaptation. A taxonomy of task-parameterized models is presented in [8], classifying existing methods in three broad categories: (1) Approaches employing M models for the M demonstrations, performed in M different situations, see e.g. [12, 16, 21, 23, 25, 29, 45]; (2) Approaches employing P models for the P frames of reference that are possibly relevant for the task, see e.g. [13, 32]; (3) Approaches employing a single model whose parameters are modulated by task parameters, see e.g. [20, 26, 49].

In the majority of these approaches, the retrieval of movements from the model parameters and the task parameters is viewed as a standard regression problem. This generality might look appealing at first sight, but it also limits the generalization scope of these models. Our work aims at increasing the generalization capability of task-parameterized models by exploiting the functional nature of the task parameters. The approach arose from the observation that the task parameters in robotics applications can most of the time be related to some form of frames of reference, coordinate systems, basis functions or local projections, whose structure can be exploited to speed up learning and provide the robot with remarkable extrapolation capability.

2.1 Motivation

The core of the approach is to represent an observed movement or behavior as a spring-damper system with varying parameters, where a generative model is used to encode the evolution of the attractor, and the variability and correlation information is used to infer the impedance parameters of the system. These impedance parameters figuratively correspond to the stiffness of a spring and to the damping coefficient of a viscous damper, with the difference that they can also be full stiffness and damping matrices. The model shares links with optimal feedback control strategies in which deviations from an average trajectory are corrected only when they interfere with task performance, resulting in a *minimal intervention principle* [43].

In its task-parameterized version, several frames of reference are interacting with each other to describe tracking behaviors in multiple coordinate systems, where statistical analysis from the perspective of each of these observers is used to estimate feedforward and feedback control terms with linear quadratic optimal control. Figure 1 presents an illustration of the overall approach, which can be decomposed into multiple steps, involving statistical modeling, dynamical systems and optimal control.

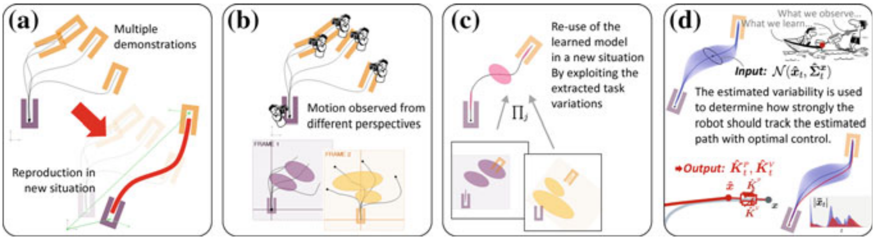

(a)
Multiple demonstrations
Reproduction in new situation
(b)
Motion observed from different perspectives
FRAME 1
FRAME 2
(c)
Re-use of the learned model in a new situation By exploiting the extracted task variations
Π_j
(d)
What we observe...
What we learn...
Input: $\mathcal{N}(\hat{x}_t, \hat{\Sigma}_t^x)$
The estimated variability is used to determine how strongly the robot should track the estimated path with optimal control.
Output: $\hat{K}_t^{\mathcal{P}}, \hat{K}_t^{\mathcal{V}}$
$\hat{x}$
$\hat{K}^{\mathcal{P}}$
$\hat{K}^{\mathcal{V}}$
x
$|\hat{\Sigma}_t|$
t

Fig. 1 Illustration of the overall approach (see main text for details). **a** Observation of a task in different situations and generalization to new contexts. Multiple demonstrations provide the opportunity to discern the structure of the task. **b** Probabilistic encoding of continuous movements in multiple coordinate systems. **c** Exploitation of variability and correlation information to adapt the motion to new situations. With cross-situational observations of the same task, the robot is able to generalize the skill to new situations. **d** Computation of the underlying optimal control strategy driving the observed behavior

2.2 Example with a Single Gaussian

Before presenting the details of the task-parameterized model, the approach is motivated by an introductory example with a single Gaussian.

Two frames will be considered, described respectively at each time step t by $\{\boldsymbol{b}_{t,1}, \boldsymbol{A}_{t,1}\}$ and $\{\boldsymbol{b}_{t,2}, \boldsymbol{A}_{t,2}\}$, representing the origin of the observer $\boldsymbol{b}_{t,j}$ and a set of basis vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, ...\}$ forming a transformation matrix $\boldsymbol{A}_{t,j} = [\boldsymbol{e}_{1,t,j},\ \boldsymbol{e}_{2,t,j},\ ...]$.

A set of demonstrations is observed from the perspective of the two frames. During reproduction, each frame expects the new datapoints to lie within the same range as that of the demonstrations. If $\mathcal{N}\big(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}\big)$ and $\mathcal{N}\big(\boldsymbol{\mu}^{(2)}, \boldsymbol{\Sigma}^{(2)}\big)$ are the normal distributions of the observed demonstrations in the first and second frames, the two frames respectively expect the reproduction attempt to lie at the intersection of the distributions $\mathcal{N}\big(\hat{\boldsymbol{\xi}}_t^{(1)}, \hat{\boldsymbol{\Sigma}}_t^{(1)}\big)$ and $\mathcal{N}\big(\hat{\boldsymbol{\xi}}_t^{(2)}, \hat{\boldsymbol{\Sigma}}_t^{(2)}\big)$. These distributions can be computed with the linear transformation property of normal distribution as

$$\hat{\boldsymbol{\xi}}_t^{(1)} = \boldsymbol{A}_{t,1}\ \boldsymbol{\mu}^{(1)} + \boldsymbol{b}_{t,1}\ , \qquad \hat{\boldsymbol{\Sigma}}_t^{(1)} = \boldsymbol{A}_{t,1}\ \boldsymbol{\Sigma}^{(1)} \boldsymbol{A}_{t,1}^{\top}\ , \tag{1}$$

$$\hat{\boldsymbol{\xi}}_t^{(2)} = \boldsymbol{A}_{t,2}\ \boldsymbol{\mu}^{(2)} + \boldsymbol{b}_{t,2}\ , \qquad \hat{\boldsymbol{\Sigma}}_t^{(2)} = \boldsymbol{A}_{t,2}\ \boldsymbol{\Sigma}^{(2)} \boldsymbol{A}_{t,2}^{\top}\ . \tag{2}$$

A trade-off thus needs to be determined during reproduction to concord with the distributions expected by each frame. The objective function can be defined as the weighted sum of quadratic error terms

$$\hat{\boldsymbol{\xi}}_t \ = \ \arg\min_{\boldsymbol{\xi}_t} \sum_{j=1}^{2} \big(\boldsymbol{\xi}_t - \hat{\boldsymbol{\xi}}_t^{(j)}\big)^{\top}\ \hat{\boldsymbol{\Sigma}}_t^{(j)^{-1}} \big(\boldsymbol{\xi}_t - \hat{\boldsymbol{\xi}}_t^{(j)}\big). \tag{3}$$

The above objective can easily be solved by differentiation, providing a point $\hat{\boldsymbol{\xi}}_t$, with an error defined by covariance $\hat{\boldsymbol{\Sigma}}_t$. This estimate corresponds to a product of Gaussians (intersection between the two Gaussians). Figure 2 illustrates this process for one of the Gaussians of Fig. 1.

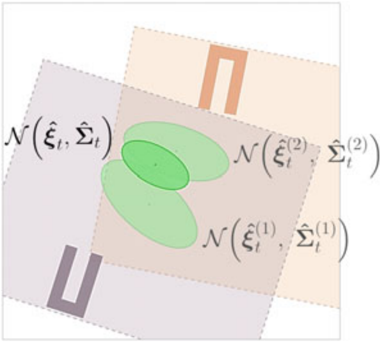

$\mathcal{N}(\hat{\xi}_t, \hat{\Sigma}_t)$
$\mathcal{N}(\hat{\xi}_t^{(2)}, \hat{\Sigma}_t^{(2)})$
$\mathcal{N}(\hat{\xi}_t^{(1)}, \hat{\Sigma}_t^{(1)})$

Fig. 2 Minimization of the objective function in Eq. (3) composed of a weighted sum of quadratic error terms, whose result corresponds to a product of Gaussians. It is easy to show that $\mathcal{N}(\hat{\boldsymbol{\xi}}_t, \hat{\boldsymbol{\Sigma}}_t)$ corresponds to the Gaussian outcoming from the product of the two Gaussians $\mathcal{N}(\hat{\boldsymbol{\xi}}_t^{(1)}, \hat{\boldsymbol{\Sigma}}_t^{(1)})$ and $\mathcal{N}(\hat{\boldsymbol{\xi}}_t^{(2)}, \hat{\boldsymbol{\Sigma}}_t^{(2)})$

3 Task-Parameterized Gaussian Mixture Model (TP-GMM)

TP-GMM is a direct extension of the objective problem presented above, by considering multiple frames and multiple clusters of datapoints (soft clustering via mixture modeling). It probabilistically encodes the relevance of candidate frames, which can change throughout the task. In contrast to approaches such as [33] that aim at extracting a single (most prominent) coordinate system located at the end of a motion segment, the proposed approach allows the superposition and transition of different coordinate systems that are relevant for the task (parallel organization of behavior primitives, adaptation to multiple viapoints in the middle of the movement, modulation based on positions, orientations or geometries of objects, etc.).

Each demonstration $m \in \{1, \ldots, M\}$ contains T_m datapoints forming a dataset of N datapoints $\{\boldsymbol{\xi}_t\}_{t=1}^N$ with $N = \sum_m^M T_m$. The task parameters are represented by P coordinate systems, defined at time step t by $\{\boldsymbol{b}_{t,j}, \boldsymbol{A}_{t,j}\}_{j=1}^P$, representing respectively the origin and the basis of the coordinate system.

The demonstrations $\boldsymbol{\xi} \in \mathbb{R}^{D \times N}$ are observed from these different viewpoints, forming P trajectory samples $\boldsymbol{X}^{(j)} \in \mathbb{R}^{D \times N}$. These samples can be collected from sensors located at the frames, or computed with

$$\boldsymbol{X}_t^{(j)} = \boldsymbol{A}_{t,j}^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_{t,j}). \tag{4}$$

The parameters of the proposed *task-parameterized GMM* (TP-GMM) with K components are defined by $\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^P\}_{i=1}^K$ (π_i are the mixing coefficients, $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Sigma}_i^{(j)}$ are the center and covariance matrix of the i-th Gaussian component in frame j).

Learning of the parameters is achieved by log-likelihood maximization subject to the constraint that the data in the different frames arose from the same source, resulting in an EM process iteratively updating the model parameters until convergence, see [10] for details. Model selection (i.e., determining the number of Gaussians in the GMM) is compatible with techniques employed in standard mixture models (*Bayesian information criterion* [37], *Dirichlet process* [34], small-variance asymptotics [27], etc.). For a movement in Cartesian space with 10 demonstrations and 3 candidate frames, the overall learning process typically takes 1–3 s. The reproduction is much faster and can be computed online (typically below 1 ms).

The learned model is then used to reproduce movements in other situations (for new position and orientation of candidate frames). A new GMM with parameters $\{\pi_i, \hat{\boldsymbol{\xi}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\}_{i=1}^K$ can thus automatically be generated with

$$\mathcal{N}\Big(\hat{\boldsymbol{\xi}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\Big) \propto \prod_{j=1}^P \mathcal{N}\Big(\hat{\boldsymbol{\xi}}_{t,i}^{(j)}, \; \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)}\Big),$$
$$\text{with} \quad \hat{\boldsymbol{\xi}}_{t,i}^{(j)} = \boldsymbol{A}_{t,j} \boldsymbol{\mu}_i^{(j)} + \boldsymbol{b}_{t,j} \;, \quad \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)} = \boldsymbol{A}_{t,j} \boldsymbol{\Sigma}_i^{(j)} \boldsymbol{A}_{t,j}^{\top}, \tag{5}$$

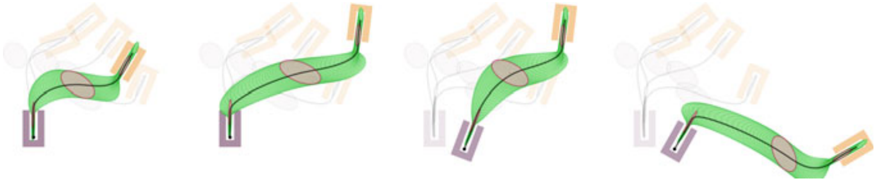


Fig. 3 Generalization capability of task-parameterized Gaussian mixture model. Each graph shows a different situation with increasing generalization complexity. In each graph, the four demonstrations and the associated adapted model parameters are depicted in semi-transparent colors

where the result of the Gaussian product is given by

$$\hat{\boldsymbol{\Sigma}}_{t,i} = \Big(\sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)^{-1}} \Big)^{-1}, \quad \hat{\boldsymbol{\xi}}_{t,i} = \hat{\boldsymbol{\Sigma}}_{t,i} \sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)^{-1}} \hat{\boldsymbol{\xi}}_{t,i}^{(j)}. \tag{6}$$

For computational efficiency, the above equations can be computed with precision matrices instead of covariances.

Several approaches can be used to retrieve movements from the proposed model. An option is to encode both static and dynamic features in the mixture model to retrieve continuous behaviors [22, 39, 51]. An alternative option is to encode time as additional feature in the GMM, and use *Gaussian mixture regression* (GMR) [18] to retrieve continuous behaviors. Similarly, if the evolution of a decay term is encoded instead of time, the system yields a probabilistic formulation of dynamical movement primitives (DMP) [20], see [11] for details. Figure 3 presents TP-GMR reproduction results for the example in Fig. 1.

4 Extension to Task-Parameterized Subspace Clustering

Classical model-based clustering will tend to perform poorly in high-dimensional spaces. A simple way of handling this issue is to reduce the number of parameters by considering diagonal covariances instead of full matrices, which corresponds to a separated treatment of each variable. Although common in robotics, such decoupling can be a limiting factor to encode movements and sensorimotor streams, because it follows a strategy that is not fully exploiting principles underlying coordination, motor skill acquisition and action-perception couplings.

The rationale is that diagonal structures are unadapted to motor skill representation because they do not encapsulate coordination information among the control variables. The good news is that a wide range of mixture modeling techniques exist between the encoding of diagonal and full covariances. At the exception of [14, 47], these techniques have only been exploited to a limited extent in robot skills acquisition. They can be studied as a subspace clustering problem, aiming to group

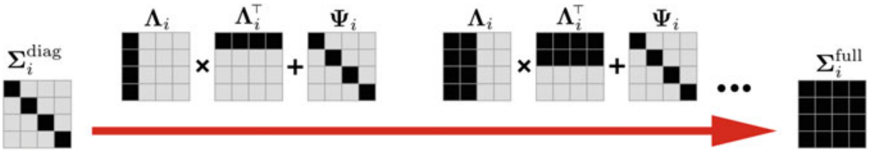


Fig. 4 The mixture of factor analyzers (MFA) covers a wide range of covariance structures for the modeling of the data, from diagonal covariances (*left*) to full covariances (*right*)

datapoints such that they can be locally projected in subspaces of reduced dimensionality. Such subspace clustering helps the analysis of the local trend of the movement, while reducing the number of parameters to be estimated, and “locking” the most important coordination patterns to efficiently cope with perturbations.

Several possible constraints can be considered, grouped in families such as parsimonious GMM [6], *mixtures of factor analyzers* (MFA) [30] or *mixtures of probabilistic principal component analyzers* [42]. Methods such as MFA provide a simple approach to the problem of high-dimensional cluster analysis with a slight modification of the generative model underlying the mixture of Gaussians to enforce low-dimensional models (i.e., noninvasive regarding the other methods used in the proposed framework). The basic idea of factor analysis (FA) is to reduce the dimensionality of the data while keeping the observed covariance structure. MFA assumes for each component i a covariance structure of the form $\boldsymbol{\Sigma}_i = \boldsymbol{\Lambda}_i \boldsymbol{\Lambda}_i^\top + \boldsymbol{\Psi}_i$, where $\boldsymbol{\Lambda}_i \in \mathbb{R}^{D \times d}$, known as the *factor loadings matrix*, typically has $d < D$ (providing a parsimonious representation of the data), and a diagonal noise matrix $\boldsymbol{\Psi}_i$.

Figure 4 shows that the covariance structure in MFA can span a wide range of covariances.

The TP-GMM presented in Sect. 3 is fully compatible with the subspace clustering approaches mentioned above. Bayesian nonparametric approaches such as [48] can be used to simultaneously select the number of clusters and the dimension of the subspace in each cluster.

The TP-MFA extension of TP-GMM opens several roads for further investigation. A possible extension is to use tied structures in the covariances to enable the organization and reuse of previously acquired synergies [17]. Another possible extension is to enable deep or hierarchical learning techniques in task-parameterized models. As discussed in [41], the prior of each FA can be replaced by a separate second-level MFA that learns to model the aggregated posterior of that FA (instead of the isotropic Gaussian), providing a hierarchical structure organization where one layer of latent variables can be learned at a time.

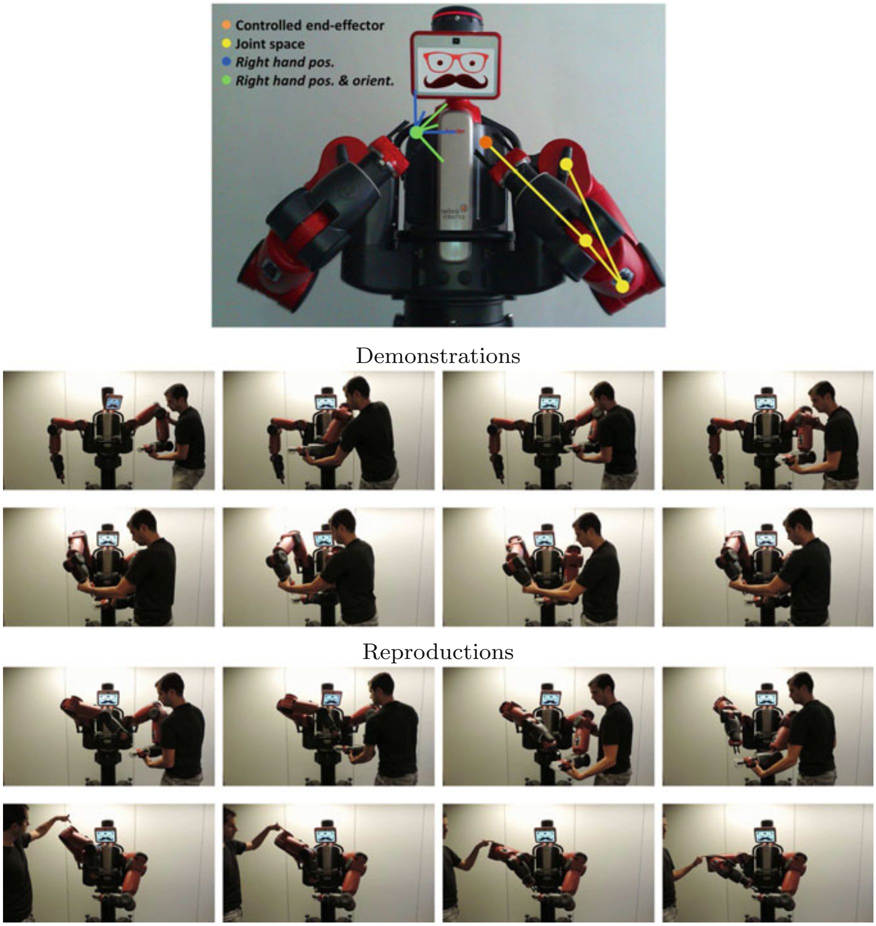


Fig. 5 Learning of two behaviors with the Baxter robot. The taught tasks consist of holding a cup horizontally with one hand, and holding a sugar cube above the cup with the other hand, where the two task primitives can be combined in parallel. The demonstrations are provided in two steps by kinesthetic teaching, namely, by holding the arms of the robot and moving them during the task while the robot compensates for the effect of gravity. This procedure allows the user to move the robot arms without feeling their weight and without feeling the motors in the articulations, while the sensors are used to record position information. Here, the data are recorded in several frames of reference (*top image*). During reproduction, the robot is controlled by following a minimal intervention principle, where the computed feedforward and feedback control commands result in different levels of stiffness obeying the extracted variation and coordination constraints of the task. *First sequence*: Brief demonstration to show the robot how to hold a cup horizontally. *Second sequence*: Brief demonstration to show how to hold a sugar cube above the cup. *Third sequence*: Manual displacement of the left arm to test the learned behavior (the coordination of the two hands was successfully learned). *Last sequence*: Combination of the two learned task primitives. Here, the user pushes the robot to show that the robot remains soft for perturbations that do not conflict with the acquired task constraints (automatic exploitation of the redundant degrees of freedom that do not conflict with the task)

5 Extension to Minimal Intervention Control

We showed in [10] that TP-GMM can be used to autonomously regulate the stiffness and damping behavior of the robot, see also Fig. 1d. It shares similarities with the solution proposed by Medina et al. in the context of risk-sensitive control for haptic assistance [31], by exploiting the predicted variability to form a minimal intervention controller (in task space or in joint space). The retrieved variability and correlation information is exploited to generate safe and natural movements within an optimal control strategy, in accordance to the predicted range of motion to reproduce the task, evaluated for the current situation. TP-GMM is fully compatible with *linear quadratic regulation* (LQR) and *model predictive control* (MPC) [4], providing an approach to learn controllers adapted to the current situation, with feedforward and feedback control commands varying in regard to external task parameters, see [10] for details.

Figure 5 demonstrates that a TP-GMM with a single Gaussian, combined with an infinite-horizon LQR, can readily be used to represent various behaviors that directly exploit the torque control capability of the robot and the redundancy, both at the level of the task and at the level of the robot kinematic structure.

It is worth noting that each frame in the TP-GMM has an associated sub-objective function as in Eq. (3), which aims at minimizing the discrepancy between the demonstrations and the reproduction attempt. By considering the combination of these sub-objectives in the overall objective, the problem can be viewed as a rudimentary form of *inverse optimal control* (IOC) [1]. This form of IOC does not have external constraints and can be solved analytically, which means that it can provide a controller without exploratory search, at the expense of being restricted to simple forms of objectives (weighted sums of quadratic errors whose weights are learned from the demonstrations). This dual view can be exploited for further research in learning from demonstration, either to bridge action-level and goal-driven imitation, or to initialize the search in IOC.

6 Extension to Multimodal Data and Projection Constraints

TP-GMM is not limited to coordinate systems representing objects in Cartesian space. It can be extended to other forms of locally linear transformations or projections, which opens many roads for further research.

The consideration of non-square $\boldsymbol{A}_{t,j}$ matrices is for example relevant to learn and reproduce soft constraints in both configuration and operational spaces (through Jacobian operators). With a preliminary model of task-parameterized movements, we explored in [9] how a similar approach could be used to simultaneously learn constraints in joint space and task space. The model also provides a principled way to learn priority constraints in a probabilistic form (through nullspace operators). The different frames correspond in this case to several subspace projections of the same movement, whose relevance is estimated statistically from the demonstrations.

A wide range of motor skills could potentially be adapted to this framework, by exploiting the functional nature of task parameters to build models that learn the local structure of the task from a small number of demonstrations. Indeed, most task parameterization in robot control can be related to some form of frames of reference, coordinate systems or basis functions, where the involvement of the frames can change during the execution of the task, with transformations represented as local linear projection operators (Jacobians for inverse kinematics, kernel matrices for nullspace projections, etc.).

The potential applications are diverse, with an objective that is well in line with the original purpose of motor primitives to be composed together serially or in parallel [15]. Further work is required to investigate in which manner TP-GMM could be exploited to provide a probabilistic view of robotics techniques that are in practice predefined, handled by *ad hoc* solutions, or sometimes inefficiently set as hard constraints. This includes the consideration of soft constraints in both configuration and operational spaces. A wide range of robot skills can be defined in such way, see e.g. the possible tasks described in Sect. 6.2.1 of [3]. In humanoids, the candidate frames could for example be employed to learn the constraints of whole-body movements from demonstration or experience, based on the regularities extracted from different subspace projections.

An important category of applications currently attracting a lot of attention concerns the problems requiring priority constraints [19, 28, 36, 44, 50]. With an appropriate definition of the frames and with an initial set of candidate task hierarchies, such constraints can be learned and encoded within a TP-GMM. Here, the probabilistic encoding is exploited to discover, from statistical analysis of the demonstrations, in which manner each subtask is prioritized.

For a controller handling constraints both in configuration and operational spaces, the most common candidate projection operators can be defined as

$$\begin{aligned}
\hat{\boldsymbol{q}}_{t,i}^{(j)} &= \boldsymbol{I} & \boldsymbol{\mu}_i^{(j)} &+ \boldsymbol{0} && (7)\\
\hat{\boldsymbol{q}}_{t,i}^{(j)} &= \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1}) & \boldsymbol{\mu}_i^{(j)} &+ \boldsymbol{q}_{t-1} - \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{x}_{t-1} && (8)\\
\hat{\boldsymbol{q}}_{t,i}^{(j)} &= \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{A}_t^{\mathcal{O}} & \boldsymbol{\mu}_i^{(j)} &+ \boldsymbol{q}_{t-1} + \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\big[\boldsymbol{b}_t^{\mathcal{O}} - \boldsymbol{x}_{t-1}\big] && (9)\\
\hat{\boldsymbol{q}}_{t,i}^{(j)} &= \boldsymbol{N}(\boldsymbol{q}_{t-1}) & \boldsymbol{\mu}_i^{(j)} &+ \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{J}(\boldsymbol{q}_{t-1})\boldsymbol{q}_{t-1} && (10)\\
\hat{\boldsymbol{q}}_{t,i}^{(j)} &= \boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1}) & \boldsymbol{\mu}_i^{(j)} &+ \boldsymbol{q}_{t-1} - \boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1})\,\boldsymbol{x}_{t-1} && (11)\\
\hat{\boldsymbol{q}}_{t,i}^{(j)} &= \underbrace{\boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{A}_t^{\mathcal{O}}}_{\boldsymbol{A}_{t,j}} & \boldsymbol{\mu}_i^{(j)} &+ \underbrace{\boldsymbol{q}_{t-1} + \boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1})\big[\boldsymbol{b}_t^{\mathcal{O}} - \boldsymbol{x}_{t-1}\big]}_{\boldsymbol{b}_{t,j}}, && (12)
\end{aligned}$$

covering a wide range of robotics applications.

Note here that the product of Gaussians is computed in configuration space ($\boldsymbol{q}$ and $\boldsymbol{x}$ represent respectively poses in joint space and task space). Equation (7) describes

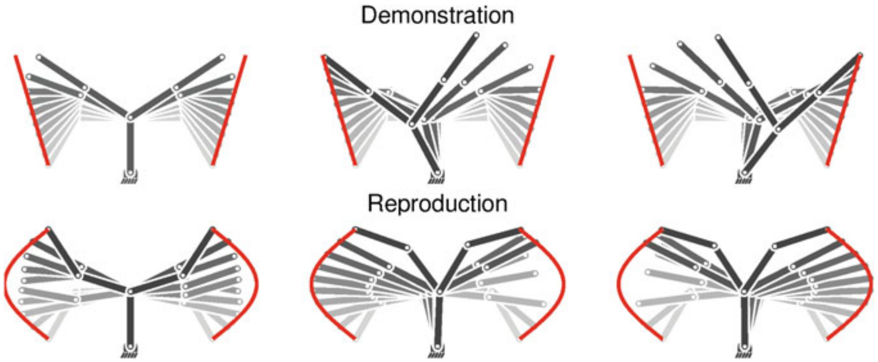


Fig. 6 Illustration of the encoding of priority constraints in a TP-GMM. The top row shows 3 demonstrations with a bimanual planar robot with 5 articulations. The color of the robot changes from *light gray* to *black* with the movement. The task consists of tracking two objects with the *left* and *right* hands (the path of the objects are depicted in *red*). In some parts of the demonstrations, the two objects could not be reached, and the demonstrator either made a compromise (*left* graph), or gave priority to the *left* or *right* hand (*middle* and *right* graphs). The *bottom* row shows reproduction attempts for new trajectories of the two objects. Although faced with different situations, the priority constraints are reproduced in a similar fashion as in the corresponding demonstrations

joint space constraints in a fixed frame. It corresponds to the canonical frame defined by $\boldsymbol{A}_{t,j} = \boldsymbol{I}$ (identity matrix) and $\boldsymbol{b}_{t,j} = \boldsymbol{0}$. Equation (8) describes absolute position constraints (in operational space), where $\boldsymbol{J}^{\dagger}$ is the Jacobian pseudoinverse used as least-norm inverse kinematics solution. Note that Eq. (8) describes a moving frame, where the task parameters change at each iteration (observation of a changing pose in configuration space). Equation (9) describes relative position constraints, where the constraint in task space is related to an object described at each time step t by a position $\boldsymbol{b}_t^{\mathcal{O}}$ and an orientation matrix $\boldsymbol{A}_t^{\mathcal{O}}$ in task space. Equation (10) describes nullspace/priority constraints in joint space, with $\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ a nullspace projection operator. Equation (11) describes absolute position nullspace/priority constraints, where the secondary objective is described in task space (for a point in the kinematic chain with corresponding Jacobian $\tilde{\boldsymbol{J}}$). Finally, Eq. (12) describes relative position nullspace/priority constraints.

The above equations can be retrieved without much effort by discretizing (with an Euler approximation) the standard inverse kinematics and nullspace control relations that can be found in most robotics textbooks, see e.g. [3].

Figure 6 presents a TP-GMM example with task parameters taking the form of nullspace bases. The frames are defined by Eqs. (9) and (12) with two different combinations of nullspaces $\boldsymbol{N}$ and Jacobians $\tilde{\boldsymbol{J}}$ corresponding to the left and right arm.

7 Discussion and Further Work

A potential limitation of the current TP-GMM approach is that it requires the experimenter to provide an initial set of frames that will act as candidate projections/transformations of the data that can potentially be relevant for the task. The number of frames can be overspecified by the experimenter (e.g., by providing an exhaustive list), at the expense of potentially requiring a large number of demonstrations to obtain sufficient statistics to discard the frames that have no role in the task. The demonstrations must also be sufficiently varied, which becomes more difficult as the number of candidate frames increases. The problem *per se* is not different from the problem of selecting the variables that will form the feature vector fed to a learning system. The only difference here is that the initial selection of frames takes the form of affine transformations instead of the initial selection of elements in a feature vector.

In practice, the experimenter selects the list of objects or landmarks in the robot workspace, as well as the locations in the robot kinematic chain that might be relevant for the task, which are typically the end-effectors of the robot, where tools, grippers or parts in contact with the environment are mounted. It should be noted here that if some frames of reference are missing during reproduction (e.g., when occlusions occur or when frames are collected at different rates), the system is still able to reproduce an appropriate behavior given the circumstance, see [2] for details.

The issue of predefining an initial set of frames of reference is not restrictive when the number of frames remains reasonably low (e.g., when they come from a set of predefined objects tracked with visual markers in a lab setting). However, for perception in unconstrained environment, the number of frames could potentially grow (e.g., detection of phantom objects), while the number of demonstrations should remain low.

Further work is thus required to detect redundant frames or remove irrelevant frames, as well as to automatically determine in which manner the frames are coordinated with each other and locally contribute to the achievement of the task. A promising route for further investigation is to exploit the recent developments in multilinear algebra and tensor methods [24, 38] that exploit the multivariate structure of data for statistical analysis and compression without transforming it to a matrix form.

In the proposed task-parameterized framework, the movement is expressed simultaneously in multiple coordinate systems, and is stored as a multidimensional array (tensor-variate data). This opens many roads for further investigation, where multilinear algebra could provide a principled method to simultaneously extract *eigenframes*, *eigenposes* and *eigentrajectories*. Multiway analysis of tensor-variate data could imaginably offer a rich set of data decomposition techniques, which has been demonstrated in computer imaging fields such as face processing [46], video analysis [52], geoscience [35] or neuroimaging [5], but which remains underexploited in robotics and motor skills acquisition.

There are several other encoding methods that can be explored within the proposed task-parameterized approach (e.g., with *hidden Markov models* (HMM), with *Gaussian processes* (GP) or with other forms of trajectory distributions). Indeed, it is worth noting that the approach is not restricted to mixture models and can be employed with other representations as long as a local measure of uncertainty is available.

8 Conclusion

An efficient prior assumption in robot learning from demonstration is to consider that skills are modulated by external task parameters. These task parameters often take the form of affine transformations, whose role is to describe the current situation encountered by the robot (body and workspace configuration). We showed that this structure can be used with different statistical modeling strategies, including standard mixture models and subspace clustering. The approach can be used in a wide variety of problems in robotics, by reinterpreting them with a structural relation between the task parameters and the model parameters represented as candidate frames of reference. The rationale is that robot skills can often be related to coordinate systems, basis functions or local projections, whose structure can be exploited to speed up learning and provide robots with better generalization capability. Early promises of the approach were discussed in a series of problems in configuration and operational spaces, including tests on a Baxter robot.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the International Conference on Machine Learning (ICML), Banff, Alberta, Canada (2004)
2. Alizadeh, T., Calinon, S., Caldwell, D.G.: Learning from demonstrations with partially observable task parameters. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3309–3314, Hong Kong, China (2014)
3. Antonelli, G.: Underwater Robots, 3rd edn. Springer International Publishing, Heidelberg (2014)
4. Astrom, K.J., Murray, R.M.: Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press, Princeton (2008)
5. Basser, P.J., Pajevic, S.: A normal distribution for tensor-valued random variables: applications to diffusion tensor MRI. IEEE Trans. Med. Imag. **22**(7), 785–794 (2003)
6. Bouveyron, C., Brunet, C.: Model-based clustering of high-dimensional data: a review. Comput. Stat. Data Anal. **71**, 52–78 (2014)
7. Brand, M., Hertzmann, A.: Style machines. In: Proceedings of the ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 183–192, New Orleans, Louisiana, USA (2000)

8. Calinon, S., Alizadeh, T., Caldwell, D.G.: On improving the extrapolation capability of task-parameterized movement models. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 610–616, Tokyo, Japan (2013)
9. Calinon, S., Billard, A.G.: Statistical learning by imitation of competing constraints in joint space and task space. Adv. Robot. **23**(15), 2059–2076 (2009)
10. Calinon, S., Bruno, D., Caldwell, D.G.: A task-parameterized probabilistic model with minimal intervention control. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3339–3344, Hong Kong, China (2014)
11. Calinon, S., Li, Z., Alizadeh, T., Tsagarakis, N.G., Caldwell, D.G.: Statistical dynamical systems for skills acquisition in humanoids. In: Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids), pp. 323–329, Osaka, Japan (2012)
12. Campbell, C.L., Peters, R.A., Bodenheimer, R.E., Bluethmann, W.J., Huber, E., Ambrose, R.O.: Superpositioning of behaviors learned through teleoperation. IEEE Trans. Robot. **22**(1), 79–91 (2006)
13. Dong, S., Williams, B.: Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. Int. J. Soc. Robot. **4**(4), 357–368 (2012)
14. Field, M., Stirling, D., Pan, Z., Naghdy, F.: Learning trajectories for robot programing by demonstration using a coordinated mixture of factor analyzers. IEEE Trans. Cybern. **46**(3), 706–717 (2016)
15. Flash, T., Hochner, B.: Motor primitives in vertebrates and invertebrates. Cur. Opin. Neurobiol. **15**(6), 660–666 (2005)
16. Forte, D., Gams, A., Morimoto, J., Ude, A.: On-line motion synthesis and adaptation using a trajectory database. Robot. Auton. Syst. **60**(10), 1327–1339 (2012)
17. Gales, M.J.F.: Semi-tied covariance matrices for hidden markov models. IEEE Trans. Speech Audio Process. **7**(3), 272–281 (1999)
18. Ghahramani, Z., Jordan, M.I.: Supervised learning from incomplete data via an EM approach. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems (NIPS), vol. 6, pp. 120–127. Morgan Kaufmann Publishers Inc, San Francisco (1994)
19. Hak, S., Mansard, N., Stasse, O., Laumond, J.P.: Reverse control for humanoid robot task recognition. IEEE Trans. Syst. Man Cybern. Part B Cybern. **42**(6), 1524–1537 (2012)
20. Ijspeert, A., Nakanishi, J., Pastor, P., Hoffmann, H., Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural Comput. **25**(2), 328–373 (2013)
21. Inamura, T., Toshima, I., Tanie, H., Nakamura, Y.: Embodied symbol emergence based on mimesis theory. Intl. J. Robot. Res. **23**(4–5), 363–377 (2004)
22. Khansari-Zadeh, S.M., Billard, A.: Learning stable non-linear dynamical systems with Gaussian mixture models. IEEE Trans. Robot. **27**(5), 943–957 (2011)
23. Kober, J., Wilhelm, A., Oztop, E., Peters, J.: Reinforcement learning to adjust parametrized motor primitives to new situations. Auton. Robot. **33**(4), 361–379 (2012)
24. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
25. Kronander, K., Khansari-Zadeh, M.S.M., Billard, A.: Learning to control planar hitting motions in a minigolf-like task. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 710–717 (2011)
26. Krueger, V., Herzog, D.L., Baby, S., Ude, A., Kragic, D.: Learning actions from observations: primitive-based modeling and grammar. IEEE Robot. Autom. Mag. **17**(2), 30–43 (2010)
27. Kulis, B., Jordan, M.I.: Revisiting k-means: new algorithms via Bayesian nonparametrics. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 1–8, Edinburgh, Scotland, UK (2012)
28. Lober, R., Padois, V., Sigaud, O.: Multiple task optimization using dynamical movement primitives for whole-body reactive control. In: Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids), Madrid, Spain (2014)
29. Matsubara, T., Hyon, S.H., Morimoto, J.: Learning parametric dynamic movement primitives from multiple demonstrations. Neural Netw. **24**(5), 493–500 (2011)

30. McLachlan, G.J., Peel, D., Bean, R.W.: Modelling high-dimensional data by mixtures of factor analyzers. Comput. Stat. Data Anal. **41**(3–4), 379–388 (2003)
31. Medina, J.R., Lee, D., Hirche, S.: Risk-sensitive optimal feedback control for haptic assistance. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1025–1031 (2012)
32. Mühlig, M., Gienger, M., Steil, J.: Interactive imitation learning of object movement skills. Auton. Robots **32**(2), 97–114 (2012)
33. Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., Barto, A.G.: Learning grounded finite-state representations from unstructured demonstrations. Int. J. Robot. Res. **34**(2), 131–157 (2015)
34. Rasmussen, C.E.: The infinite Gaussian mixture model. In: Solla, S.A., Leen, T.K., Mueller, K.R. (eds.) Advances in Neural Information Processing Systems (NIPS), pp. 554–560. MIT Press, Cambridge (2000)
35. Renard, N., Bourennane, S., Blanc-Talon, J.: Denoising and dimensionality reduction using multilinear tools for hyperspectral images. IEEE Geosci. Remote Sens. Lett. **5**(2), 138–142 (2008)
36. Saveriano, M., An, S., Lee, D.: Incremental kinesthetic teaching of end-effector and null-space motion primitives. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3570–3575 (2015)
37. Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
38. Signoretto, M., Van de Plas, R., De Moor, B., Suykens, J.A.K.: Tensor versus matrix completion: a comparison with application to spectral data. IEEE Signal Process. Lett. **18**(7), 403–406 (2011)
39. Sugiura, K., Iwahashi, N., Kashioka, H., Nakamura, S.: Learning, generation, and recognition of motions by reference-point-dependent probabilistic models. Adv. Robot. **25**(6–7), 825–848 (2011)
40. Tang, J., Singh, A., Goehausen, N., Abbeel, P.: Parameterized maneuver learning for autonomous helicopter flight. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1142–1148 (2010)
41. Tang, Y., Salakhutdinov, R., Hinton, G.: Deep mixtures of factor analysers. In: Proceedings of the International Conference on Machine Learning (ICML), Edinburgh, Scotland (2012)
42. Tipping, M.E., Bishop, C.M.: Mixtures of probabilistic principal component analyzers. Neural Comput. **11**(2), 443–482 (1999)
43. Todorov, E., Jordan, M.I.: Optimal feedback control as a theory of motor coordination. Nat. Neurosci. **5**, 1226–1235 (2002)
44. Towell, C., Howard, M., Vijayakumar, S.: Learning nullspace policies. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 241–248 (2010)
45. Ude, A., Gams, A., Asfour, T., Morimoto, J.: Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans. Robot. **26**(5), 800–815 (2010)
46. Vasilescu, M.A.O., Terzopoulos, D.: Multilinear analysis of image ensembles: tensorFaces. Computer Vision (ECCV). Lecture Notes in Computer Science, vol. 2350, pp. 447–460. Springer, Heidelberg (2002)
47. Vijayakumar, S., D'souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Comput. **17**(12), 2602–2634 (2005)
48. Wang, Y., Zhu, J.: DP-space: Bayesian nonparametric subspace clustering with small-variance asymptotics. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 1–9, Lille, France (2015)
49. Wilson, A.D., Bobick, A.F.: Parametric hidden Markov models for gesture recognition. IEEE Trans. Pattern Analysis Mach. Intell. **21**(9), 884–900 (1999)
50. Wrede, S., Emmerich, C., Ricarda, R., Nordmann, A., Swadzba, A., Steil, J.J.: A user study on kinesthetic teaching of redundant robots in task and configuration space. J. Hum.-Robot Interact. **2**, 56–81 (2013)

51. Zen, H., Tokuda, K., Kitamura, T.: Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. Comput. Speech Lang. **21**(1), 153–173 (2007)
52. Zhao, Q., Zhou, G., Adali, T., Zhang, L., Cichocki, A.: Kernelization of tensor-based models for multiway data analysis: processing of multidimensional structured data. IEEE Signal Process. Mag. **30**(4), 137–148 (2013)

Modeling Objects as Aspect Transition Graphs to Support Manipulation

Li Yang Ku, Erik Learned-Miller and Roderic Grupen

1 Introduction

In the fields of human psychophysics and neurophysiology, the study of visual object recognition is often motivated by the question of how humans recognize 3-D objects while receiving only 2-D light patterns on the retina [29]. Two types of models for object recognition have been proposed to answer this question. The structural description model represents each object by a small number of view-invariant primitives and their position in an object-centered reference frame [23]. Alternatively, image-based models represent each object as a collection of viewpoint-specific local features. Since the development of these models, experiments in human psychophysics and neurophysiology have provided converging evidence for image-based models. In experiments done by Bülthoff and Edelman [2, 6], it was shown that when a new object is presented to a human subject, a small set of canonical views are formed despite the fact that each viewpoint is presented to the subject for the same amount of time. Experiments on monkeys further confirmed that a significant percentage of neurons in the inferior temporal cortex responded selectively to a subset of views of a known object [20]. However, how an infinite set of possible views can be effectively reduced to a smaller set of canonical views remains an open question. Different approaches such as view interpolation [24] and linear combinations of views [31] have been proposed.

L.Y. Ku (✉) · E. Learned-Miller · R. Grupen
College of Information and Computer Science, University of Massachusetts Amherst,
Amherst, MA, USA
e-mail: lku@cs.umass.edu

E. Learned-Miller
e-mail: elm@cs.umass.edu

R. Grupen
e-mail: grupen@cs.umass.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_8

Closely related to the image-based models in the field of psychophysics, aspect graphs were first introduced as a way to represent 3-D objects using multiple 2-D views in the field of computer vision [16]. An aspect graph contains distinctive views of an object captured from a viewing sphere centered on the object. Research on aspect graphs has focused on the methodologies for automatically computing aspect graphs of polyhedra [10] and general curved objects [17]. The set of viewpoints on the viewing sphere is partitioned into regions that have the same qualitative topological structure as an image of the geometric contours of the object. However, work done in this field was mostly theoretical and was not applicable in practice [7]. One of the difficulties faced in this work concerned the large number of aspects that exist for normal everyday objects. An object can generate millions of different aspects, but many of these may be irrelevant at the scale of the observation. In this work, we propose an object model that provides a consistent treatment for classifying observations into aspects within a practically-sized subset of all possible aspects for most types of objects including deformable objects.

Object and tool manipulation are essential skills for a humanoid robot, and recognizing known objects and tools is often a first step in manipulation tasks. In computer vision and robotics, object recognition is often defined as the process of labeling segments in an image or fitting a 3-D model to an observed point cloud. The object models used to accomplish these tasks usually include information about visual appearance and shape. However, what these object recognition systems provide is merely a label for each observed object. The sequence of actions that the robot should perform based on the object label are often manually defined. Without linking actions to object labels these object models themselves have limited utility to the robot.

Both aspect graphs and image-based models attempt to model 3-D objects with multiple 2-D views. Research in aspect graphs has encountered difficulties in determining the threshold to differentiate two distinctive views while for image-based models how to generalize from unfamiliar to canonical views remains an open question. In this article we propose an object model that addresses both of these issues and incorporates actions in a coherent way. In particular, we show how aspects can be chosen in a unique and repeatable way that is defined by the object itself, and in a way that supports manipulation.

While many of our examples use images and visual processing, our methodology applies to other modes of perception such as audition and haptics. Below, we use the terms “observation” and “aspect” instead of “view” and “canonical view” to reflect the more general nature of our approach beyond just visual processing.

The three main contributions of this paper are the following. (1) We define a principle that determines whether two observations should be differentiated or generalized to one aspect based on the actor’s capability. (2) We propose an image-based visual servoing algorithm that allows the actor to manipulate an object to cause the features in an image to conform with an aspect in memory. (3) We introduce a method for determining whether a sequence of non-deterministic manipulation actions can, under certain assumptions, be guaranteed to transition between two aspects. We demonstrate our object model and our visual servoing algorithm on a tool-grasping task using the Robonaut 2 simulator.

2 Related Work

Besides work done in aspect graphs and image-based models mentioned in the last section, our work also relates to a body of work in hybrid control theory. In [3], a controller is described as a funnel that guides the robot state to convergence; multiple controllers can be combined to funnel robot states to a desired state that no one single controller can reach alone. In [30], an algorithm that combines linear quadratic regulators into a nonlinear policy was also introduced. However under certain situations the goal state may not be reachable through a combinations of controllers that act like funnels. For example, the visual servoing controller implemented in our experiment controls the end effector to a certain pose based on the robot hand's visual appearance. However to reach the goal state, a controller that transitions from a state where the robot hand is not visible to one in which the visual servoing controller can be executed is required. Such a controller can be an open loop controller that moves the end effector to a memorized pose and may not necessarily converge to a certain state like a funnel.

In this work we introduce the notion of a *slide* as a metaphor for this kind of action that transitions from one set of states to another (see Fig. 1). Uncertainty of the state may increase after transitioning down a slide, but may still reach the goal state if a funnel-slide-funnel structure is carefully designed. We investigate how a sequence of these two kinds of controllers will change how an object is observed. In previous (on-going) work we have referred to funnels as *track* control actions and slides as *search* control actions [11]. The search control action orients the visual sensor to where the target is likely be found therefore transitioning states like a slide; the track control action keeps the target in the visual center and converges to a subset of states like a funnel. Figure 1 illustrates the funnel-slide-funnel concept using the same style of figure demonstrated in previous work by Burridge et al. [3].

There is also a good deal of related work in visual servoing. This work can be classified into two major types: position-based servoing, where servoing is based on the estimated pose; and image-based servoing, where servoing is based directly on visual features [14]. The image-based servoing approach has the advantage that it

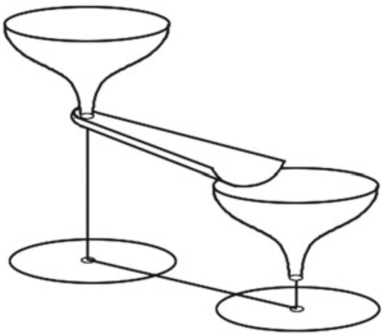


Fig. 1 Funnel-slide-funnel structure. We use the funnel metaphor introduced in [3] to describe a closed-loop controller or a track control action [11] that converges to a subset of states and the slide metaphor to describe an open-loop controller or a search control action [11] that causes state transitions

performs with an accuracy independent of extrinsic camera calibration and does not require an accurate model of the target object or end effector. Our visual servoing approach belongs to this class of image-based servoing techniques.

Our work is inspired by Jägersand and Nelson [15], in which Broyden's method is used to estimate the visuomotor Jacobian online. Our algorithm uses a similar update approach but is implemented on top of a changing set of features. Some other work in visual servoing has also investigated approaches that do not rely on a predefined set of features. In [26], a set of robust SIFT features are selected to perform visual servoing. In [12] moments of SIFT features that represent six degrees of motion are designed. An approach that is based on the image entropy was also introduced in [4]. However these approaches all assume a setting in which the camera is mounted on the end effector. In this article we are interested in a setting that is more similar to human manipulation. Unlike a system where the camera is mounted on the end effector, only part of the observed features move in correspondence with the end effector. Our algorithm is used to guide the robot end effector, within the field of view, to a pose that is defined relative to an object that was memorized. The features that are controllable are learned and reused.

Our work also has many connections to prior work on *affordances*. The term affordance [9] has many interpretations. We prefer the definition of affordance as "the opportunities for action provided by a particular object or environment" [8]. Affordances can be used to explain the functionality and utility of things in the environment. Our object models are based on this interactionist view of perception and action that focuses on learning relationships between objects and actions specific to the robot. An approach to bind affordances of objects with the robot was also introduced by Stoytchev [27]. In this work, the robot learns sequences of actions that will lead to invariant features on objects through random exploration. In the object model introduced in [33], predefined base affordances are associated with object surface types. Instead of defining object affordances from a human perspective, our object models memorize how robot actions change perception with a graph representation.

The aspect transition graph model employed in this work was first introduced by Sen [25]. In our previous work [18, 19], we introduced a mechanism for learning these models without supervision, from a fixed set of actions and observations. We used these models to support belief-space planning techniques where actions are chosen to minimize the expected future model-space entropy, and we showed that these techniques can be used to condense belief over objects more efficiently. In this article we extend the aspect transition graph model to handle an infinite variety of observations and to handle continuous actions. We start with a discussion of our aspect transition graph model.

3 Object Model

The aspect transition graph (ATG) object model discussed in this paper is an extension of the original concept of an aspect graph. In addition to distinctive views, the ATG object model summarizes how actions change viewpoints or the state of the object and thus, the observation. We define the term "observation" to be the combination of all sensor feedback of the robot at a particular time and the "observation space" as the space of all possible observations. This limits the model to a specific robot, but allows the model to present object properties other than viewpoint changes. Extensions to tactile, auditory and other sensors is possible with this representation. An ATG model of an object can be used to plan manipulation actions for that object to achieve a specific target aspect. For example, in order for the robot to pick up an object, the target aspect is a view where the robot's end effector surrounds the object. We expect that this view will be common to many such tasks and that it can be the expected outcome of a sequence of slides (i.e. like moving the effector to the same field of view as the target object) and funnels (like visually servoing features from the hand into the pregrasp configuration relative to the object).

Definitions

We define an "aspect" as a single observation that is stored in the object model. This usage is consistent with the term "canonical view" coined in the psychophysics literature to describe image-based models. As we will see below, many observations will not be stored in the object's memory and hence will not be categorized as aspects. We will discuss in detail below how a given observation is categorized as an aspect or not.

An ATG object model is represented using a directed *multigraph*[1] $G = (\mathcal{X}, \mathcal{U})$, composed of a set of aspect nodes $\mathcal{X}$ connected by a set of action edges $\mathcal{U}$ that capture the probabilistic transition between aspects. An action edge U is a triple (X_1, X_2, A) consisting of a source node X_1, a destination node X_2 and an action A that transitions between them. Note that there can be multiple action edges (associated with different actions) that transition between the same pair of nodes. In contrast to aspect graphs and image-based models that differentiate views based on visual appearance, we argue that, in general, discriminating between object observations should depend on whether the actor is capable of manipulating the object such that the observation converges to a target aspect. That is, we define aspects that are functions of the visual servoing and action abilities of the robot.

Figure 2 shows an example of an ATG model that contains two aspects x_1, x_2 and one action edge u connecting the two aspects in the observation space. An aspect is represented as a single dot in the figure. The smaller ellipses around x_1, x_2 represent the ε-region of the corresponding aspect. Inside the ε-region, the observation is close to the target aspect, and the funnel action is considered to have "converged". The ε-region is task dependent; a task that requires higher precision such as picking up

[1] A multigraph allows multiple edges between a given pair of vertices.

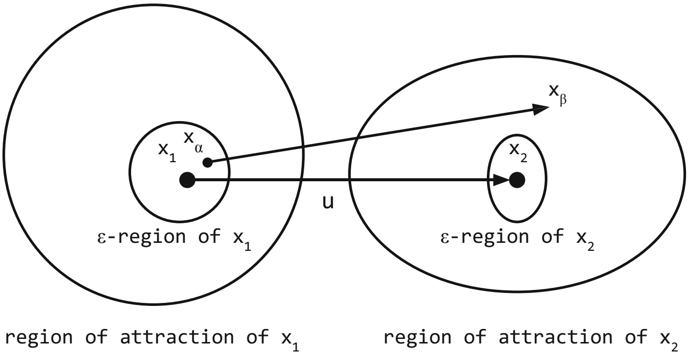


Fig. 2 An ATG model containing two aspects x_1 and x_2, each a likely result of applying a funnel action within their respective regions of attraction. The edge labeled *u* is a model-referenced "slide" action that reliably maps the ε-region of x_1 to the interior of the region of attraction of x_2

a needle will require a smaller ε-region. Each aspect x is located in the ε-region but does not have to be in the center. The location and shape of the ε-region also depends on the given task since certain dimensions in the observation space might be less relevant when performing certain tasks.

The larger ellipses surrounding the ε-regions are the region of attraction of the "funnel" controller referenced to aspects x_1 and x_2. Observations within the region of attraction converge to the ε-region of the target aspect by running a closed-loop controller that does not rely on additional information from the object model. In our experiment, a visual servoing controller is implemented to perform gradient descent to minimize the observation error. The region of attraction for using such a controller is the set of observations from which a gradient descent error minimization procedure leads to the ε-region of the target aspect.

Slides

The arrow in Fig. 2 that connects the two aspects is an action edge (x_1, x_2, a) that represents a "slide" action. Action a is an open-loop controller that causes aspect transitions. Instead of converging to an aspect, "slide" actions tend to increase uncertainty in the observation space. If a funnel is used to describe a convergent controller then a slide is suitable for describing this type of action. Figure 1 illustrates this metaphor with an example structure that allows transitions from a converged aspect to the mouth of another funnel.

We implement slide actions as open-loop controllers. In our experiments, a slide action a can be represented in the form $a = \phi|_{\tau}^{\tilde{\sigma}}$ where ϕ represents the potential function that the controller tries to minimize, $\tilde{\sigma}$ represents a set of memorized controller parameters, and τ represents the motor resources the action controls. An example is an end point position controller that moves to a relative pose with respect to the center of an object point cloud. Under situations when there is no randomness in observation, action execution and the environment, executing action a from aspect x_1 will transition reliably to aspect x_2.

Convergence

The arrow in Fig. 2 that connects the observation x_α within the ε-region of x_1 to observation x_β represents a scenario where action a is executed when x_α is observed in a system in which actions have stochastic outcomes. We define ε_u as the maximum error between the aspect x_2 and the observation x_β when action a is executed while the current observation is within the ε-region of aspect x_1. ε_u can be caused by a combination of kinematic and sensory errors generated by the robot or randomness in the environment. If the region of attraction of the controller that converges to aspect x_2 covers the observation space within ε_u from x_2, by running the convergent controller we are guaranteed to converge within the ε-region of aspect x_2 under such an environment. Figure 1 illustrates this using the funnel and slide metaphor. As long as the end of the slide is within the mouth of the next funnel we can guarantee convergence to the desired state even when open loop controllers are within the sequence. The target aspect x_2 is determined by estimating the most likely observation after executing action a through the Bayesian filtering algorithm.

Completeness and Sufficiency

We call an Aspect Transition Graph model *complete* if the union of the regions of attraction over all aspects cover the whole observation space and a path exists between any pair of aspects. A complete ATG object model allows the robot to manipulate the object from any observation to one of the aspects. Complete ATG object models are informative but often hard to acquire and do not exist for irreversible actions. On the other hand, it is not always necessary to have a complete ATG to accomplish a task. For example, a robot can accomplish most drill related tasks without modeling the bottom of the drill. Therefore, we define an Aspect Transition Graph object model to be *sufficient* if it can be used to accomplish all required tasks of the object. In this work we will focus on sufficient ATG object models.

4 Visual Servoing

In this section we introduce an image-based visual servoing algorithm under the control basis framework [13]. This visual servoing controller is used to converge from an observation within the region of attraction to the ε-region of the corresponding aspect. An action is written in the form $\phi|^\sigma_\tau$, where ϕ is a potential function, σ represents sensory resources allocated, and τ represents the motor resources allocated [13]. The control basis framework provides a means for robot systems to explore combinations of sensory and motor controls. Although only visual data are used in this work, the control basis framework allows us to combine controllers that utilize sensory resources of different modalities in future work. In our experiment the visual

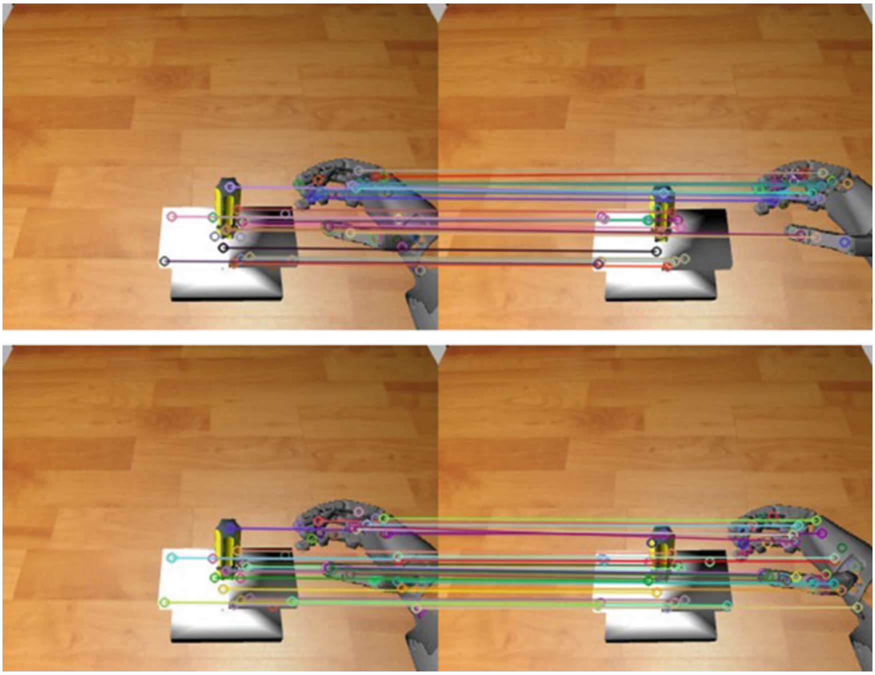


Fig. 3 Visual servoing sequences. Each image pair shows the target aspect (*left*) and the current observation (*right*). A *line* in between represents a pair of matching keypoints. The *top* image pair represents the starting observation and the *bottom* image pair represents when the controller converged

servoing controller is used to control the end effector of the robot to reach a pose relative to a target object using visual sensor feedback. Unlike many visual servoing approaches, our visual servoing algorithm does not require a set of predefined visual features on the end effector or target object nor does it require an inverse kinematic solution for the robot. The only information required is the current observation and the target aspect. Figure 3 shows a trial of our visual servoing algorithm converging to a stored target aspect.

Potential Function

In the control basis framework, a potential function ϕ represents an error function that the controller minimizes. To reach minimum error a closed loop controller performs gradient descent on the potential function to converge to a minimum. Artificial potential functions that guarantee asymptotically stable behavior are usually used to avoid local minima [11]. However in visual servoing, potential functions with a unique minimum often do not exist due to occlusion, lighting and noisy sensory data. Instead of trying to define a potential function with a unique minimum, we define a potential function with possibly many local minima and call the region in

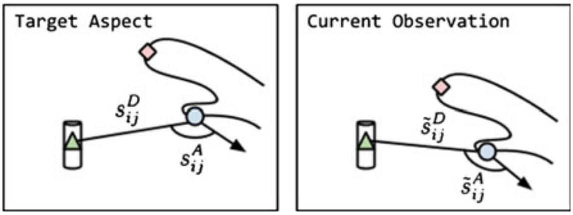


Fig. 4 Components of the signature of the target aspect (*left*) and the current observation (*right*). The circle and the triangle represent the ith and jth matched keypoints

which gradient descent converges to a particular minimum the region of attraction. If the current aspect is within the region of attraction we can guarantee convergence to the target aspect through gradient descent.

Our potential function is defined as the weighted squared Euclidean distance between the signature of the current observation $\tilde{\boldsymbol{s}}$ and the signature of the target aspect $\boldsymbol{s}$. This approach can be used with most feature detectors and feature descriptors. In our experiment the Fast-Hessian detector and the SURF descriptor [1] are implemented. A depth filter that uses the depth image is first used to filter out most keypoints that belong to the background. The first step to calculate the signature of an observation is to find a subset K of keypoints in the current observation that match to keypoints in the target aspect. The signature of an observation can then be calculated based on this subset K of keypoints. The signature is a combination of the distance signature vector $\boldsymbol{s}^D$ and the angle signature vector $\boldsymbol{s}^A$. $\boldsymbol{s}^D$ is a signature vector that consists of Euclidean distances s^D_{ij} between all pairs of keypoints (k_i, k_j) in K: $s^D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Here x_i, y_i are the X Y image coordinates of keypoint $k_i \in K$. The angle signature vector $\boldsymbol{s}^A$ consists of angle differences s^A_{ij} between all pairs of keypoints (k_i, k_j) in K: $s^A_{ij} = \omega_{ij} - \theta_i$. Here ω_{ij} represents the orientation of the ray from keypoint k_i to keypoint k_j and θ_i represents the orientation of keypoint k_i. Figure 4 illustrates examples of s^D_{ij} and s^A_{ij} of the target aspect and the current observation.

The potential ϕ is then the scaled squared Euclidean distance between distance signature vectors of the target aspect $\boldsymbol{s}^D$ and the current observation $\tilde{\boldsymbol{s}}^D$ plus the weighted squared Euclidean distance between angle signature vectors of the target aspect $\boldsymbol{s}^A$ and the current observation $\tilde{\boldsymbol{s}}^A$;

$$\phi = \frac{1}{N_D} \cdot \sum_{\{i,j|k_i,k_j \in K\}} (s^D_{ij} - \tilde{s}^D_{ij})^2 + \sum_{\{i,j|k_i,k_j \in K\}} w^A_{ij} \cdot (s^A_{ij} - \tilde{s}^A_{ij})^2,$$

where $N_D = |K| \cdot (|K| - 1)/2$ and $w^A_{ij} = s^D_{ij} / \sum_{\{i,j|k_i,k_j \in K\}} s^D_{ij}$. Here $|K|$ is the number of matched keypoints between the current observation and the target aspect and w^A_{ij} is a normalized weight proportional to the keypoint pair distance s^D_{ij} in the target aspect. The purpose of w^A_{ij} is to weight angle differences more heavily for keypoints that are far apart.

Gradient Descent

In order to perform gradient descent on the potential function we need to be able to estimate the potential-motor Jacobian defined as $J = \partial\phi(\sigma)/\partial\boldsymbol{\tau}$. A seven degree freedom arm is used in our experiment, therefore $\boldsymbol{\tau} = [q_1, q_2, \ldots, q_7]$ where q_i represents the ith joint in Robonaut-2's right arm. The control signal that leads to the greatest descent can then be calculated by the expression: $\Delta\boldsymbol{\tau} = -c(J^{\#}\phi(\sigma))$, where c is a positive step size and $J^{\#}$ is the Moore–Penrose pseudoinverse [22].

In order to calculate the partial derivative of the potential function ϕ with respect to each joint q, we introduce the visuomotor Jacobian defined as $J_v = \partial V/\partial\boldsymbol{\tau}$, where V is the X Y positions and orientations of the set of keypoints detected in the current observation that match to keypoints in the target aspect based on its feature descriptor. Given $\Delta\boldsymbol{\tau}$ and J_v we can calculate the change in the keypoint positions and angles through $\Delta V = J_v \cdot \Delta\boldsymbol{\tau}$. Since the potential only depends on matched pairs we can calculate an estimated potential for every joint value.

Learning the Visuomotor Jacobian

Our visuomotor Jacobian that models how features change with respect to joint values is inspired by work done in understanding how humans obtain a sense of agency by observing their own hand movements [32]. Our approach learns that certain feature positions on the robot end effector are controllable while features in the background are not. Our visuomotor Jacobians for each aspect are updated online using a Broyden-like method $J_{v_{t+1}} = J_{v_t} + (\mu(\Delta V - J_{v_t}\Delta\boldsymbol{\tau})\Delta\boldsymbol{\tau}^T/\Delta\boldsymbol{\tau}^T\Delta\boldsymbol{\tau})$, where J_{v_t} is the visuomotor Jacobian at time t and $\mu \in (0, 1]$ is a factor that specifies the update rate [21]. When $\mu = 1$ the updating formula will converge to the correct Jacobian J_v after m noiseless orthogonal moves and observations, where m is the dimension of J_v. In our experiment we set $\mu = 0.1$ to make the estimation more robust. The visuomotor Jacobians for each aspect are initialized randomly for the first run and memorized afterwards. The more trials the controller runs the more accurate the estimated J_v is on average. Using Broyden's method to estimate Jacobians on-line for visual servoing was first introduced in [15].

5 Experimental Results

The aspect transition graph object model in conjunction with the visual servoing algorithm introduced in previous sections are tested on a tool grasping task on the NASA Robonaut-2 simulator [5]. The goal of the task is to control Robonaut-2's right hand to a pose where a screwdriver on a tool stand is in between the robot's right thumb, index finger and middle finger as shown in Fig. 5. An ATG object model consisting of three aspects, that is sufficient for this task, was built through demonstration. We show that the "slide-funnel-slide-funnel" controller sequence decreases the average pose error over a "slide-slide" controller sequence.

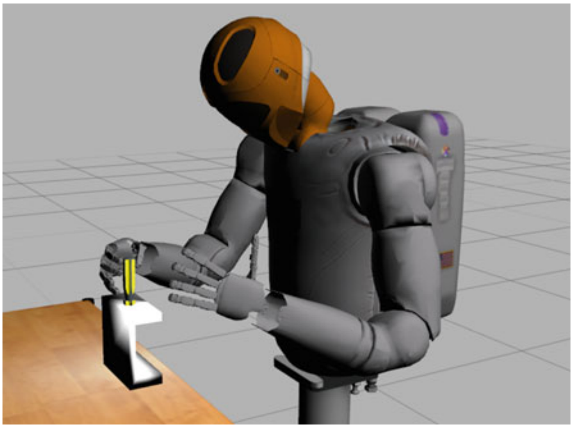


Fig. 5 Robonaut 2 approaching a pregrasp pose for a screwdriver on a tool stand in simulation

Building ATG Models

In this experiment our ATG object model is built through a teleoperated demonstration. An interface was implemented to allow the demonstrator to indicate when to create a new aspect in the object model. The demonstrator can control the robot end effector through interactive markers implemented by the MoveIt! platform [28]. When a new aspect is created, the action edge that connects the previous aspect to this new aspect can be inferred.

The ATG object model used in this experiment consists of three aspects. The first aspect represents an observation in which the screwdriver is on a tool stand on a table and is 0.6 m in front of the robot. In addition, no parts of the robot are visible. The left image in Fig. 6 is the corresponding observation of this aspect. The second aspect represents an observation where the robot's right hand is about 0.07 m right of the screwdriver. The action edge between the first and second aspects represents an action that moves the robot's right hand to a pose relative to the center of the segmented point cloud observed in the first aspect. This point cloud is segmented based on the distance to the camera. The middle image in Fig. 6 is the corresponding observation of this aspect. The third aspect represents an observation where the robot's right thumb, index and middle finger surrounds the screwdriver handle. The right image in Fig. 6 is the corresponding observation of this aspect. The action edge in between the second and third aspects represents an action that moves the robot's right hand to a pose relative to the right hand pose of the previous aspect. The relative action frame is determined based on the closest observable feature to the end effector. An even better approach would be to assign action frames based on the intention of the demonstrator but this is beyond the scope of this paper.

Region of Attraction

The region of attraction of the second and third aspect of the ATG object model with respect to the visual servoing controller can be analyzed. It is possible to also have a

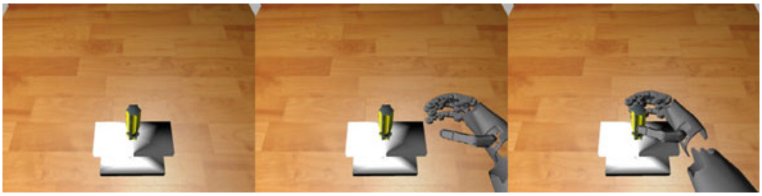


Fig. 6 The first, second, and third aspect stored in the ATG model through demonstration are shown from *left* to *right*. In the first aspect, the object on top of the table is a screwdriver on a tool stand. In the second aspect, the robot hand is in a position where a straight movement toward the screwdriver would lead to a pregrasp pose. The third aspect represents a pregrasp pose. This is the goal aspect for the pregrasp task designed in this experiment

controller that is capable of converging to the first aspect through controlling joints in the robot's neck and waist, however since we assume the robot starts in a similar pose with similar observation this controller is not implemented in this experiment. The region of attraction of an aspect is defined as the observation space in which a closed loop convergence controller that does not rely on additional information from the object model can converge to the ε-region of the aspect. An aspect or observation lies in a high dimensional observation space and can be varied by multiple different parameters or noise. In this experiment we are interested in two types of noise. (1) Noise in the relative pose between the robot hand and the object. This kind of noise can be caused by kinematic errors from executing an action or imperfect object positions calculated from a noisy point cloud. This type of noise will result in a different end effector pose relative to the object. (2) Noise in the object position. This kind of noise can be caused by placing the tool stand and screwdriver in a different position than the position previously observed in the demonstration. This type of noise can cause the estimated object center position to vary and will affect the visual servoing controller since the object and the robot end effector will look visually different from a different angle. In this experiment our goal is to find the region of attraction of the second and third aspects with respect to these two kinds of noise.

These two kinds of noise are artificially added to our experiment and the number of gradient descent iterations required to reach the ε-region of the aspect are recorded. In this experiment we only consider noise on the X-Y plane for easier visualization and analysis. For each type of noise and each aspect we tested 289 different combination of noise in the X and Y axes roughly within the scale that the visual servoing controller can handle. The results for adding noise in the relative pose between the robot hand and the object to the second aspect are shown in Fig. 7. The plot on the left indicates how many iterations the visual servoing controller executed till convergence for different noise values. Each color tile is one single experiment and dark blue means the controller converges fast while orange means the controller took longer to converge. A yellow tile means that the controller could not converge within the 1000 iteration threshold. We call the region of attraction the set of observations that include the aspect plus the set of noise positions that corresponds to a

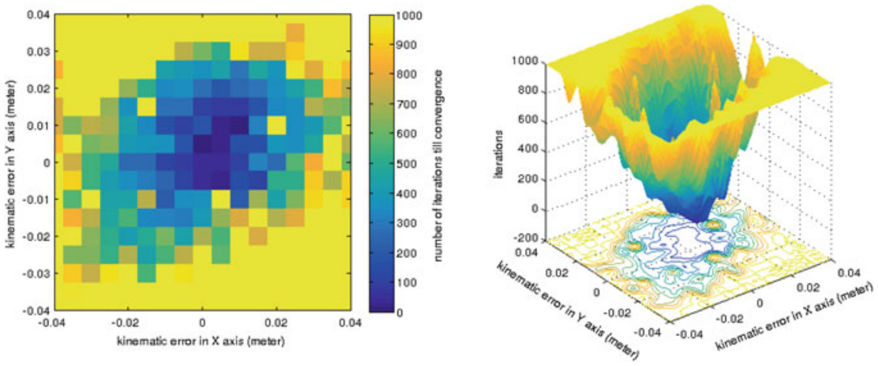


Fig. 7 Iteration till convergence with respect to noise in the relative pose between the robot hand and the object for the second aspect

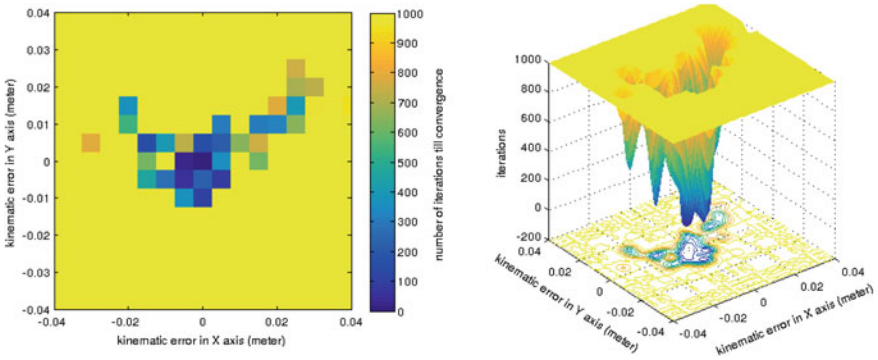


Fig. 8 Iteration till convergence with respect to noise in the relative pose between the robot hand and the object for the third aspect

non yellow tile connected to the origin. The plot on the right is a visualization of the same result in 3D which has some resemblance to the funnel metaphor used in Fig. 1.

The results for adding noise in the relative pose between the robot hand and the object to the third aspect are shown in Fig. 8. Note that this aspect has a smaller region of attraction with more tolerance in the direction perpendicular to the hand opening. If there is a large error in the *Y* axis the robot's hand may end up in front or behind the screwdriver. Under such situations without additional information the visual servoing controller will not be able to avoid colliding with the screwdriver while trying to reach the goal. The results for adding noise in the object position are shown in Fig. 9. Notice that the regions of attraction are much larger for this type of noise.

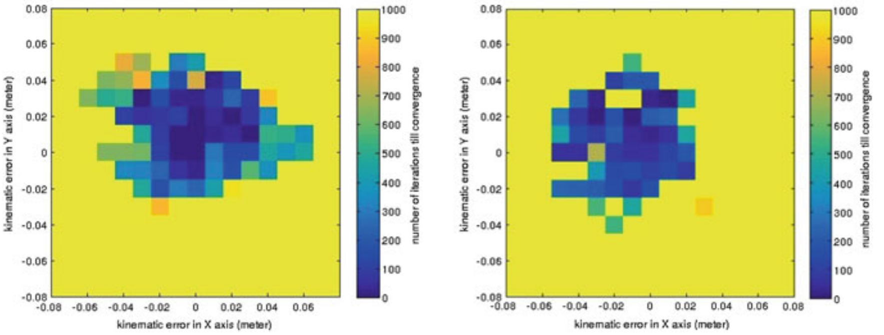


Fig. 9 Iteration till convergence with respect to noise in the object position for the second aspect (*left image*) and the third aspect (*right image*)

Convergence and Accuracy

By analyzing the observed regions of attraction of the visual servo controller that converges to the two aspects we can estimate the magnitude of noise this "slide-funnel-slide-funnel" controller sequence can tolerate. Through Figs. 7 and 8 we can see that the visual servo controller has a region of attraction with about 1.5 cm radius of kinematic noise around the second aspect and about 0.5 cm radius of kinematic noise around the third aspect. We evaluate these sequences of actions by comparing the final end effector position in the X-Y plane to the demonstrated pose relative to the screwdriver. We tested noise of three different magnitudes to each open-loop action; 0.5, 1.0, and 1.5 cm for the action that transitions from the first aspect to the second aspect and 0.1, 0.2, and 0.3 cm for the action that transitions from the second aspect to the third aspect. For each combination of noise we test eight uniformly distributed

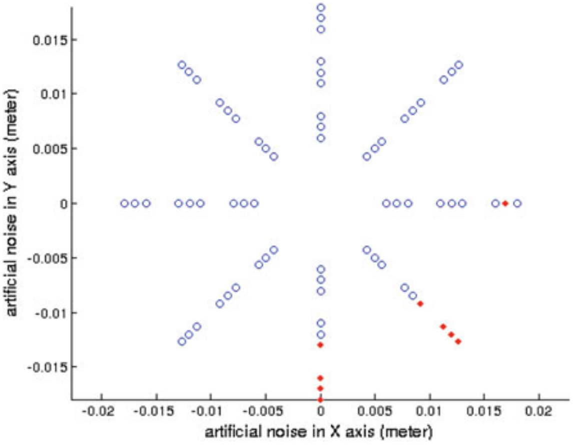


Fig. 10 Convergence with respect to artificial noise added to the test cases. Each *dot* represents a test case where the X Y value represents the summed magnitude and direction of the manually added kinematic noise. A *red* diamond indicates that the controller fails to converge to the third aspect while a *blue circle* indicates that the action sequence converged

Table 1 Average position error in the X-Y plane in centimeters

	Complete test set (cm)	"Slide-funnel-slide-funnel" structure converged test set (cm)
"slide-slide" structure	2.24	2.06
"slide-funnel-slide-funnel" structure	0.99	0.75

directions. Among the 72 test cases 100% of them converged to the second aspect and 87.5% of them converged to the third aspect.

We did not reach a 100% overall convergence rate for two possible reasons. First, in addition to the artificial noise, randomness in the action planner and simulator also exist in the system. Second, the region of attractions shown in the previous section are estimated based on visual similarity. Two observations can be visually similar but position wise quite different therefore causing a false estimate of convergence. Figure 10 shows the test cases that the controller fails to converge on; most of the failed test cases are located in the lower right corner. This is consistent with the shape of the region of attraction of the controller with respect to the third aspect shown in Fig. 8. The final poses of the end effector relative to the screwdriver are recorded and compared to the demonstrated pose.

We further compare the result to a sequence of "slide-slide" controllers without visual servoing acting as a funnel. The average position error is shown in Table 1. The "slide-funnel-slide-funnel" structure reduces the error by 55.8% and has an average error of 0.75 cm in the X-Y plane when only considering test cases that converged.

6 Conclusion

In this paper we introduce an image-based object model that categorizes different observations of an object into a subset of aspects based on interactions instead of only on visual appearance. We further propose that a sequence of controllers that form a "funnel-slide-funnel" structure based on this object model can have high rates of success even when open-loop controllers are within the sequence. To demonstrate this proposition we created an aspect transition graph object model that represents a pregrasp action through a teleoperation demonstration. In addition, we introduced a novel visual servoing controller that funnels the current observation to a memorized aspect using a changing set of visual features. The regions of attraction with respect to the end effector pose of the visual servoing controller are then identified by manually adding kinematic noise to the end effector position. Based on this region of attraction we identified the magnitude of kinematic noise this sequence of controllers is capable of handling and showed that under an environment with a similar magnitude of noise this sequence of actions decreases the average final position error significantly.

The biggest drawback of the current approach is its scalability to model more complex objects. In this work we define aspects by manually indicating meaningful observations. In future work we plan to identify transitions autonomously and investigate hierarchical models that reuse existing sub-structures.

Acknowledgements The authors would like to thank Mitchell Hebert, Hee-Tae Jung, Michael Lanighan, Dirk Ruiken, Shiraj Sen, and Takeshi Takahashi for their contributions. This material is based upon work supported under Grant NASA-GCT-NNX12AR16A and a NASA Space Technology Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. Computer vision–ECCV 2006, pp. 404–417. Springer, Heidelberg (2006)
2. Bülthoff, H.H., Edelman, S.: Psychophysical support for a two-dimensional view interpolation theory of object recognition. Proc. Natl. Acad. Sci. **89**(1), 60–64 (1992)
3. Burridge, R.R., Rizzi, A.A., Koditschek, D.E.: Sequential composition of dynamically dexterous robot behaviors. Int. J. Robot. Res. **18**(6), 534–555 (1999)
4. Dame, A., Marchand, E.: Entropy-based visual servoing. In: IEEE International Conference on Robotics and Automation, ICRA'09, 2009, pp. 707–713. IEEE (2009)
5. Dinh, P., Hart, S.: NASA Robonaut 2 Simulator (2013). Accessed 7 July 2014
6. Edelman, S., Bülthoff, H.H.: Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. Vision Res. **32**(12), 2385–2400 (1992)
7. Faugeras, O., Mundy, J., Ahuja, N., Dyer, C., Pentland, A., Jain, R., Ikeuchi, K., Bowyer, K.: Why aspect graphs are not (yet) practical for computer vision. CVGIP: Image Underst. **55**(2), 212–218 (1992)
8. Gibson, J.J.: The Perception of the Visual World. Houghton Mifflin, Massachusetts (1950)
9. Gibson, J.J.: Perceiving, acting, and knowing: toward an ecological psychology. The Theory of Affordance. Lawrence Erlbaum Associates, Michigan (1977)
10. Gigus, Z., Malik, J.: Computing the aspect graph for line drawings of polyhedral objects. IEEE Trans. Pattern Anal. Mach. Intell. **12**(2), 113–122 (1990)
11. Hart, S.W.: The development of hierarchical knowledge in robot systems. Ph.D. Thesis, University of Massachusetts Amherst (2009)
12. Hoffmann, F., Nierobisch, T., Seyffarth, T., Rudolph, G.: Visual servoing with moments of sift features. In: IEEE International Conference on Systems, Man and Cybernetics, 2006, SMC'06. vol. 5, pp. 4262–4267. IEEE (2006)
13. Huber, H.: A hybrid architecture for adaptive robot control (2000)
14. Hutchinson, S., Hager, G.D., Corke, P.I.: A tutorial on visual servo control. IEEE Trans. Robot. Autom. **12**(5), 651–670 (1996)
15. Jägersand, M., Nelson, R.: On-line estimation of visual-motor models using active vision. Image **11**, 1 (1996)
16. Koenderink, J.J., van Doorn, A.J.: The internal representation of solid shape with respect to vision. Biol. Cybern. **32**(4), 211–216 (1979)
17. Kriegman, D.J., Ponce, J.: Computing exact aspect graphs of curved objects: solids of revolution. Int. J. Comput. Vision **5**(2), 119–135 (1990)
18. Ku, L.Y., Sen, S., Learned-Miller, E.G., Grupen, R.A.: Action-based models for belief-space planning. In: Workshop on Information-Based Grasp and Manipulation Planning, at Robotics: Science and Systems (2014)

19. Ku, L.Y., Sen, S., Learned-Miller, E.G., Grupen, R.A.: Aspect transition graph: an affordance-based model. In: Second Workshop on Affordances: Visual Perception of Affordances and Functional Visual Primitives for Scene Analysis, at the European Conference on Computer Vision (2014)
20. Logothetis, N.K., Pauls, J., Poggio, T.: Shape representation in the inferior temporal cortex of monkeys. Current Biol. **5**(5), 552–563 (1995)
21. More, J.J., Trangenstein, J.A.: On the global convergence of broydens method. Math. Comput. **30**(135), 523–540 (1976)
22. Nakamura, Y.: Advanced Robotics: Redundancy and Optimization. Addison-Wesley, Boston (1991)
23. Palmeri, T.J., Gauthier, I.: Visual object understanding. Nat. Rev. Neurosci. **5**(4), 291–303 (2004)
24. Poggio, T., Edelman, S.: A network that learns to recognize 3d objects. Nature **343**(6255), 263–266 (1990)
25. Sen, S.: Bridging the gap between autonomous skill learning and task-specific planning. Ph.D. Thesis, University of Massachusetts Amherst (2013)
26. Shademan, A., Janabi-Sharifi, F.: Using scale-invariant feature points in visual servoing. In: Optics East, International Society for Optics and Photonics, pp. 63–70 (2004)
27. Stoytchev, S.: Toward learning the binding affordances of objects: a behavior-grounded approach. In: Proceedings of AAAI Symposium on Developmental Robotics, pp. 17–22 (2005)
28. Sucan, I.A., Chitta, S.: Moveit! (Online)
29. Tarr, M.J., Bülthoff, H.H.: Image-based object recognition in man, monkey and machine. Cognition **67**(1), 1–20 (1998)
30. Tedrake, R.: LQR-trees: feedback motion planning on sparse randomized trees (2009)
31. Ullman, S., Basri, R.: Recognition by linear combinations of models. IEEE Trans. Pattern Anal. Mach. Intell. **13**(10), 992–1006 (1991)
32. Van Den Bos, E., Jeannerod, M.: Sense of body and sense of action both contribute to self-recognition. Cognition **85**(2), 177–187 (2002)
33. Varadarajan, K.M., Vincze, M.: AfRob: the affordance network ontology for robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 1343–1350. IEEE (2012)

An Approximate Inference Approach to Temporal Optimization for Robotics

Konrad Rawlik, Dmitry Zarubin, Marc Toussaint and Sethu Vijayakumar

1 Introduction

Control of sensorimotor systems, artificial or biological, is inherently both a spatial and temporal process. Not only do we have to specify *where* the plant has to move to but also *when* it reaches that position. In some control schemes, the temporal component is implicit. For example, with an infinite horizon, discounted cost based controller, movement duration results from the application of the feedback loop. In other cases it is explicit, for example in finite horizon objective based formulations, where the time horizon is set explicitly as a parameter of the problem [10].

Although control based on an optimality criterion is certainly attractive, practical approaches for stochastic systems are currently limited to the finite horizon objective or the first exit time objective. The former does not optimize temporal aspects of the movement, i.e., duration or the time when costs for specific sub-goals of the problem are incurred, assuming them as given *a priori*. However, how should one choose these temporal parameters? This question is non-trivial and important, even when considering a simple reaching problem. The solution of using an *a priori* fixed duration, chosen experimentally, can result in not reaching the goal, having to use an unrealistic range of control commands or excessive (wasteful) durations for short distance tasks. The alternative first exit time formulation, on the other hand, either

K. Rawlik (✉) · S. Vijayakumar
University of Edinburgh, Scotland, UK
e-mail: konrad.rawlik@roslin.ed.ac.uk

S. Vijayakumar
e-mail: sethu.vijayakumar@ed.ac.uk

D. Zarubin · M. Toussaint
University of Stuttgart, Stuttgart, Germany
e-mail: dmitry.zarubin@ipvs.uni-stuttgart.de

M. Toussaint
e-mail: marc.toussaint@informatik.uni-stuttgart.de

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_9

assumes specific exit states in the cost function and computes the shortest duration trajectory which fulfils the task, or assumes a time stationary task cost function and computes the control which minimizes the joint cost of movement duration and task cost [1, 5, 15]. This is directly applicable only to tasks which do not require sequential achievement of multiple goals. While this limitation could be overcome by chaining together individual time optimal single goal controllers, such a sequential approach has several drawbacks. First, if we are interested in placing a cost on overall movement duration, we are restricted to linear costs if we wish to remain time optimal. A second more important flaw is that future goals should influence our control even before we have achieved the previous goal.

In this paper, we extend standard finite horizon Stochastic Optimal Control (SOC) problem formulation with additional cost terms on temporal aspects of a control policy.

2 Problem Formulation

2.1 Finite Horizon Stochastic Optimal Control Problem

Let us consider a general controlled process, with state $x \in \mathbb{R}^{D_x}$ and controls $u \in \mathbb{R}^{D_u}$, given by the stochastic differential equation of the form

$$\mathrm{d}x = f(x, u)\,\mathrm{d}t + \mathrm{d}\xi \;, \quad \left\langle \mathrm{d}\xi\;\mathrm{d}\xi^{\top} \right\rangle = Q \;. \tag{1}$$

with non-linear dynamics f and Brownian motion ξ. Fixing a finite time horizon t_f we denote by $x(\cdot)$ and $u(\cdot)$ the state and control trajectories over the interval $t \in [0, t_f]$. For a given state-control trajectory we define the cost function as

$$C(x(\cdot), u(\cdot)) = \int_0^{t_f} c(x(t), u(t), t)\,\mathrm{d}t + c_f(x(t_f)) \;, \tag{2}$$

where $c(x, u, t)$ is a cost rate for being in state x and applying controls u at time t, and c_f denotes a final state cost term. The finite horizon stochastic optimal control problem is to find the (non-stationary) control policy $\pi^* : (x, t) \to u$ that minimizes the expected total cost given a start state $x(0)$ and t_f,

$$\pi^* = \underset{\pi}{\operatorname{argmin}} \left\langle C(x(\cdot), u(\cdot)) \right\rangle_{x(\cdot), u(\cdot)|\pi, x(0)} \;. \tag{3}$$

Here we take the expectation w.r.t. the distribution $P(x(\cdot), u(\cdot) \mid \pi, x(0))$ over state-control trajectories conditional on the given start state and control policy.

2.2 Temporal Optimisation Problem

In practical robotics applications cost can generally be divided into subgoals, where costs depend only on state and incur at intermediate time instances, and stationary costs incurred throughout the movement. We express this by considering a cost of the following form,

$$C(x(\cdot), u(\cdot), \mathscr{T}) = \int_0^{t_f} c(x(t), u(t))\,\mathrm{d}t + \sum\nolimits_{i=1}^{f} c_i(x(t_i)) + C_{\mathscr{T}}(\mathscr{T}) \tag{4}$$

where $\mathscr{T} = \{t_1, \dots, t_f\}$ is a set of time instances—the *time course*—at which specific subgoals, captured by the corresponding c_i's, are to be fulfilled. For instance, in a reaching movement, a cost that is a function of the distance to the target is incurred only at the final time t_f, while intermediate costs may represent subgoals like the alignment of an orientation some time *before* the reaching of a target. In our temporal optimisation framework, our objective shall be the optimisation of the time course $\mathscr{T}$ itself, including an explicit cost term $C_{\mathscr{T}}(\mathscr{T})$ that arbitrarily penalizes these time intervals. Note that this objective is broader than the duration optimisation, i.e., choice of only t_f, but of course includes it as the special case $\mathscr{T} = \{t_f\}$.

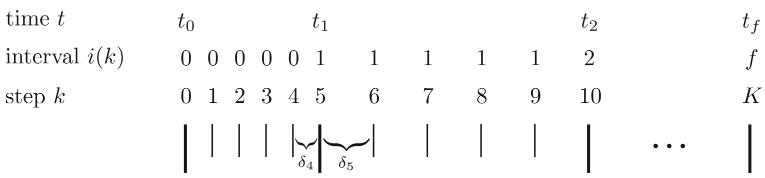
The problem now is to find the joint optimum for the control policy and the time course $\mathscr{T}$,

$$(\pi^*, \mathscr{T}^*) = \underset{\pi, \mathscr{T}}{\operatorname{argmin}} \left\langle C(x(\cdot), u(\cdot), \mathscr{T}) \right\rangle_{x(\cdot), u(\cdot) | \pi, x(0)} . \tag{5}$$

2.3 Time Discretization

While our approach can equally be described fully in a continuous time framework, the presentation will be simplified when assuming a time discretization. Below we briefly discuss a continuous time formulation.

We discretize the time interval $[0, t_f]$ in K time steps, where each interval $[t_i, t_{i+1}]$ is discretized in K/f steps of uniform length $\delta_k = (t_{i(k)+1} - t_{i(k)})/K/f$, and $i(k) = \lfloor fk/K \rfloor$ denotes the interval that the k^{th} time step belongs to (see Fig. 1 for illustration). Conversely, by $k(i) = iK/f$ we denote the step index that



time t	t_0					t_1					t_2		t_f
interval $i(k)$	0	0	0	0	0	1	1	1	1	1	2		f
step k	0	1	2	3	4	5	6	7	8	9	10		K

Fig. 1 Illustration of the notation used (in the case $K/f = 5$)

corresponds to the i^{th} intermediate cost c_i. Choosing different numbers of time steps per interval $[t_i, t_{i+1}]$ of non-uniform step lengths is a straight-forward extension of all the following.

In the discrete time case the problem takes the general form

$$x_{k+1} = f(x_k, u_k, \delta_k) + \varepsilon \;, \quad \varepsilon \sim \mathcal{N}(0, Q(\delta_k)) \tag{6}$$

$$C(x_{1:K}, u_{1:K}, \mathscr{T}) = \sum\nolimits_{k=0}^{K} c(x_k, u_k)\delta_k + \sum\nolimits_{i=1}^{f} c_i(x_{k(i)}) + C_{\mathscr{T}}(\mathscr{T}) \tag{7}$$

although the ideas presented can be easily adapted to alternative forms. For notational convenience, we will absorb the task costs $c_i(x_{k(i)})$ in the running costs by defining

$$\tilde{c}_k(x_k, u_k, \delta_k) = c(x_k, u_k)\delta_k + [k\%K = 0]\; c_{i(k)}(x_k) \tag{8}$$

$$C(x_{1:K}, u_{1:K}, \mathscr{T}) = \sum\nolimits_{k=0}^{K} \tilde{c}_k(x_k, u_k, \delta_k) + C_{\mathscr{T}}(\mathscr{T}) \;, \tag{9}$$

where $k\%K$ denotes the modulo operator.

If instead we would like to stay in a time continuous framework we would define $d(t)$ as a function of time, thus augmenting the state space by a dimension. The quantity $d(t)$ can be regarded as a general resource variable and the general problem formulation (4) reformulated as a first exit time problem - details can be found in [8]. Several algorithms applicable to problems with general non-linear dynamics have been developed, e.g. DDP [11], ILQG [13] to name a couple, all of which can be directly applied to this reformulation of the temporal optimisation problem. However, our experience has shown that naive application of such algorithms, in particular those listed, to the problem of temporal optimisation fails. This is generally due to the nature of these approximate algorithms as local optimisers. With a poor initialisation, setting $d(\cdot) = 0, \pi(\cdot, \cdot) = 0$, i.e., not moving for no time or close approximations thereof, often proves to be a dominant local minimum. We are therefore compelled to seek alternative optimisation schemes, which avoid the collapse of the solution to such undesirable outcomes. In the following we describe an approach based on alternate optimisation of the policy and $\mathscr{T}$. This is formulated in the AICO framework, which frames the problem as an inference problem, although a similar approach can be followed within classical stochastic optimal control formulations leading to similar results.

3 Approximate Inference Approach

In previous work [9] it has been shown that a general SOC problem can be reformulated in the context of approximate inference, or more precisely, as a problem of minimizing a Kullback–Leibler divergence. This alternative problem formulation is useful in particular for derivation of approximation methods which would be non-obvious to derive in the classical formulation. In the following we will adopt the approximate inference perspective to propose a specific approximation method to solve the temporal optimization problem.

3.1 AICOT Formulation

In the inference control formulation, given a stochastic control policy $\pi_k(u_k|x_k)$ we define the process

$$P(x_{1:K}, u_{1:K}|\pi, \mathscr{T}) = \pi(u_0|x_0) \prod_{k=1}^{K} \pi(u_k|x_k) P(x_{k+1}|x_k, u_k, \delta_k) \,, \tag{10}$$

where $P(x_{k+1}|x_k, u_k, \delta_k)$ is given by the discrete time dynamics (6). We further introduce an auxiliary (binary) random variable r_k with the likelihood

$$P(r_k = 1|x_k, u_k, \mathscr{T}) = \exp\{-\eta \tilde{c}_k(x_k, u_k, \delta_k)\} \,, \tag{11}$$

which can be interpreted as indicating (probabilistically) whether a task is fulfilled (or rather whether costs are low). It is straight-forward to verify that

$$C(x_{1:K}, u_{1:K}, \mathscr{T}) - C_{\mathscr{T}}(\mathscr{T}) = -\log P(r_{1:K} = 1|x_{1:K}, u_{1:K}) \,, \tag{12}$$

that is, we translated task and control costs into neg-log-likelihoods. In [9] it has been show how *for fixed* $\mathscr{T}$ computing the posterior process $P(x_{1:K}, u_{1:K}|r_{1:K} = 1, \mathscr{T})$, that is, the distribution over state-control trajectories conditioned on *always* observing "task fulfillment" is related to solving the stochastic optimal control problem. In particular, this posterior also includes the posterior policy $P(u_k|x_k, r_{1:K} = 1, \mathscr{T})$, i.e. the posterior probability of choosing a control u_k in state x_k conditioned on constant "task fulfillment", which can be used in an interactive procedure to find the optimal control policy.

In the context of temporal optimisation we are interested in the computation of the posterior

$$P(\mathscr{T}, x_{1:K}, u_{1:K}|r_{1:K} = 1) \propto P(x_0) \prod_{k=0}^{K} P(x_{k+1}|x_k, u_k, \mathscr{T}) \exp\{-C(x_{1:K}, u_{1:K}, \mathscr{T})\} \,.$$

From this the MAP policy, and in this case MAP $\mathscr{T}$, are extracted. As this problem will in general be intractable, we proceed in two steps

$$\mathscr{T}^{\text{MAP}} = \underset{\mathscr{T}}{\text{argmax}}\, P(\mathscr{T} \mid r_{1:K} = 1) \tag{13}$$

$$\pi^{\text{MAP}} = \underset{\pi}{\text{argmax}}\, P(\pi \mid \mathscr{T}^{\text{MAP}}, r_{1:K} = 1) \tag{14}$$

Note that the second step reduces exactly to standard AICO and may be solved with any of the methods proposed by [9, 14]. The main focus in the following is therefore on solving (13). The proposed approach is based on an iterative procedure alternating between approximation of the distribution $P(x_{1:K}, u_{1:K} | \mathscr{T}^{\text{old}}, r_{1:K} = 1)$ and utilisation of this distribution to obtain an improved $\mathscr{T}^{\text{new}}$. We call this general method AICOT. Two alternative forms of the improvement step are proposed, one gradient and one EM based. The relative merits of these two methods are then discussed in Sect. 3.4.

3.2 Gradient Descent

We first consider direct optimisation of (13) by gradient descent. Let

$$\mathscr{L}(\mathscr{T}) = \log P(\mathscr{T} \mid r_{1:K} = 1) \tag{15}$$

and note that

$$\nabla \mathscr{L}(\mathscr{T}) \propto \frac{1}{\mathscr{L}(\mathscr{T})} \cdot \nabla P(r_{1:K} = 1 | \mathscr{T}) - \nabla C_{\mathscr{T}}(\mathscr{T})$$

In the general case $P(r_{1:K} = 1 | \mathscr{T})$ will not be tractable. We therefore propose taking, similar to the standard AICO algorithms, a Gaussian approximation. For brevity, let $z_{1:K} = (x_{1:K}, u_{1:K})$ denote the state-control trajectory. We define

$$\tilde{p}(z_{1:K} | \mathscr{T}) \approx P(r_{1:K} = 1, z_{1:K} | \mathscr{T})$$

as the unnormalized Gaussian approximation to $P(z_{1:K} | r_{1:K} = 1, \mathscr{T})$. Using this approximation

$$\nabla_{\mathscr{T}} \mathscr{L}(\mathscr{T}) \approx \nabla_{\mathscr{T}} \int_{z_{1:K}} \tilde{p}(z_{1:K} | \mathscr{T}) \, .$$

We derive the approximate gradient, assuming a state-control LQ approximation, that is, we consider (6) and (9) are locally in the form

$$f(z_k, \delta_k) \approx a_k(\delta_k) + A_k(\delta_k)\, z_k + B_k(\delta_k) u_k \ , \quad Q_k = Q \delta_k \tag{16}$$

$$\tilde{c}_k(z_k, k) \approx [k \% K = 0] \frac{1}{2} x_k^\top C_k(\mathscr{T}) x_k - c_k(\mathscr{T})^\top x_k + \frac{1}{2} u_k^\top H u_k \ , \tag{17}$$

where all terms may depend non-linearly on $\mathscr{T}$, or δ_k. In the interest of an uncluttered notation we will not further note this dependence explicitly. Equation (17) assumes that the running costs are quadratic in u; as in [14] the squared control costs can equivalently translated to a Gaussian prior over u that combines with the process noise Q_k to a an uncontrolled process with noise $Q_k + B_k H^{\text{-}1} B_k$.

We can now write the unnormalized posterior $\tilde{p}$ as the product of an uncontrolled process and a Gaussian likelihood,

$$\tilde{p}(z_{1:K}|\mathscr{T}) = \underbrace{\mathscr{N}(z_{1:K}|\boldsymbol{\mu}, \Sigma)}_{\text{dynamics prior}} \cdot \underbrace{\mathscr{N}[z_{1:K}|c, C]}_{\text{cost likelihood}}$$

where $\mathscr{N}[x|a, A] \propto \exp\{-\frac{1}{2}x^\top Ax + x^\top a\}$ is a Gaussian in canonical form, with precision matrix A and mean $A^{\text{-}1}a$, $c = (c_1, \dots, c_K)^\top$ is as in (17) neglecting the $u_k^\top H u_k$ terms, and $C = \text{diag}(C_1, \dots, C_K)$. The elements of $\boldsymbol{\mu}$ are given by

$$\boldsymbol{\mu}_i = (A_0 \cdots A_{i-1})\, z_0 + \sum\nolimits_{k=1}^{i-1} (A_{k+1} \cdots A_{i-1})\, a_k$$

and Σ is the symmetric matrix with

$$\Sigma_{ij} = \Sigma_{ji}^\top = (A_{j-1} \cdots A_i)\ \sum\nolimits_{k=0}^{i-1} (A_{i-1} \cdots A_k)(Q_k + B_k H^{\text{-}1} B_k)(A_{i-1}^\top \cdots A_k)^\top$$

for $i \leq j$. In practise, given a local linearization the unnormalized posterior $\tilde{p}(z_{1:K}|\mathscr{T})$ can be computed with same computational complexity as a Riccati or Kalman filter iterating over k [14].

Now let us define $\hat{\mathbf{z}}$ to be the subset of $z_{1:K}$ which have an associated intermediate cost, i.e., $\hat{\mathbf{z}} = \{z_k :\ [k\%C = 0] = \{z_k :\ c_k \neq 0, C_k \neq 0\}\}$. (Note that, if we subsumed the control costs $u_k^\top H u_k$ in the uncontrolled process, only at $[k\%K = 0]$ we have cost terms.) As we can marginalize the uncontrolled process for all $z_k \notin \hat{\mathbf{z}}$, we can retrieve $\tilde{p}(\hat{\mathbf{z}}|\mathscr{T})$ as

$$\tilde{p}(\hat{\mathbf{z}}|\mathscr{T}) = \mathscr{N}(\hat{\boldsymbol{\mu}}|\hat{\mathbf{C}}^{-1}\hat{\boldsymbol{c}}, \hat{\Sigma} + \hat{\mathbf{C}}^{-1})$$

where $\hat{\boldsymbol{\mu}}$ and $\hat{\Sigma}$ denote the appropriate sub-vector and -matrix of μ and Σ respectively. Hence, with $\boldsymbol{m} := \hat{\mathbf{C}}^{-1}\hat{\boldsymbol{c}}$ and $\mathbf{M} := \hat{\Sigma} + \hat{\mathbf{C}}^{-1}$, the approximate derivatives take the general form

$$\nabla \int_{z_{1:K}} \tilde{p}(z_{1:K}|\mathscr{T}) = \mathscr{N}(\hat{\boldsymbol{\mu}}|\boldsymbol{m}, \mathbf{M}) \Big[\boldsymbol{g}^\top[\nabla(\boldsymbol{m} - \hat{\boldsymbol{\mu}})] \ \ -\frac{1}{2}\,\text{Tr}\left(\mathbf{M}^{-1}\nabla\mathbf{M}\right) + \frac{1}{2}\boldsymbol{g}^\top[\nabla\mathbf{M}]\boldsymbol{g}\Big]$$

where $\boldsymbol{g} = \mathbf{M}^{-1}(\hat{\boldsymbol{\mu}} - \boldsymbol{m})$.

Combining the results, the overall approximation to the derivatives is obtained as

$$\begin{aligned}\nabla_{\delta_k}\mathscr{L}(\mathscr{T}) \approx -\nabla_{\delta_k} C_{\mathscr{T}}(\mathscr{T}) + \big[&\boldsymbol{g}^\top[\nabla_{\delta_k}(\boldsymbol{m} - \hat{\boldsymbol{\mu}})] \\ &-\frac{1}{2}\,\text{Tr}\left(\mathbf{M}^{-1}\nabla_{\delta_k}\mathbf{M}\right) + \frac{1}{2}\boldsymbol{g}^\top[\nabla_{\delta_k}\mathbf{M}]\boldsymbol{g}\Big] \ .\end{aligned} \tag{18}$$

The gradient $\nabla_{\delta_k}\mathbf{M}$ and $\nabla_{\delta_k}(\boldsymbol{m} - \hat{\boldsymbol{\mu}})$ are straight-forward by their definition. With this we can use any gradient based scheme to obtain a new $\mathscr{T}^{\text{new}}$, which in turn gives rise to a new approximation.

3.3 Expectation Maximisation

The solution to (13) can alternatively be obtained using an Expectation Maximisation approach. Specifically, we form the bound

$$\mathscr{L}(\mathscr{T}) > \int_{z_{1:K}} \underbrace{P(z_{1:K}|r_{1:K} = 1, \mathscr{T})}_{p(z_{1:K})} \log P(r_{1:K} = 1, z_{1:K}|\mathscr{T})$$

which is alternately maximised with respect to p and $\mathscr{T}$, in an E- and M-step.

In the E-Step we aim to calculate the posterior over the unobserved variables, i.e. the trajectories, given the current parameter values δ_k,

$$p(z_{1:K}) = P(z_{1:K}|r_{1:K} = 1, \mathscr{T}) \; .$$

We approximate this with $\tilde{p}$ using AICO as before. In the M-Step, we solve

$$\mathscr{T}^{\text{new}} = \underset{\mathscr{T}}{\text{argmin}} \underbrace{\big\langle \log P(r_{1:K} = 1, z_{1:K}|\mathscr{T})\big\rangle_{\tilde{p}}}_{:=\tilde{\mathscr{L}}(\mathscr{T})} \; ,$$

where $\tilde{p}$ is the approximation calculated in the E-Step based on $\mathscr{T}^{\text{old}}$. We may expand the objective as

$$\tilde{\mathscr{L}}(\mathscr{T}) = \sum\nolimits_{k=0}^{K-1} \left(\big\langle \log P(z_{k+t}|z_k, d_k)\big\rangle - \big\langle \tilde{c}_k(z_k, d_k)\big\rangle\right) + \mathscr{C},$$

where $\langle \cdot \rangle$ denotes the expectation with respect to $\tilde{q}$ and $\mathscr{C}$ is a constant. The required expectations, $\langle \tilde{c}_k(z_k, d_k)\rangle$ and

$$\big\langle \log P(z_{k+t}|z_k, d_k)\big\rangle = -\frac{D_z}{2} \log |Q_k| - \frac{1}{2}\big\langle (z_{k+t} - f(z_k))^\top Q_k^{-1} (z_{k+1} - f(z_k))\big\rangle \; ,$$

are in general not tractable. As previously, we therefore resort to a LQ approximation. This leads in the general case to an expression which can not be maximised analytically w.r.t. $\mathscr{T}$. However, if the approximation and discretization are chosen such that the system is also linear in δ, i.e.,

$$f(z_k) \approx (a_k + A_k z_k)\delta_k \; , \quad Q_k = Q\delta_k \; , \quad c_k(z_k, \delta_k) \approx \left(\frac{1}{2} z_k^\top C_k z_k - c_k^\top z_k\right)\delta_k$$

it can be shown that,

$$
\begin{aligned}
\frac{\partial}{\partial d_k}\tilde{\mathscr{L}}(\mathscr{T}) =& \delta_k^{-2} g_2 + \delta_k^{-1} g_1 + \left(g_0 + 2\,\frac{\mathrm{d}}{\mathrm{d}\delta} C_{\mathscr{T}}\Big|_{d_k} \right), \qquad (19)\\
g_1 =& -\frac{D_z^2}{2}\,, \quad g_2 = \frac{1}{2}\,\mathrm{Tr}\left(Q_k^{-1}(\langle z_{k+1} z_{k+1}^\top\rangle - 2\langle \mathbf{z_{k+1}} z_k^\top\rangle + \langle z_k z_k^\top\rangle)\right)\\
g_0 =& -\frac{1}{2}\Big[\mathrm{Tr}(A_k Q_k^{-1} A_k^\top \langle z_k z_k^\top\rangle) + a_k^\top Q_k^{-1} a_k\\
&+ 2 a_k^\top Q_k^{-1} A_k \langle x_k\rangle + \mathrm{Tr}(C_k \langle z_k z_k^\top\rangle) - 2 c_k^\top \langle z_k\rangle\Big].
\end{aligned}
$$

In the general case we may use an efficient gradient ascent to compute the M-step (for fixed $\tilde{p}$) and improve on δ_k's. However, in the specific case where $C_{\mathscr{T}}$ is a linear function of δ_k's, (19) is quadratic in δ_k^{-1} and the unique extremum under the constraint $\delta_k > 0$ can be found analytically.

3.4 Discussion

The two proposed methods have different merits. From the point of view of computational complexity the EM based updates are preferable as they only require computation of the pair marginals (z_k, z_{k+1}) and operate entirely on matrices which are the size of z_k's dimension. The gradient method instead requires computation of the covariance of all cost conditioned states and controls. Due to the inversion of this matrix, gradient updates are usually more expensive to compute.

While computationally attractive, EM updates suffer from numerical instability in many problems. In general, the deficiency of EM algorithms in near deterministic regimes is a well known problem, e.g., [1]. In our case it leads to instability when $Q \approx 0$ or if the posterior trajectories are severally constrained by the cost terms. The problem arises in the M-Step, which may be written as

$$
\underset{\mathscr{T}}{\operatorname{argmax}} - \mathrm{KL}\left(p(z_{1:K}|\mathscr{T}^{\mathrm{old}}) \| P(z_{1:K}|r_{1:K} = 1, \mathscr{T})\right) + \log \int_{z_{1:K}} P(r_{1:K} = 1, z_{1:K}|\mathscr{T})
$$

It is now apparent that for deterministic dynamics no change in δ_k is possible, lest the KL divergence becomes infinite.

4 Experiments

4.1 Evaluation on Basic Via-Point Tasks

We first evaluate the proposed method in simulation on a simple plant. As a basic plant, we used a simulation of a 2 degrees of freedom planar arm, consisting of two links of equal length. The state of the plant is given by $x = (\boldsymbol{q}, \dot{\boldsymbol{q}})$, with $\boldsymbol{q} \in \mathbb{R}^2$ the joint angles and $\dot{\boldsymbol{q}} \in \mathbb{R}^2$ associated angular velocities. The controls $\boldsymbol{u} \in \mathbb{R}^2$ are the joint space accelerations. We also added some noise with diagonal covariance.

For all experiments, we used a trajectory cost of the form

$$C(x_{1:K}, u_{1:K}, \mathscr{T}) = c(x_{1:K}) + \sum_{k=0}^{K} \delta_k \, u_k^\top C^u u_k + \alpha \delta_k(\mathscr{T}) \tag{20}$$

where $C^u = 10^4 \cdot \mathbf{I}$. Note that $\alpha \sum_{k=0}^{K} \delta_k$, where δ_k depends on $\mathscr{T}$, penalizes the total movement duration linearly. The state dependent cost was

$$c(x_{1:K}) = \sum_{i=1}^{f} (\phi_n(x_{\hat{k}_i}) - \boldsymbol{y}_i^*)^\top \Lambda_i (\phi_n(x_{\hat{k}_i}) - \boldsymbol{y}_i^*) \ , \tag{21}$$

where the tuplets $(\hat{k}_i, \phi_i, \Lambda_i, \boldsymbol{y}_i^*)$, consisting of a time step, a task space mapping, a diagonal weight matrix and the desired state in task space, define goals. For example, for point targets, the task space mapping is $\phi(x) = (x, y, \dot{x}, \dot{y})^\top$, i.e., the map from x to the vector of end point positions and velocities in task space coordinates, and $\boldsymbol{y}^*$ is the target coordinate.

4.1.1 Variable Distance Reaching Task

In order to evaluate the behaviour of AICOT we applied it to a reaching task with varying start-target distance. Specifically, for a fixed start point we considered a series of targets lying equally spaced along a line in task space. It should be noted that although the targets are equally spaced in task space and results are shown with respect to movement distance in task space, the distances in joint space scale non-linearly. The state cost (21) contained a single term incurred at the final discrete step with $\Lambda_f = 10^6 \cdot \mathbf{I}$. Figure 2c, d shows the movement duration ($= \sum_{k=0}^{K} \delta_k$) and standard reaching cost[1] for different temporal-cost parameters α (we used $\alpha_0 = 2 \cdot 10^7$), demonstrating that AICOT successfully trades-off the movement duration and standard reaching cost for varying movement distances. In Fig. 2b, we compare the reaching costs of AICOT with those obtained with a fixed duration approach, in this case AICO. Note that although with a fixed, long duration (e.g., AICO with duration T = 0.41) the control and error costs are reduced for short movements, these movements

[1] n.b. the *standard reaching cost* is the sum of control costs and cost on the endpoint error, without taking duration into account.

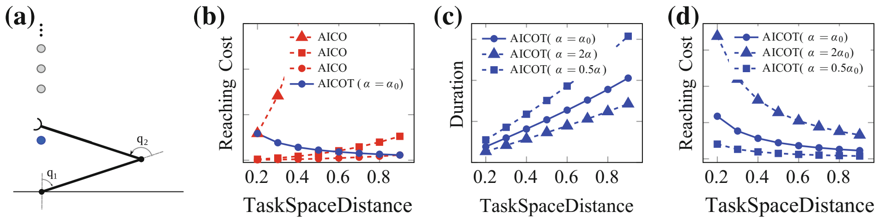


Fig. 2 Temporal scaling behaviour using AICOT. **a** Schematic of plant together with mean start position ○ and list of targets ○ **b** Comparison of reaching costs (control + error cost) for AICOT and a fixed duration approach, i.e. AICO. **c** and **d** Effect of changing time-cost weight α, (effectively the ratio between reaching cost and duration cost) on duration and reaching cost (control + state cost)

necessarily have up to 4× longer durations than those obtained with AICOT. For example for a movement distance of 0.2 application of AICOT results in a optimised movement duration of 0.07 (cf. Fig. 2c), making the fixed time approach impractical when temporal costs are considered. Choosing a short duration on the other hand (AICO (T=0.07)) leads to significantly worse costs for long movements. We further emphasis that the fixed durations used in this comparison were chosen post hoc by exploiting the durations suggested by AICOT; in absence of this, there would have been no practical way of choosing them apart from experimentation. Furthermore, we would like to highlight that, although the results suggests a simple scaling of duration with movement distance, in cluttered environments and plants with more complex forward kinematics, an efficient decision on the movement duration cannot be based only on task space distance.

4.1.2 Via Point Reaching Task

We also evaluated the proposed algorithm in a more complex via point task. The task requires the end-effector to reach to a target, having passed at some point through a given second target, the via point. This task is of interest as it can be seen as an abstraction of a diverse range of complex sequential tasks that requires one to achieve a series of sub-tasks in order to reach a final goal. This task has also seen some interest in the literature on modelling of human movement using the optimal control framework [12]. Here the common approach is to choose the time point at which one passes the via point such as to divide the movement duration in the same ratio as the distances between the start point, via point and end target. This requires on the one hand prior knowledge of these movement distances and on the other, makes the implicit assumption that the two movements are in some sense independent.

Here, we demonstrate the ability of our approach to solve such sequential problems, adjusting movement durations between sub-goals in a principled manner, and show that it improves upon the standard modelling approach. Specifically, we apply AICOT to the two via point problems illustrated in Fig. 3a with randomised start

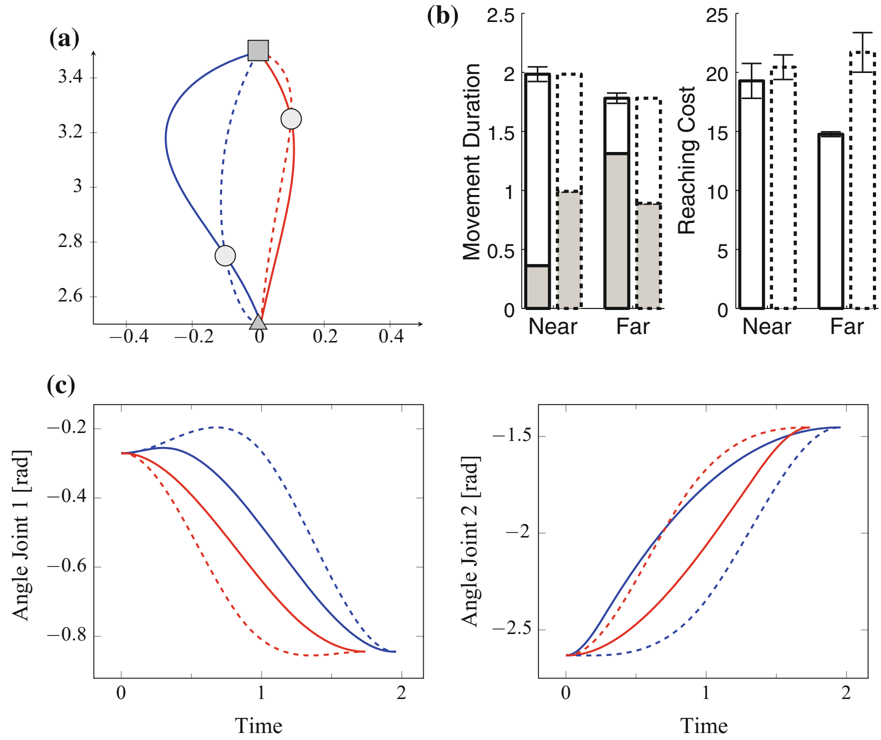


Fig. 3 Comparision of AICOT (——) to AICO with the common modelling approach (- -) with fixed times on a via point task. **a** End point task space trajectories for two different via points ○ obtained for a fixed start point △. **c** The corresponding joint space trajectories. **b** Movement durations and reaching costs (control + error costs) from 10 random start points. The proportion of the movement duration spend before the via point is shown in light gray (mean in the AICOT case)

states.[2] For comparison, we follow the standard modelling approach and apply AICO to compute the controller. We again choose the movement duration for the standard case post hoc to coincide with the mean movement duration obtained with AICOT for each of the individual via point tasks. Each task is expressed using a cost function consisting of two point target cost terms. Specifically, (21) takes the form

$$c(x_{1:K}) = (\phi(x_{\frac{K}{2}}) - \boldsymbol{y}_v^*)^\top \Lambda_v(\phi(x_{\frac{K}{2}}) - \boldsymbol{y}_v^*) + (\phi(x_K) - \boldsymbol{y}_e^*)^\top \Lambda_e(\phi(x_K) - \boldsymbol{y}_e^*) ,$$

[2] For the sake of clarity, Fig. 3a, c show mean trajectories of controllers computed for the mean start state.

with diagonal matrices

$$\begin{aligned}\Lambda_v &= \mathrm{diag}(\lambda_{pos}, \lambda_{pos}, 0, 0)\\ \Lambda_e &= \mathrm{diag}(\lambda_{pos}, \lambda_{pos}, \lambda_{vel}, \lambda_{vel}) \ ,\end{aligned}$$

where $\lambda_{pos} = 10^5$ & $\lambda_{vel} = 10^7$ and vectors $\boldsymbol{y}_v^* = (\cdot, \cdot, 0, 0)^\top$, $\boldsymbol{y}_e^* = (\cdot, \cdot, 0, 0)^\top$ desired states for individual via point and target, respectively. Note that the cost function does not penalise velocity at the via point but encourages the stopping at the target. While admittedly the choice of incurring the via point cost at the middle of the movement ($\frac{K}{2}$) is likely to be a sub-optimal choice for the standard approach, one has to consider that in more complex task spaces, the relative ratio of movement distances may not be easily accessible and one may have to resort to the most intuitive choice for the uninformed case as we have done here. Note that although for AICOT this cost is incurred at the same discrete step, we allow δ_k before and after the via point to differ, but constrain them to be constant throughout each part of the movement, hence, allowing the cost to be incurred at an arbitrary point in real time. We sampled the initial position of each joint independently from a Gaussian distribution with a variance of 3°. In Fig. 3a, c, we show mean trajectories in task space and joint space for controllers computed for the mean initial state. Interestingly, although the end point trajectory for the *near* via point produced by AICOT may look sub-optimal than that produced by the standard AICO algorithm, closer examination of the joint space trajectories reveal that our approach results in more efficient actuation trajectories. In Fig. 3b, we illustrate the resulting average movement durations and costs of the mean trajectories. As can be seen, AICOT results in the expected passing times for the two via points, i.e., early vs. late in the movement for the near and far via point, respectively. This directly leads to a lower incurred cost compared to un-optimised movement durations.

4.1.3 Sequential and Joint Planning

In order to highlight the shortcomings of sequential time optimal control, we compare planning a complete movement, referred to as joint optimisation, to planning a sequence of individually optimised movements. We again use the via-point task of the previous section and performed (i) planning using AICOT on the entire task (ii) using AICOT to plan for to reaching tasks – start point to via-point and via-point to final target – by splitting the cost function. In the latter the end state of the first reaching movement, rather then the via-point, was used as initial state for the second sub-task. Figure 4 summarises the results. As can be seen in Fig. 4a the two approaches lead to solutions with substantially different end-effector trajectories in task space. The joint optimisation, accounting for the need to continue to the eventual target after the via-point, yields a different approach angle. The profound effect this has on the incurred cost can be seen in Fig. 4b. While the joint planning incurs higher cost before the via-point the overall cost is more than halved. Importantly, as

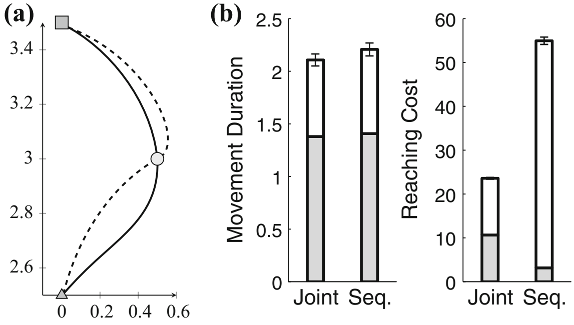


Fig. 4 Joint (——) vs. sequential (- -) optimisation using our approach on a via-point task as described in the main text. **a** Task space trajectories for the fixed start point △. Via-point and target are indicated by ○ and ■, respectively. **b** The movement durations and reaching costs for 10 random start points. The mean proportion of the movement duration spend before the via point is shown in light grey

Table 1 Results for application of AICOT to the robotic manipulation with obstacles in the reaching tasks illustrated in Fig. 5. Shown are the mean ratio of expected cost relative to AICO and it's standard deviation

Method	Simple obstacles	Complex obstacle
AICO	1	1
AICOT (end cost)	0.585 (± 0.337)	0.635 (± 0.085)
AICOT (full)	0.549 (± 0.311)	0.123 (± 0.047)

the plot of the movement durations illustrates, this reduction in cost is not achieved by an increase in movement duration, with both approaches leading to not significantly different durations. However, one should note that this effect would be less pronounced if the cost required stopping at the via-point, as it is the velocity away from the end target which is the main problem for the sequential planner.

4.2 7-DOF Robotic Manipulation Tasks

We now turn to evaluating the method for planning with the 7-DOF Kuka lightweight robot. Our aim is two fold, on the one hand to demonstrate scalability to practical applications, and on the other hand, to demonstrate that in practical tasks temporal optimisation can significantly improve the results compared to naive selection of the movement durations.

The state of the plant is given by $x = (\boldsymbol{q}, \dot{\boldsymbol{q}})$, with $\boldsymbol{q} \in \mathbb{R}^7$ the joint angles and $\dot{\boldsymbol{q}} \in \mathbb{R}^7$ the associated angular velocities. The controls $\boldsymbol{u} \in \mathbb{R}^7$ are the joint space accelerations. We also added some i.i.d. noise with diagonal covariance. The trajectory cost takes the general form

$$C(x_{1:K}, u_{1:K}, \mathscr{T}) = \sum\nolimits_{k}^{K} \Big(\sum\nolimits_{m=1}^{M} \|\phi_m(x_k) - \boldsymbol{y}_m^*\|^2_{\Lambda_{m,k}} + u_k^\top \delta_k C^u u_k \Big) \tag{22}$$

where the tuplets $(\phi_m, \Lambda_{m,k}, \boldsymbol{y}_m^*)$ define the task variables, consisting of a task space mapping, a time varying diagonal weight matrix and the desired state in task space.

In each task we compare three methods:

- **AICOT(full)** is the complete algorithm as described in Sect. 3.2.
- **AICOT(end cost)** is the algorithm as described in Sect. 3.2. However, the gradient is calculated taking only the reaching cost into account, i.e., ignoring joint limit and obstacle costs. The intention is to illustrate that selection of duration needs to take into account the entire problem and can not be simply based on a target-distance law as could be derived from, e.g., Fig. 2.
- **AICO** is the algorithm with fixed duration. This is to provide a comparison to the naive approach prevalent in the literature. Note however that, we set the duration the mean duration obtained by *AICOT(end cost)*. Hence it was in some sense adapted to the task distribution. Without AICOT, selection would have, at best, relied on manual selection based on an individual task instance or, at worst, a random guess. Both approaches lead to substantially worse results.

4.2.1 Simple Obstacle Reaching Task

We first consider a standard reaching task with obstacles. The task is defined via the following set of task variables

- **Reaching**: with $\phi_1(x) \in \mathbb{R}^6$ the arm's end effector position and velocity. The cost is incurred in the final time step only, i.e., $\Lambda_{1,k\neq K} = 0$, and $\boldsymbol{y}^*$ indicates the desired state end-effector positions with zero velocities.
- **Joint Limits**: with $\phi_2(x) \in \mathbb{R}$ a scalar indicating danger of violating joint limits. Specifically,

$$\phi_2(x) = \sum\nolimits_{j} \mathscr{H}(d_j - \varepsilon)^2 \,, \tag{23}$$

  with d_j the distance to the joint limit between of joint j, $\mathscr{H}$ the heavy-side function and margin $\varepsilon = 0.1$rad. This task variable is considered throughout the trajectory, i.e. $\Lambda_{2,1} = \Lambda_{2,1} = \cdots = \Lambda_{2,K}$.
- **Collisions**: with $\phi_2(x) \in \mathbb{R}$ a scalar indicating proximity of obstacles. Specifically ϕ_2 takes the general form (23) with d_j the shortest distance between a pair j of collidable objects, i.e. the set of links of the arm and obstacles, and margin $\varepsilon = 0.02$ m. Like the joint limits, this task variable is also considered throughout the trajectory.

Although the resulting finite cost functions can not guarantee that collisions with obstacles or joint limits will not occur, such approximations are typical in the literature (e.g., [4, 14]) and lead, under appropriate weighting between the reaching and collision components, to good results with low collision probability.

We consider a randomised task with two spherical obstacles, an example configuration being illustrated in Fig. 5a. Specifically, both the target and obstacle positions are randomly sampled, the latter so that they lie near the direct path to the target so as to influence the solution. The results are summarised in Table 1. As different task instances can give rise to very different expected costs, we compare expected costs relative to AICO, i.e., the improvement of the methods over the baseline without temporal optimisation. The expected costs are estimated from sampled trajectories and we consider 50 task instances. As can be seen, temporal optimisation improves upon the naive application of AICO. In particular note that, instance specific durations as given by AICOT(end cost) improve significantly on selecting an informed constant duration (the mean duration over task instances). Furthermore, taking the entire problem into account leads to increasing gains as the problem complexity increases.

In general we note that a possible straightforward extension of the gradient based algorithm whereby we solve the problem incrementally, by using the solution of a reduced problem with intermediate cost terms removed, i.e., the AICO-T (end cost) approach, as an initialization of AICO-T (full) can significantly improve the computational complexity of the gradient based method for problems with many intermediate costs terms.

4.2.2 Complex Obstacle

We now consider a generic instance of a task involving manipulation in constrained spaces. It comprises the same basic task variables as used with the simple obstacle above. However instead of using spherical obstacles we use a wall with two holes as illustrated in Fig. 5b. The end-effector starts reaching through one of the holes and the reaching target lies in the other hole. Due to their local nature direct application of AICO fails in this task, as do alternative local solvers like, e.g., iLQG. However, in the context of AICO [16] suggested using parallel inference in the normal state space and a abstract topological representations to overcome limitations of local planning in such tasks. With a suitable topological representation the task becomes nearly linear in the alternative representation, which then serves to regularise further inference in the plant's state space. Here we use the interaction mesh representation suggested by [16], a scale and position invariant representation of relative positions of the plant and markers in the environment. This representation has been used for this task by [4] who also used AICO. For this experiment we again sampled the position of the wall relative to the manipulator and compared the relative expected costs averaged over 50 task instances. The results are shown in the second column of Table 1.

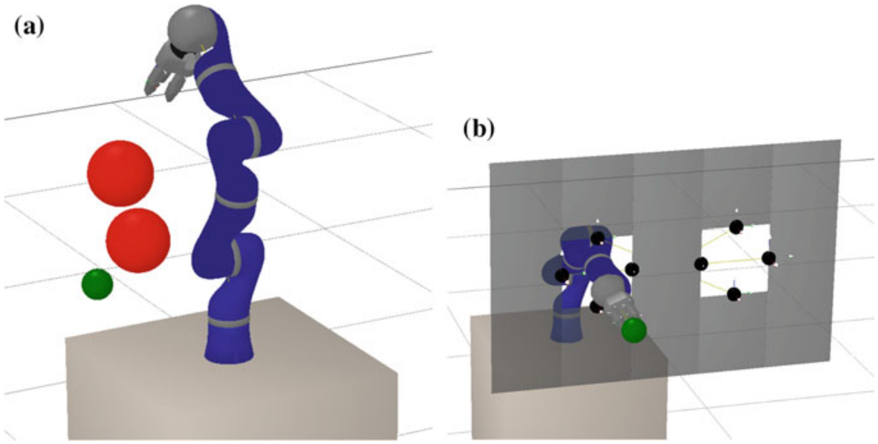


Fig. 5 Example configurations for the tasks used with KUKA 7-DOF robotic system in simulation. **a** The simple obstacle task. The manipulator has to reach with it's end-effector to the target 🟢 whilst avoiding the obstacles 🔴. The task is randomised by sampling both the target and the obstacle positions. **b** The complex obstacle task. The manipulator starts in one hole and has to reach for the target 🟢 in the other, whilst avoiding collisions with the wall. The position of the wall is randomised

5 Conclusion

The contribution of this paper is a novel method for jointly optimizing a trajectory and its time evolution (temporal scale and duration) in the stochastic optimal control framework. In particular, two extension of the AICO method of [14] with complementary strength and weaknesses are presented. The gradient based approach, on the one hand, is widely applicable but can become computationally demanding. Meanwhile, the EM method provides an algorithm with lower computational cost, is however only applicable for certain classes of problems.

The experiments have concentrated on demonstrating the benefit of temporal optimisation in manipulation tasks. However, arguably it is dynamic movements which can benefit most from temporal adjustment. An example of this was seen in the brachiation task of [7], where our framework was applied to brachiation with variable stiffness actuation, showing that an coordinated interplay of stiffness and temporal adjustment gives rise to gains in performance. We anticipate that, with the general rise of interest in variable impedance, e.g., in throwing [2], locomotion [3] or climbing robots [6], temporal optimisation will become a necessity if the capabilities of the dynamical system are to be fully exploited. Our framework provides a principled step in this direction.

Acknowledgements This work was supported by the EU FP7 project TOMSY (IST-FP7-270436).

References

1. Barber, D., Furmston, T.: Solving deterministic policy (PO)MDPs using EM and antifreeze. In: Proceedings of the 1st International Workshop on Learning and data Mining for Robots (2009)
2. Braun, D., Howard, M., Vijayakumar, S.: Optimal variable stiffness control: formulation and application to explosive movement tasks. Auton. Robot. **33**(3), 237–253 (2012)
3. Enoch, A., Sutas, A., Nakaoka, S., Vijayakumar, S.: BLUE: A bipedal robot with variable stiffness and damping. In: Proceedings of IEEE/RAS International Conference on Humanoid Robots (2012)
4. Ivan, V., Zarubin, D., Toussaint, M., Vijayakumar, S.: Topology-based representations for motion planning and generalisation in dynamic environments with interactions. International Journal of Robotics Research (2013, in press)
5. Kulchenko, P., Todorov, E.: First-exit model predictive control of fast discontinuous dynamics: application to ball bouncing. In: Proceedings of the IEEE International Conference on Robotics and Automation (2011)
6. Long, A., Murphey, T.D., Lynch, K.: Optimal motion planning for a class of hybrid dynamical systems with impacts. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4220–4226 (2011)
7. Nakanishi, J., Vijayakumar, S.: Exploiting passive dynamics with variable stiffness actuation in robot brachiation. In: Robotics: Science and Systems VIII (2012)
8. Rawlik, K.: Approximate inference approaches to stochastic optimal control. Ph.D. thesis, University of Edinburgh (2013)
9. Rawlik, K., Toussaint, M., Vijayakumar, S.: On Stochastic Optimal Control and Reinforcement Learning by approximate inference. In: Proceedings Robotics: Science and Systems VIII (2012)
10. Stengel, R.F.: Optimal Control and Estimation (Dover Books on Advanced Mathematics). Dover Publications, New York (1986)
11. Theodorou, E., Tassa, Y., Todorov, E.: Stochastic differential dynamic programming. In: Proceedings of the American Control Conference, pp. 1125–1132. IEEE (2010)
12. Todorov, E., Jordan, M.: Optimal feedback control as a theory of motor coordination. Nat. Neurosci. **5**(11), 1226–1235 (2002)
13. Todorov, E., Li, W.: A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proceedings of the American Control Conference, pp. 300–306 (2005)
14. Toussaint, M.: Robot trajectory optimization using approximate inference. In: Proceedings of the 26th International Conference on Machine Learning, pp. 1049–1056. ACM (2009)
15. Toussaint, M., Storkey, A.: Probabilistic inference for solving discrete and continuous state Markov Decision Processes. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 945–952 (2006)
16. Zarubin, D., Ivan, V., Toussaint, M., Komura, T., Vijayakumar, S.: Hierarchical motion planning in topological representations. In: Proceedings of Robotics: Science and Systems VIII (2012)

Part II
Humanoids and Legged Locomotion

Session Summary

The advancement of walking skills is a fundamental perquisite for legged robots to permit the utility of these machines within real world workspaces. This entails substantial progress and a radical change in the locomotion control paradigm from the slowly adaptive pre-planned bipedal gait generators and balancing stabilizers towards more rapidly modulated generators combined with reflexive behaviours that will allow humanoids and legged robots in general to cope with uneven terrains and ground uncertainties and rapidly modulate their gait to adapt. Balancing as part for the general locomotion control problem cannot be ensured in all cases through only gait regulation actions, and lately, the research community have moved towards more human-like whole body balancing skills exploiting the potential of whole-body motions which can be harmonized for the purpose of compensating specific motion dynamics, reducing the overall locomotion effort and generating effective balancing reactions and recovery abilities under unpredicted moderate to strong disturbances. The implementation requirements of all these techniques have recently paved the development of several new tools in the areas of state estimation, whole body motion optimization, locomotion and balancing modelling. In this session, eight contributions were presented that dealt with the development of innovative models and controllers for legged robot locomotion and balancing.

The first talk by Roy Featherstone introduced a new model of the dynamics of balancing in the plane, which makes use of only two parameters of the robots balancing behaviour, both simple functions of basic physical properties of the robot mechanism. A third parameter describes the effect of other movements on the robots balance. Based on this this model, a high-performance balance controller was then presented as a simple four-term control law with gains that are trivial functions of the two model parameters and a single value chosen by the user that determines the overall speed of balancing. The model and the balance controller were first applied to a double pendulum, and then extended to a general planar mechanism. Simulation results were presented showing the controllers performance at following commanded motion trajectories while simultaneously maintaining the robots balance. In the second talk of the session, Michele Focchi proposed a methodology for foot slip detection in legged robots and estimation of the friction parameters using only proprioceptive sensors. Indeed, the majority of locomotion controllers and state estimation algorithms rely on the assumption that the stance feet are not slipping. The capability to detect foot slippage at the very beginning and promptly recover the traction is crucial for the body stability of legged robots and can assist to avoid falling. Having detected the foot slip, a recovery strategy, which exploits the capabilities of a whole body controller, implemented for locomotion, was used to optimize the ground reaction forces (GRFs). The proposed method is general and can be applied to any legged robot. In this talk, the application of the method to a quadruped robot was presented demonstrating its effectiveness while the robot was walking on challenging terrains. Nicholas Perrin presentation concerned with the development of novel approaches to solve for 3D locomotion with multiple non-coplanar contacts. Going further than the classical Zero Moment Point-based method, two new techniques were presented. Both formulations are based on model predictive control (MPC) to generate dynamically balanced trajectories with no restrictions on the trajectory of the centre of mass. The first formulation treats the balance criterion as an objective function and solves the control problem using a sequence of alternating convex quadratic programs. The second formulation considers the criterion as a constraint and solves a succession of convex quadratic ally constrained quadratic programs (QCQPs). The main feature of the proposed MPC schemes is that they can be efficiently solved through a succession of convex optimization problems, when the problem formulation is of particular forms, such as bilinear problems or non-convex QCQPs. In his talk, Jean-Paul Laumond briefly discussed The Yoyo-Man, a research action that investigates the synergies of anthropomorphic locomotion. A seminal hypothesis is made in which the wheel is considered as a plausible model of bipedal walking. The presentation reported on preliminary results developed along three perspectives combining biomechanics, neurophysiology and robotics. Firstly, from a motion capture data basis of human walkers, the center of mass (CoM) is identified as a geometric center from which the motions of the feet are organized. It was then demonstrated how rimless wheels that model most passive walkers are better controlled when equipped with a stabilized mass on top of them. CoM and head play complementary roles that define what is called the Yoyo-Man. The next talk by Katie Byl discussed on the evaluation of robustness of bipedal locomotion on variable-height terrains. The work considers a point-foot biped on a variable-height terrain and measure robustness by the expected number of steps before failure. The proposed method uses quantification of robustness to benchmark and optimize a given (low-level) controller. Two particular control strategies as case demonstrations were studied. One scheme is the now-familiar hybrid zero dynamics approach and the other is a method using piece-wise reference trajectories with a sliding mode control. The presentation provided a methodology for optimization of a broad variety of parameterizable gait control strategies and illustrates dramatic increases in robustness due to both gait optimization and choice of control strategy. Guilherme Maeda talk dealt with an interaction learning method suited for semi-autonomous robots that work with or assist a human partner. The method aims at generating a collaborative trajectory of the robot as a function of the current action of the human. The trajectory generation is based on action recognition and prediction of the human movement given intermittent observations of his/her positions under unknown speeds of execution. The problem typically arises from motion capture systems in scenarios that lead to marker occlusion. The ability to predict the human movement while observing the initial part of his/her trajectory allows for faster robot reactions and eliminates the need of time alignment of the training data. The method models the coupling between human–robot movement primitives, and it is scalable in relation to the number of tasks. The method is evaluated using a 7-DoF lightweight robot arm equipped with a 5-finger hand in a multitask collaborative assembly experiment, also comparing results with a previous method based on time aligned trajectories. Avik De in his talk discussed a notion of parallel composition to achieve for the first time a stability proof and an empirical demonstration of a steady-state gait on a highly coupled 3DOF legged platform controlled by two simple (decoupled) feedback laws that provably stabilize in isolation of two simple 1DOF mechanical subsystems. A limit cycle was stabilized on a tailed monoped to excite sustained sagittal plane translational hopping energized by tail-pumping during stance. The constituent subsystems for which the controllers are nominally designed were a purely vertical bouncing mass (controlled by injecting energy into its springy shaft) and a purely tangential rimless wheel (controlled by adjusting the inter-spoke stepping angle). The presentation described the use of averaging methods in legged locomotion to prove that this parallel composition of independent 1DOF controllers achieves an asymptotically stable closed-loop hybrid limit cycle for a dynamical system that approximates the 3DOF stance mechanics of a physical tailed monoped. In the last talk of the session, Steve Tonneau discussed the challenge of multiped locomotion in cluttered environments as a problem of planning acyclic sequences of contacts that characterize the motion. To overcome the inherent combinatorial difficulty, the work proposed to divide the problem in two sub-problems: first, planning a guide trajectory for the root of the robot, and then, generating relevant contacts along this trajectory were considered. The presentation introduces theoretical contributions to these two sub-problems. A theoretical characterization of the guide trajectory, named true feasibility, which guarantees that a guide can be mapped into the contact manifold of the robot was introduced. As opposed to previous approaches, this property makes it possible to assert the relevance of a guide trajectory without explicitly computing contact configurations. Guide trajectories are then easily mapped into a valid sequence of contacts, and a particular sequence with desirable properties, such as robustness, efficiency and naturalness, is considered. Based on this, a complete acyclic contact planner was then introduced and its performance was demonstrated by producing a large variety of motions.

A New Simple Model of Balancing in the Plane

Roy Featherstone

1 Introduction

This paper considers the problem of a planar robot that is actively balancing on a single point of support while simultaneously executing motion commands. In particular, the same motion freedom that is used for balancing is also subject to motion commands. The robot is therefore overloaded in the sense that the number of task variables to be controlled exceeds the number of actuator variables. Such overloading is physically possible, and is routinely exhibited by circus performers and the like, as well as by inverted pendulum robots [8] and wheeled robots that use the same motion freedom both for balancing and for transport [4, 10].

The main contribution of this paper is a new model of the plant (i.e., the robot mechanism) in which the essential features of the robot's balancing behaviour have been reduced to just two numbers. A third number summarizes the disturbance caused by other movements being performed by the robot. The model is obtained by exploiting a property of joint-space momentum variables. The advantages of this model are: (1) it is exceptionally simple; (2) it applies to general planar robots, including robots with kinematic loops; (3) it takes into account the effect of other movements of the robot (i.e., movements for accomplishing tasks other than balancing); (4) the model parameters have a clear physical meaning that is easy to understand; (5) they can be computed efficiently using standard dynamics algorithms; and (6) a high-performance balance controller is easily obtained by a simple feedback control law acting directly on the new plant model.

A second contribution is the new balance controller derived from the plant model. It resembles the one presented in [1, 3], and shares its robustness to effects such as torque limits, modelling errors and slippage at the point of support. However, it is simpler, and it can easily be applied to a general planar robot. It differs from the

R. Featherstone (✉)
Department of Advanced Robotics, Istituto Italiano di Tecnologia,
via Morego 30, 16163 Genova, Italy
e-mail: roy.featherstone@iit.it

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_10

typical approach to balance control in the literature, as exemplified by [7, 9, 13], in that it is a four-term controller using full state feedback, rather than a three-term output-zeroing controller with a one-dimensional zero dynamics. Note that the great majority of literature in this area is actually on swing-up control (e.g. [11, 12]) which is not considered here. The paper concludes with some simulation results showing the performance of the new controller at balancing an inverted triple pendulum while simultaneously following a variety of motion commands.

2 The New Model

A fundamental aspect of balancing is that the controller must control more state variables than the available controls. To understand how this can be done, consider the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u)$, in which $\boldsymbol{x}$ is a vector of state variables and u is a control input. (Bold letters denote vectors.) If $\boldsymbol{x}$ has the property that $x_{i+1} = \dot{x}_i$ for every i, then any control policy that successfully controls x_1 has the side-effect of controlling all of the other elements of $\boldsymbol{x}$. Furthermore, the condition $x_{i+1} = \dot{x}_i$ is sufficient but not necessary, and can be relaxed to some extent. Balancing is an activity that can be accomplished in this way; and the new model described here is essentially a good choice of $\boldsymbol{x}$, having a simple function f, which allows balancing to be achieved using a simple control law for u.

Figure 1 shows a planar 2R mechanism representing an inverted double pendulum. Joint 1 is passive and represents the point contact between the foot of the mechanism and a supporting surface (the ground). It is assumed that the foot neither slips nor loses contact with the ground. The state variables of this robot are q_1, q_2, $\dot{q}_1$ and $\dot{q}_2$. The total mass of the robot is m; the coordinates of its centre of mass (CoM) relative to the support point are c_x and c_y; and it is assumed that the support point is

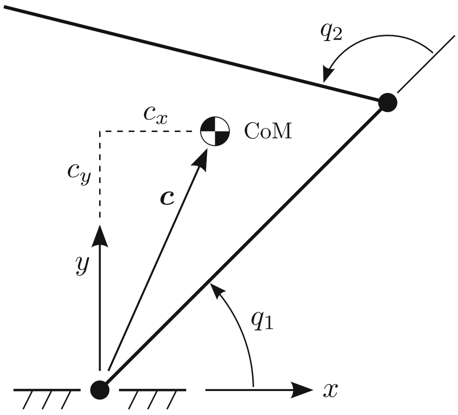


Fig. 1 Planar 2R robot mechanism representing an inverted double pendulum actuated at joint 2

stationary, i.e., it is not a rolling contact. The equation of motion of the robot is

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_2 \end{bmatrix}, \tag{1}$$

where H_{ij} are elements of the joint-space inertia matrix, C_i are elements of the bias vector containing Coriolis, centrifugal and gravitational terms, $\ddot{q}_i$ are the joint accelerations, and τ_2 is the torque at joint 2. The conditions for the robot to be in a balanced position are: $c_x = 0$, $\dot{q}_1 = 0$ and $\dot{q}_2 = 0$. The robot is also subject to the position command $q_2 = q_\mathrm{c}$, where q_c is an input to the controller.

Any mechanism that balances on a single point has the following special property, which is central to the activity of balancing: the only force that can exert a moment about the support point is gravity. If we define L to be the total angular momentum of the robot about the support point then we find that

$$\dot{L} = -mgc_x\,, \tag{2}$$

where g is the magnitude of gravitational acceleration (a positive number). This equation implies

$$\ddot{L} = -mg\dot{c}_x \tag{3}$$

and

$$\dddot{L} = -mg\ddot{c}_x\,. \tag{4}$$

We also have

$$L = p_1 = H_{11}\dot{q}_1 + H_{12}\dot{q}_2\,, \tag{5}$$

which follows from a special property of joint-space momentum that is proved in the appendix: if p_i is the momentum variable of joint i then, by definition, $p_i = \sum_j H_{ij}\dot{q}_j$; but if the mechanism is a kinematic tree then p_i is also the component in the direction of motion of joint i of the total momentum of the subtree beginning at body i. As the whole robot rotates about joint 1, it follows that p_1 is the total angular momentum of the robot about the support point, hence $p_1 = L$.

Observe that $\dot{L}$ is simply a constant multiple of c_x, and that L and $\ddot{L}$ are both linear functions of the robot's velocity, implying that the condition $L = \ddot{L} = 0$ is equivalent to $\dot{q}_1 = \dot{q}_2 = 0$. So the three conditions for balance can be written as

$$L = \dot{L} = \ddot{L} = 0\,. \tag{6}$$

Thus, any controller that successfully drives L to zero will cause the robot to balance, but will not necessarily bring q_2 to the commanded angle.

We now introduce a fictitious extra joint between joint 1 and the base, which is a prismatic joint acting in the x direction. To preserve the numbering of the existing joints, the extra joint is called joint 0. This joint never moves, and therefore never has any effect on the dynamics of the robot. Its purpose is to increase the number of coefficients in the equation of motion, which now reads

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ \tau_2 \end{bmatrix} . \tag{7}$$

The position and velocity variables of joint 0 are always zero, and τ_0 takes whatever value is necessary to ensure that $\ddot{q}_0 = 0$. The reason for adding this joint is that the special property of joint-space momentum, which we used earlier to deduce that $p_1 = L$, also implies that p_0 is the linear momentum of the whole robot in the x direction. So $p_0 = m\dot{c}_x$. With the extra coefficients in Eq. 7 we can write

$$p_0 = H_{01}\dot{q}_1 + H_{02}\dot{q}_2 = m\dot{c}_x = -\ddot{L}/g \,, \tag{8}$$

so that we now have a pair of linear equations relating L and $\ddot{L}$ to the two joint velocities:

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} . \tag{9}$$

Solving this equation for $\dot{q}_2$ gives

$$\dot{q}_2 = Y_1 L + Y_2 \ddot{L} \,, \tag{10}$$

where

$$Y_1 = \frac{H_{01}}{D} \,, \qquad Y_2 = \frac{H_{11}}{gD} \tag{11}$$

and

$$D = H_{12}H_{01} - H_{11}H_{02} \,. \tag{12}$$

Clearly, this only works if $D \neq 0$. The physical significance of $D = 0$ is explained below. From a control point of view, a problem also arises if $Y_1 = 0$, and this too is discussed below.

We now have all the component parts of the new plant model, which is shown in Fig. 2 in the form of a block diagram. The state variables are q_2, L, $\dot{L}$ and $\ddot{L}$, which

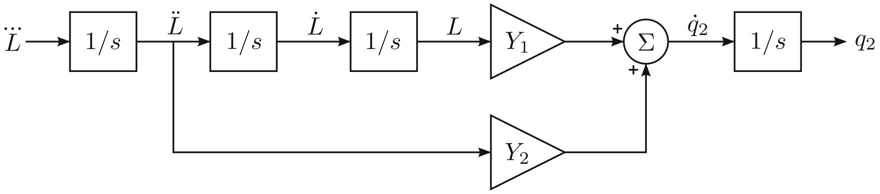


Fig. 2 New plant model for balancing

replace the original state variables. As will be shown in the next section, a simple feedback control law closed around this plant can make q_2 follow a commanded trajectory while maintaining the robot's balance. To be more accurate, what really happens is that the control law tips the robot slightly off balance so that the necessary balance recovery movement just happens to make q_2 follow the commanded trajectory. Once q_2 has reached its final position, the other state variables settle to zero, thereby satisfying the conditions for balance in Eq. 6.

Observe that the new plant model has only two parameters: the two gains Y_1 and Y_2. These gains are calculated directly from the elements of the joint-space inertia matrix in Eq. 7, which in turn can be calculated using any standard method for calculating the joint-space inertia matrix of a robot. Thus, no special code is needed to calculate the model parameters.

Physical Meaning of Y_1 and Y_2

The two gains Y_1 and Y_2 are related in a simple way to two physical properties of the mechanism: the natural time constant of toppling and the linear velocity gain [6]. The former quantifies the rate at which the robot begins to fall in the absence of movement of the actuated joint. The latter measures the degree to which motion of the actuated joint influences the motion of the CoM.

If there is no movement in the actuated joint then the robot behaves as if it were a single rigid body, and its motion is governed by the equation of motion of a simple pendulum:

$$I\ddot{\theta} = mgc(\cos(\theta_0) - \cos(\theta)) \tag{13}$$

where I is the rotational inertia of the robot about the support point, $c = |\boldsymbol{c}|$ is the distance between the CoM and the support point, $\theta = \tan^{-1}(c_y/c_x)$ is the angle of the CoM from the x axis, and the term $mgc\cos(\theta_0)$ is a hypothetical constant torque acting at the support point, which serves to make θ_0 an equilibrium point of the pendulum. Linearizing this equation about θ_0, and defining $\phi = \theta - \theta_0$, results in the following equation:

$$I\ddot{\phi} = mgc_y\phi\,, \tag{14}$$

which has solutions of the form

$$\phi = A\mathrm{e}^{t/T_\mathrm{c}} + B\mathrm{e}^{-t/T_\mathrm{c}} \tag{15}$$

where A and B are constants depending on the initial conditions, and T_c is the natural time constant of the pendulum, given by

$$T_\mathrm{c}^2 = \frac{I}{mgc_y}\,. \tag{16}$$

If $c_y > 0$ then T_c is real and Eq. 15 contains both a rising and a decaying exponential. This is characteristic of an unstable equilibrium. If $c_y < 0$ then T_c is imaginary and Eq. 15 is a combination of sines and cosines, which is characteristic of a stable equilibrium. But if $c_y = 0$ then we are at the boundary between stable and unstable equilibrium and T_c is unbounded. As we are considering the problem of a robot balancing on a supporting surface, it is reasonable to assume $c_y > 0$.

From the definition of the joint-space inertia matrix [5, Sect. 6.2] we have $H_{01} = \boldsymbol{s}_0^\mathrm{T} \boldsymbol{I}_0^\mathrm{c} \boldsymbol{s}_1$ and $H_{11} = \boldsymbol{s}_1^\mathrm{T} \boldsymbol{I}_1^\mathrm{c} \boldsymbol{s}_1$, where $\boldsymbol{s}_0 = [0\ 1\ 0]^\mathrm{T}$, $\boldsymbol{s}_1 = [1\ 0\ 0]^\mathrm{T}$ and

$$\boldsymbol{I}_0^\mathrm{c} = \boldsymbol{I}_1^\mathrm{c} = \begin{bmatrix} I & -mc_y & mc_x \\ -mc_y & m & 0 \\ mc_x & 0 & m \end{bmatrix} \tag{17}$$

(planar vectors and matrices—see [5, Sect. 2.16]). It therefore follows that $H_{01} = -mc_y$ and $H_{11} = I$, implying that

$$T_\mathrm{c}^2 = \frac{-H_{11}}{gH_{01}} \,. \tag{18}$$

On comparing this with Eq. 11 it can be seen that

$$T_\mathrm{c}^2 = \frac{-Y_2}{Y_1} \,. \tag{19}$$

The linear velocity gain of a robot mechanism, G_v, as defined in [6], is the ratio of a change in the horizontal velocity of the CoM to the change in velocity of the joint (or combination of joints) that is being used to manipulate the CoM. For the robot in Fig. 1 the velocity gain is

$$G_\mathrm{v} = \frac{\Delta \dot{c}_x}{\Delta \dot{q}_2} \,, \tag{20}$$

where both velocity changes are caused by an impulse about joint 2. The value of G_v can be worked out via the impulsive equation of motion derived from Eq. 7:

$$\begin{bmatrix} \iota_0 \\ 0 \\ \iota_2 \end{bmatrix} = \begin{bmatrix} H_{00} & H_{01} & H_{02} \\ H_{10} & H_{11} & H_{12} \\ H_{20} & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \Delta \dot{q}_1 \\ \Delta \dot{q}_2 \end{bmatrix} , \tag{21}$$

where ι_2 is an arbitrary nonzero impulse. Solving this equation for ι_0 gives

$$\begin{aligned} \iota_0 &= H_{01} \Delta \dot{q}_1 + H_{02} \Delta \dot{q}_2 \\ &= \big(H_{02} - \frac{H_{01} H_{12}}{H_{11}} \big) \Delta \dot{q}_2 = \frac{-D}{H_{11}} \Delta \dot{q}_2 \,. \end{aligned} \tag{22}$$

But ι_0 is the ground-reaction impulse in the x direction, which is the step change in horizontal momentum of the whole robot; so we also have $\iota_0 = m \Delta \dot{c}_x$, and the velocity gain is therefore

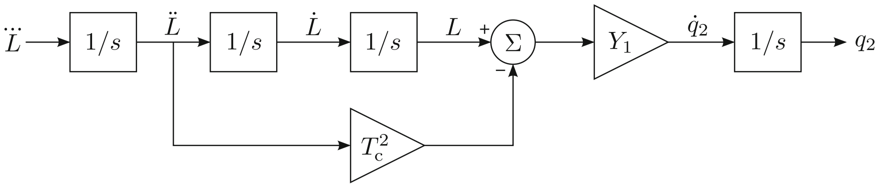


Fig. 3 Alternative version of new plant model for balancing

$$G_v = \frac{\Delta \dot{c}_x}{\Delta \dot{q}_2} = \frac{\iota_0}{m \Delta \dot{q}_2} = \frac{-D}{m H_{11}}. \tag{23}$$

The two plant gains can now be written in terms of T_c and G_v as follows:

$$Y_1 = \frac{1}{m g T_c^2 G_v}, \qquad Y_2 = \frac{-1}{m g G_v}, \tag{24}$$

and another interesting formula for Y_1 is

$$Y_1 = \frac{c_y}{I G_v}. \tag{25}$$

Equation 19 suggests a small modification to the plant model in Fig. 2, in which Y_2 is replaced with T_c^2 as shown in Fig. 3. In this version of the model, it can be seen that everything to the left of Y_1 is concerned with the balancing motion of the robot, while Y_1 describes how the balancing motion affects joint 2. It was mentioned earlier that the balance controller works by tipping the robot slightly off balance, so that the corrective motion causes q_2 to follow the commanded trajectory. The model in Fig. 3 makes this idea a little clearer.

We are now in a position to explain the physical significance of the conditions $D \neq 0$, which is required by the plant model, and $Y_1 \neq 0$, which is required by the control law in the next section. $D \neq 0$ is equivalent to $G_v \neq 0$, and it is the condition for joint 2 to have an effect on the horizontal motion of the CoM. If $D = 0$ in some particular configuration then it is physically impossible for the robot to balance itself in that configuration. $Y_1 = 0$ occurs when $c_y = 0$, which is on the boundary between unstable and stable equilibrium. A similar analysis appears in [1, 3].

3 The Balance Controller

The new plant model is interesting in its own right, but its usefulness lies in the simplicity of the balance controller and the ease with which it can be designed and implemented. Consider the following four-term control law:

$$\dddot{L} = k_{dd}(\ddot{L} - \ddot{L}_c) + k_d(\dot{L} - \dot{L}_c) + k_L(L - L_c) + k_q(q_2 - q_c). \tag{26}$$

When the plant in Fig. 2 is subjected to this control law, the resulting closed-loop equation of motion is

$$\begin{bmatrix} \dddot{L} \\ \ddot{L} \\ \dot{L} \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} k_{dd} & k_d & k_L & k_q \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Y_2 & 0 & Y_1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{L} \\ \dot{L} \\ L \\ q_2 \end{bmatrix} - \begin{bmatrix} k_{dd}\ddot{L}_\mathrm{c} + k_d\dot{L}_\mathrm{c} + k_L L_\mathrm{c} + k_q q_\mathrm{c} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{27}$$

and the characteristic equation of the coefficient matrix is

$$\lambda^4 - k_{dd}\lambda^3 - (k_d + k_q Y_2)\lambda^2 - k_L\lambda - k_q Y_1 = 0\,. \tag{28}$$

The simplest way to choose the gains is by pole placement. As the speed of balancing is determined mainly by the slowest pole, a sensible approach is to place all of the poles at a point $-p$ on the negative real axis, the value of p being chosen by the user, and choose the gains to make Eq. 28 match the polynomial

$$(\lambda + p)^4 = \lambda^4 + 4p\lambda^3 + 6p^2\lambda^2 + 4p^3\lambda + p^4 = 0\,. \tag{29}$$

The resulting gains are

$$\begin{aligned} k_{dd} &= -4p & k_L &= -4p^3 \\ k_d &= -6p^2 + p^4 Y_2/Y_1 & k_q &= -p^4/Y_1\,. \end{aligned} \tag{30}$$

Clearly, another polynomial could be used in place of Eq. 29. The choice of p is not critical, but also not arbitrary: if it is too small then balancing happens too slowly, and if it is too large then the robot overshoots too much. A graph illustrating this effect can be found in [1, p. 37]. Simulation studies suggest that a value around 1.2–1.5 times $1/T_\mathrm{c}$ is about right.

It can be seen from Eq. 30 that this choice of gains is not possible if $Y_1 = 0$. However, this problem is unavoidable because Y_1 appears in the constant term of Eq. 28, so if $Y_1 = 0$ then $\lambda = 0$ is always a root of the characteristic equation regardless of the choice of gains.

The input q_c in Eq. 26 specifies the trajectory that q_2 is being commanded to follow. It can be arbitrary in the sense of not being required to have any particular algebraic form. However, a sufficiently wild or pathological command will cause the robot to fall over. Simulation studies suggest that the most likely cause of failure is if the command makes the robot enter a region of configuration space where the velocity gain is close to zero.

The inputs L_c, $\dot{L}_\mathrm{c}$ and $\ddot{L}_\mathrm{c}$ in Eq. 26 help to improve the tracking accuracy of time-varying trajectories. The simplest choice for these variables is to set them to zero. In this case, the balance controller converges accurately to q_c when it is constant, but does not track accurately when q_c is changing. Nonzero values can improve the tracking accuracy. For example, setting

$$L_c = \frac{\dot{q}_c}{Y_1} \tag{31}$$

produces accurate tracking of linear ramps (constant $\dot{q}_c$). (L_d in [1, 3] achieves the same effect.) Additionally setting

$$\dot{L}_c = \frac{\ddot{q}_c}{Y_1} - \frac{\dot{Y}_1}{Y_1} L_c \tag{32}$$

achieves accurate tracking of parabolic curves (constant $\ddot{q}_c$). However, the improved tracking comes at the expense of increased overshoots and a tendency to over-react to the high-frequency component of the command signal.

The value computed by Eq. 26 is $\dddot{L}$, but the output of the control system has to be either a torque command or an acceleration command for joint 2; that is, either τ_2 or $\ddot{q}_2$. These quantities are computed as follows. First, from Eq. 4 we have $\ddot{L} = -mg\ddot{c}_x$; but $m\ddot{c}_x$ is the x component of the ground reaction force acting on the robot, which is τ_0. So $\dddot{L} = -g\tau_0$. Substituting this into Eq. 7 and rearranging to put all of the unknowns into a single vector produces the equation

$$\begin{bmatrix} 0 & H_{01} & H_{02} \\ 0 & H_{11} & H_{12} \\ -1 & H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \tau_2 \\ \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} -\ddot{L}/g - C_0 \\ -C_1 \\ -C_2 \end{bmatrix}, \tag{33}$$

which can be solved for both τ_2 and $\ddot{q}_2$.

4 Extension to More General Robots

If a robot has more than one actuated motion freedom then two aspects of the balance problem change: (1) there is now a choice of which motion to use for balancing, and (2) there are now motion freedoms that are separate from the balancing activity, and can be controlled with little regard to the balancing activity. These motions can be designed to lie in the balance null space, which is the space of motions that the robot can make that do not affect c_x. However, it may be easier to design these motions to have very little effect on c_x rather than no effect at all.

Let us now replace the double pendulum with a general planar mechanism, retaining only the fictitious prismatic joint and the passive revolute joint that models the contact with the ground. The rest of the mechanism is assumed to be fully actuated, and it may contain kinematic loops. Let $\boldsymbol{y} = [y_0\ y_1\ y_2\ \boldsymbol{y}_3^{\mathrm{T}}]^{\mathrm{T}}$ be a vector of generalized coordinates in which $y_0 = q_0$, $y_1 = q_1$, y_2 is the coordinate expressing the movement to be used for balancing, and $\boldsymbol{y}_3$ is a vector containing the rest of the generalized coordinates. The movement expressed by y_2 can be any desired combination of the actuated joint motions. In effect, y_2 is the variable of a user-defined virtual joint that is a generalization of joint 2 in the previous sections. The equation of motion of this system is

$$\begin{bmatrix} H_{00} & H_{01} & H_{02} & \boldsymbol{H}_{03} \\ H_{10} & H_{11} & H_{12} & \boldsymbol{H}_{13} \\ H_{20} & H_{21} & H_{22} & \boldsymbol{H}_{23} \\ \boldsymbol{H}_{30} & \boldsymbol{H}_{31} & \boldsymbol{H}_{32} & \boldsymbol{H}_{33} \end{bmatrix} \begin{bmatrix} 0 \\ \ddot{q}_1 \\ \ddot{y}_2 \\ \ddot{\boldsymbol{y}}_3 \end{bmatrix} + \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ \boldsymbol{C}_3 \end{bmatrix} = \begin{bmatrix} \tau_0 \\ 0 \\ u_2 \\ \boldsymbol{u}_3 \end{bmatrix}, \tag{34}$$

in which u_2 and $\boldsymbol{u}_3$ are the generalized forces corresponding to y_2 and $\boldsymbol{y}_3$, and H_{ij} are now the elements and submatrices of a generalized inertia matrix. This equation replaces Eq. 7. Equations 2–4 and 6 remain valid, but Eq. 5 becomes

$$L = H_{11}\dot{q}_1 + H_{12}\dot{y}_2 + \boldsymbol{H}_{13}\dot{\boldsymbol{y}}_3 \,. \tag{35}$$

Likewise, Eq. 8 becomes

$$-\ddot{L}/g = H_{01}\dot{q}_1 + H_{02}\dot{y}_2 + \boldsymbol{H}_{03}\dot{\boldsymbol{y}}_3 \,, \tag{36}$$

and so Eq. 9 becomes

$$\begin{bmatrix} L \\ \ddot{L} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} \\ -gH_{01} & -gH_{02} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{y}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{H}_{13} \\ -g\boldsymbol{H}_{03} \end{bmatrix} \dot{\boldsymbol{y}}_3 \,. \tag{37}$$

Solving this equation for $\dot{y}_2$ gives

$$\dot{y}_2 = Y_1 L + Y_2 \ddot{L} - \boldsymbol{Y}_3 \dot{\boldsymbol{y}}_3 \,, \tag{38}$$

where Y_1 and Y_2 are as given in Eq. 11, and

$$\boldsymbol{Y}_3 = \frac{\boldsymbol{E}}{D} \tag{39}$$

where

$$\boldsymbol{E} = \boldsymbol{H}_{13} H_{01} - H_{11} \boldsymbol{H}_{03} \tag{40}$$

(cf. Eq. 12). The modified plant model is shown in Fig. 4. Observe that the influence of the non-balance motions is limited to the value of the scalar signal $\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3$. If this signal is zero then these motions have no effect.

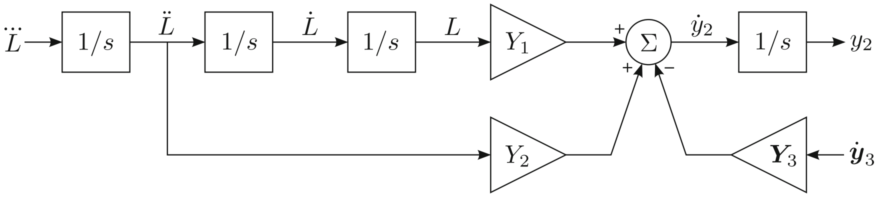


Fig. 4 Modified plant model for a general planar robot

The design of the control system is largely unaffected by $\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3$. In particular, Eq. 28 is unaffected, and the gains are still as given in Eq. 30. However, there is scope to include terms in L_c and $\dot{L}_c$ to counteract the disturbances caused by $\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3$. For example, one could use

$$L_c = \frac{\dot{y}_c}{Y_1} + \frac{\boldsymbol{Y}_3\dot{\boldsymbol{y}}_3}{Y_1} \tag{41}$$

in place of Eq. 31. Simulation studies indicate that this modification successfully compensates for the low-frequency component of the disturbance, but causes the balance controller to over-react to the high-frequency component. A low-pass filter may help in this regard, but it is probably better to design the non-balance motion to lie substantially in the balance null space so that $\boldsymbol{Y}_3$ is close to zero.

Finally, the generalized forces must be calculated and mapped to the actuated joints. The first step is to solve

$$\begin{bmatrix} 0 & \mathbf{0} & H_{01} & H_{02} \\ 0 & \mathbf{0} & H_{11} & H_{12} \\ -1 & \mathbf{0} & H_{21} & H_{22} \\ \mathbf{0} & -\mathbf{1} & \boldsymbol{H}_{31} & \boldsymbol{H}_{32} \end{bmatrix} \begin{bmatrix} u_2 \\ \boldsymbol{u}_3 \\ \ddot{q}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} -\dddot{L}/g - C_0 - \boldsymbol{H}_{03}\ddot{\boldsymbol{y}}_3 \\ -C_1 - \boldsymbol{H}_{13}\ddot{\boldsymbol{y}}_3 \\ -C_2 - \boldsymbol{H}_{23}\ddot{\boldsymbol{y}}_3 \\ -\boldsymbol{C}_3 - \boldsymbol{H}_{33}\ddot{\boldsymbol{y}}_3 \end{bmatrix}, \tag{42}$$

which is the generalization of Eq. 33. In this equation, $\ddot{\boldsymbol{y}}_3$ is the desired acceleration calculated by a separate motion control law responsible for $\boldsymbol{y}_3$. The final step is to calculate

$$\boldsymbol{\tau}_a = \boldsymbol{G}^{-T} \begin{bmatrix} u_2 \\ \boldsymbol{u}_3 \end{bmatrix}, \tag{43}$$

where $\boldsymbol{\tau}_a$ is the vector of force variables at the actuated joints, and $\boldsymbol{G}$ is the matrix (chosen by the user) that maps $[\dot{y}_2 \; \dot{\boldsymbol{y}}_3^T]^T$ to the vector of actuated joint velocities.

5 Simulation and Analysis

This section presents a simulation experiment in which the balance controller makes an inverted triple pendulum perform a variety of manoeuvres while maintaining its balance. A triple pendulum is chosen because it is the simplest mechanism that exhibits all of the phenomena discussed in this paper.

The robot is a 3R planar kinematic chain that moves in the vertical plane. Joint 1 is passive, and the robot is pointing straight up in the configuration $q_1 = q_2 = q_3 = 0$. The link lengths are 0.2, 0.25 and 0.35 m, and the masses are 0.7, 0.5 and 0.3 kg. The links are modelled as point masses with the mass located at the far end of each link. These are the parameters of a mechanism identified in [6] as being good at balancing.

The control system consists of the balance controller of Sect. 3, which controls the generalized coordinate y_2, plus a PD position controller with exact inverse dynamics, which controls y_3. The tracking accuracy of the latter is essentially perfect everywhere except where there is a step change in commanded velocity. The balance controller is based on Eq. 26; the gains are as given in Eq. 30 with $p = 7$ rad/s; L_c and $\dot{L}_c$ are as given in Eqs. 41 and 32; and $\ddot{L}_c = \dddot{y}_c / Y_1$. The position controller's gains are chosen to put both poles at 14 rad/s in order to make the point that the control of variables not used for balancing can take place at a higher frequency than that chosen for the balance controller.

Figure 5a shows the command signals and response, both expressed in generalized coordinates. Times are expressed in seconds and angles in radians. To show the effect of L_c, $\dot{L}_c$ and $\ddot{L}_c$, this graph includes a signal 'y2o' showing what the response of the balance controller would have been if $L_c = \dot{L}_c = \ddot{L}_c = 0$. Note that these are relatively large, fast motion commands. Comparable graphs in the literature (e.g. [7]) typically show slower, smaller movements which can be tracked more accurately.

The commands consist of a step, a ramp and a sine wave for y_2 while y_3 is held at zero, a ramp of y_3 while y_2 is held at zero, and finally a ramp of y_2 with y_3 held at 1.5. Up until the final ramp, y_2 and y_3 are defined by $y_2 = q_2$ and $y_3 = q_3$, but then y_2 is redefined to be $y_2 = q_2 - q_3$. So the final ramp involves q_2 ramping from 0 to 1.5 while q_3 ramps from 1.5 to 0. This can be clearly seen in Fig. 5b, which shows the motion of the robot expressed in joint space.

One obvious feature of Fig. 5a is the reverse movements at the beginning of each y_2 manoeuvre and the overshoots at the end. These movements are physically necessary for maintaining the robot's balance. However, the magnitudes of some of these movements (e.g. the one at 9.5 s) are probably larger than the minimum necessary. Note also that the ramp in y_3 disturbs y_2 only at the beginning and end of the ramp. In the middle portion, the balance controller has successfully compensated for the disturbance caused by this motion, thanks to the term $Y_3 \dot{y}_3 / Y_1$ in Eq. 41.

Figure 5c, d show the values of T_c and Y_1. Observe that T_c varies in a narrow range from approximately 0.21–0.233 s even though the robot is making large changes in its configuration. This is a property of the robot mechanism, and will vary from one robot to the next. However, for most balancing robots it will typically be the case that T_c does not vary very much. This suggests that assuming a constant value for T_c could be a good approximation.

For the first 13.5 s Y_1 varies in a range from approximately 26 to 33. However, at the point where y_2 is redefined, it jumps to 282, and then rises to 311 and drops to 88 over the course of the final ramp and its overshoot. So for the first 13.5 s the plant model is only slightly nonlinear, with the two gains varying in a narrow range, but then the situation changes when y_2 is redefined.

The explanation can be found in Fig. 5e, which plots the velocity gains of joints 2 and 3 along with their difference, which is the velocity gain of the motion freedom $q_2 - q_3$ [6]. For the first 13.5 s $G_v(y_2) = G_v(q_2)$; but then y_2 is redefined, and for the remaining time $G_v(y_2) = G_v(q_2) - G_v(q_3)$. As $G_v(y_2)$ appears in the denominator of Eq. 24, this accounts for the large change in Y_1.

So from this brief analysis we can conclude the following: the robot is generally well-behaved, and the plant model is only slightly nonlinear, up until the beginning of the final ramp. But then the balance controller is given an especially bad new

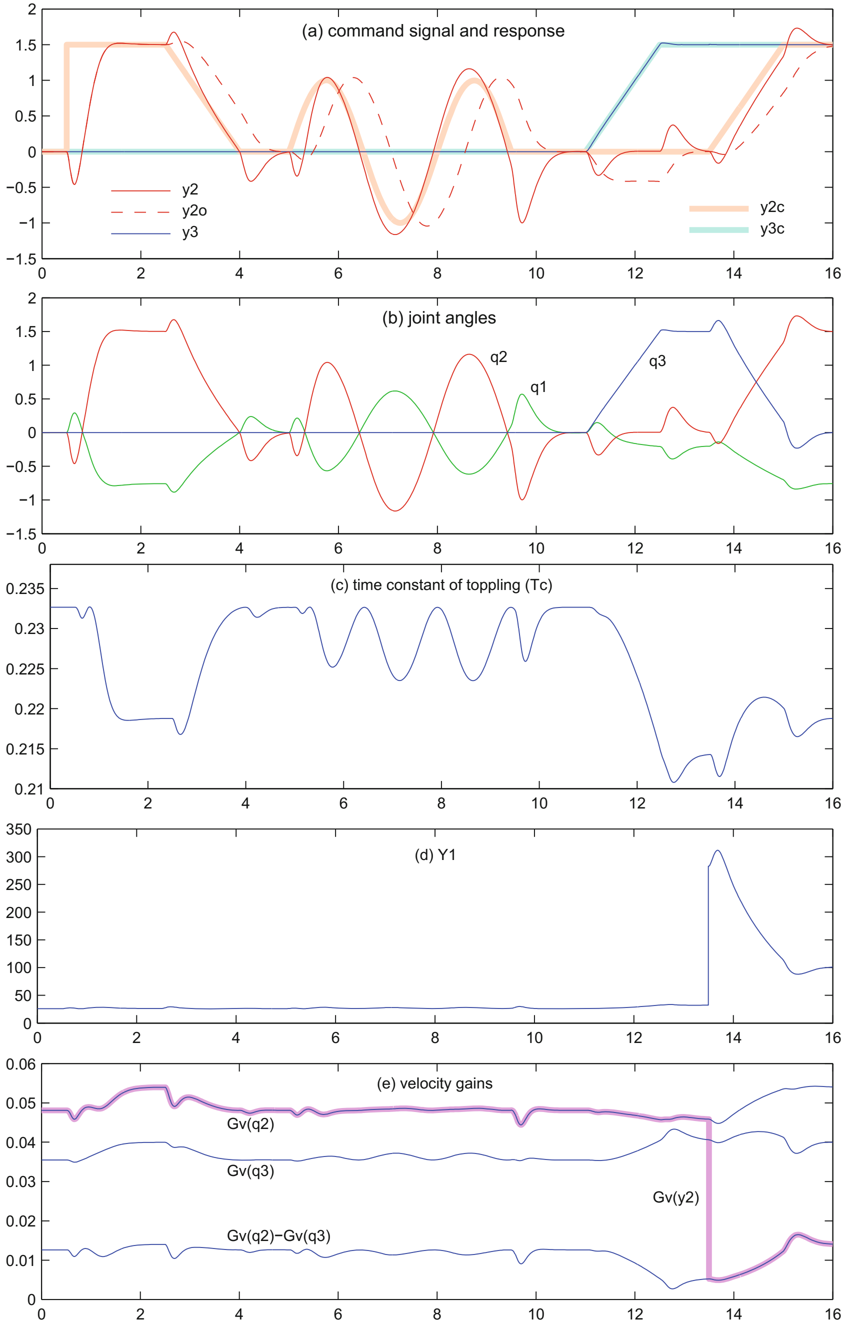


Fig. 5 Simulation results for balancing a triple pendulum (times in seconds, angles in radians)

definition of y_2: a motion that has almost no effect on the CoM (i.e., a velocity gain close to zero). So the final ramp is an especially difficult command to follow, and that is why the controller does not track this ramp as accurately as the first ramp. Without an analysis of the physics of the balancing process, it is not at all obvious why the tracking of the final ramp is not as good as the first.

6 Conclusion

This paper has presented a new model of the physical process of balancing by a general planar robot. The essential parameters of the robot's balancing behaviour are reduced to just two numbers, plus a third number to describe the influence of all other movements on the balancing behaviour. All three numbers can be computed efficiently using standard dynamics algorithms. The model gives rise to a simple balance controller that allows the robot to balance while performing other motions; and simulation results are presented showing the controller making a triple pendulum perform a variety of large, fast movements while maintaining its balance. The controller allows complete freedom in choosing which movements are to be used for balancing and which for other tasks.

As planar balancing is a solved problem, the contribution of this paper is to simplify the problem and its solution without loss of generality, and to present an approach to balancing that appeals more to the physical process of balancing and less to the control theory. Clearly, the ultimate objective is a simpler theory of balancing in 3D, and a first step in that direction appears in [1, 2].

Acknowledgements The work presented here owes much to the work of Morteza Azad as described in [1].

Appendix

This appendix proves the result that $p_i = \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{h}_{\nu(i)}$, where p_i and $\boldsymbol{s}_i$ are the momentum variable and axis vector of joint i, and $\boldsymbol{h}_{\nu(i)}$ is the total momentum of the subtree or self-contained subsystem consisting of body i and its descendants. A self-contained subsystem, in this context, is defined to be a subsystem in which every kinematic loop that involves at least one body in the subsystem is entirely contained within the subsystem. In general, $\boldsymbol{s}_i$ and $\boldsymbol{h}_{\nu(i)}$ will be spatial vectors. However, if the whole system is planar then they may instead be planar vectors.

Consider a kinematic tree consisting of N bodies and joints numbered from 1 to N according to a regular numbering scheme. Without loss of generality, we assume that every joint has only a single degree of freedom (DoF), which means that every multi-DoF joint has already been replaced by a kinematically equivalent chain of single-DoF joints connected by massless bodies, and that these extra bodies and joints are already included in N.

Let $\boldsymbol{p}$ and $\dot{\boldsymbol{q}}$ denote the joint-space momentum and velocity vectors of the tree, or the spanning tree if there are kinematic loops. By definition, the two are related by the equation

$$\boldsymbol{p} = \boldsymbol{H}\dot{\boldsymbol{q}} \,, \tag{44}$$

where $\boldsymbol{H}$ is the joint-space inertia matrix. This implies that

$$p_i = \sum_{j=1}^{N} H_{ij}\dot{q}_j \,. \tag{45}$$

The definition of $\boldsymbol{H}$ for a general kinematic tree with single-DoF joints is

$$H_{ij} = \begin{cases} \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{I}_{\nu(i)}\boldsymbol{s}_j & \text{if } i \in \nu(j) \\ \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{I}_{\nu(j)}\boldsymbol{s}_j & \text{if } j \in \nu(i) \\ 0 & \text{otherwise} \end{cases} \tag{46}$$

where $\boldsymbol{s}_i$ is the joint axis vector (i.e., joint motion subspace vector) of joint i, $\boldsymbol{I}_i$ is the inertia of body i (spatial or planar as appropriate), $\nu(i)$ is the set of all bodies in the subtree beginning at body i, and $\boldsymbol{I}_{\nu(i)} = \sum_{j\in\nu(i)} \boldsymbol{I}_j$.

Let $\bar{\kappa}(i)$ be the set of all bodies on the path between body i and the base (body 0), excluding both body i and the base, and let $\kappa(i) = \bar{\kappa}(i) \cup \{i\}$ be the same set including body i. If we use the terms 'ancestor' and 'descendant' in an inclusive sense, meaning that body i is both an ancestor and a descendant of itself, and use the term 'proper ancestor' in an exclusive sense, then the sets $\nu(i)$, $\kappa(i)$ and $\bar{\kappa}(i)$ can be seen to be the sets of descendants, ancestors and proper ancestors, respectively, of body i. $\kappa(i)$ is also the set of joints on the path between body i and the base.

We now rewrite Eq. 46 as follows:

$$H_{ij} = \begin{cases} \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{I}_{\nu(i)}\boldsymbol{s}_j & \text{if } j \in \bar{\kappa}(i) \\ \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{I}_{\nu(j)}\boldsymbol{s}_j & \text{if } j \in \nu(i) \\ 0 & \text{otherwise} \end{cases} \tag{47}$$

which makes it clear that H_{ij} is nonzero only if $j \in \bar{\kappa}(i)$ or $j \in \nu(i)$. Substituting Eq. 47 into Eq. 45 gives

$$\begin{aligned}
p_i &= \boldsymbol{s}_i^{\mathrm{T}}\Big(\sum_{j\in\bar{\kappa}(i)} \boldsymbol{I}_{\nu(i)}\boldsymbol{s}_j\dot{q}_j + \sum_{j\in\nu(i)} \boldsymbol{I}_{\nu(j)}\boldsymbol{s}_j\dot{q}_j\Big) \\
&= \boldsymbol{s}_i^{\mathrm{T}}\Big(\sum_{j\in\bar{\kappa}(i)} \sum_{k\in\nu(i)} \boldsymbol{I}_k\boldsymbol{s}_j\dot{q}_j + \sum_{j\in\nu(i)} \sum_{k\in\nu(j)} \boldsymbol{I}_k\boldsymbol{s}_j\dot{q}_j\Big) \\
&= \boldsymbol{s}_i^{\mathrm{T}}\Big(\sum_{k\in\nu(i)} \sum_{j\in\bar{\kappa}(i)} \boldsymbol{I}_k\boldsymbol{s}_j\dot{q}_j + \sum_{k\in\nu(i)} \sum_{j\in\nu(i)\cap\kappa(k)} \boldsymbol{I}_k\boldsymbol{s}_j\dot{q}_j\Big) \\
&= \boldsymbol{s}_i^{\mathrm{T}} \sum_{k\in\nu(i)} \boldsymbol{I}_k \sum_{j\in\kappa(k)} \boldsymbol{s}_j\dot{q}_j \,. \qquad (48)
\end{aligned}$$

The step from the second to the third line works as follows: $\sum_{j\in\nu(i)}\sum_{k\in\nu(j)}$ is the sum over all j, k pairs where j is a descendant of i and k is a descendant of j, whereas $\sum_{k\in\nu(i)}\sum_{j\in\nu(i)\cap\kappa(k)}$ is the sum over all j, k pairs where k is a descendant of i, and j is both a descendant of i and an ancestor of k; but these two sets of pairs are the same. The step from the third to the fourth line exploits the fact that $\nu(i)\cap\kappa(k)$ is the set of all ancestors of body k from i onwards, whereas $\bar{\kappa}(i)$ is the set of all ancestors of body k prior to i, so the union of the two sets is $\kappa(k)$.

Now let $\boldsymbol{v}_k$ be the velocity of body k, let $\boldsymbol{h}_k = \boldsymbol{I}_k\boldsymbol{v}_k$ be the momentum of body k, and let $\boldsymbol{h}_{\nu(i)} = \sum_{k\in\nu(i)} \boldsymbol{h}_k$ be the total momentum of the subtree beginning at body i. The velocity of any body in a rigid-body system is the sum of the joint velocities along any one path between that body and the base, so $\boldsymbol{v}_k = \sum_{j\in\kappa(k)} \boldsymbol{s}_j\dot{q}_j$. We can now further simplify Eq. 48 as follows:

$$p_i = \boldsymbol{s}_i^{\mathrm{T}} \sum_{k\in\nu(i)} \boldsymbol{I}_k\boldsymbol{v}_k = \boldsymbol{s}_i^{\mathrm{T}} \sum_{k\in\nu(i)} \boldsymbol{h}_k = \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{h}_{\nu(i)} \,, \qquad (49)$$

which establishes the desired result for the case of a kinematic tree. If the system contains kinematic loops then we find that Eq. 49 no longer holds for all joints, but does still hold for any joint that is not involved in any kinematic loop. This is equivalent to the condition stated at the beginning that the subsystem consisting of the bodies in $\nu(i)$ be self-contained.

References

1. Azad, M.: Balancing and hopping motion control algorithms for an under-actuated robot. Ph.D. thesis, The Australian National University, School of Engineering (2014)
2. Azad, M., Featherstone, R.: Balancing control algorithm for a 3D under-actuated Robot. In: Proceedings of the IEEE/RSJ International Conference Intelligent Robots & Systems, pp. 3233–3238. Chicago, IL, 14–18 Sept (2014)
3. Azad, M., Featherstone, R.: Angular momentum based balance controller for an under-actuated planar robot. Auton. Robot. (2015). http://dx.doi.org/10.1007/s10514-015-9446-z
4. Double Robotics ‘Double’ telepresence robot. http://www.doublerobotics.com (2014). Accessed Aug 2015
5. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, New York (2008)

6. Featherstone, R.: Quantitative measures of a robot's ability to balance. In: Robotics Science & Systems. Rome. http://www.roboticsproceedings.org/rss11/p26.html. 13–17 July (2015). Accessed Aug 2015
7. Grizzle, J.W., Moog, C.H., Chevallereau, C.: Nonlinear control of mechanical systems with an unactuated cyclic variable. IEEE Trans. Automat. Control **50**(5):559–576 (2005). http://dx.doi.org/10.1109/TAC.2005.847057
8. Hauser, J., Murray, R.M.: Nonlinear controllers for non-integrable systems: the acrobot example. Proc. Am. Control Conf. Nov **3–5**, 669–671 (1990)
9. Miyashita, N., Kishikawa, M., Yamakita, M.: 3D motion control of 2 links (5 DOF) underactuated manipulator named AcroBOX. In: Proceedings of the American Control Conference, pp. 5614–5619. Minneapolis, MN, 14–16 June (2006)
10. Segway Inc. Personal transporter. http://www.segway.com (2015). Accessed Aug 2015
11. Spong, M.W.: The swing up control problem for the acrobot. IEEE Control Syst. Mag. **15**(1), 49–55 (1995)
12. Xin, X., Kaneda, M.: Swing-up control for a 3-DOF gymnastic robot with passive first joint: design and analysis. IEEE Trans. Robot. **23**(6), 1277–1285 (2007)
13. Yonemura, T., Yamakita, M.: Swing up control of acrobot based on switched output functions. In: Proceedings of the SICE 2004 Aug, vol. 4–6, pp. 1909–1914. Sapporo, Japan (2004)

Slip Detection and Recovery for Quadruped Robots

Michele Focchi, Victor Barasuol, Marco Frigerio, Darwin G. Caldwell and Claudio Semini

1 Introduction

Being able to deal with slippage is of great importance for legged robots which are meant to traverse unstructured terrains. In particular, a strategy for detecting slippage and recover from it, becomes crucial when whole body inverse dynamics approaches are implemented for robot control [1, 14, 18]. Actually, they rely on the assumption that the stance feet constraints are not violated (e.g. they are not moving or are supposed to move very little [8]). Indeed, a violation of the stance constraints makes the inverse dynamics compute joint torques which are not physically meaningful anymore. This would result in: (i) errors in the realization of the desired body wrench (because the slip limits the amount of tangential force that the ground is able to deliver) (ii) degeneration of the support triangle. These two facts would eventually lead to a loss of stability even in case of very slow motions. On the same line, kinematic-based state estimation or odometry techniques [3, 6], which rely on the assumption that none of the stance feet is slipping, are prone to drift if the amount of slip is relevant (or if there is a compliance between the base and the ground which is not modelled). Even if the controller incorporates the optimization of the ground

M. Focchi (✉) · V. Barasuol · M. Frigerio · D.G. Caldwell · C. Semini
Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT),
via Morego, 30, 16163 Genova, Italy
e-mail: michele.focchi@iit.it

V. Barasuol
e-mail: victor.barasuol@iit.it

M. Frigerio
e-mail: marco.frigerio@iit.it

D.G. Caldwell
e-mail: darwin.caldwell@iit.it

C. Semini
e-mail: claudio.semini@iit.it

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_11

reaction forces, there are two types of uncertainties which can cause slippage during locomotion:

1. Uncertainty on the estimate of the surface normal $\mathbf{n}$. This is commonly estimated by vision [17] or by fitting a plane (gradient-based terrain detection) through the feet that are on the ground (*stance feet*) [26]. The fitting plane can be a very crude approximation of the terrain surface inclination which can have local discontinuities (e.g. like the ramp illustrated in Fig. 4). Any deviation from a planar shape results in errors in the estimation of the inclination of the surface which is under the foot at the moment of the touch-down.
2. Uncertainty on the friction coefficient μ. Most of the times μ cannot be known in advance and is commonly inferred according to semantic information (e.g. ice) coming from vision [22].

An earlier work on slip recovery is from Takemura [26], who presented both a *long term* and a *short term* strategy for slip recovery. The former aims to change gait frequency and stride length when approaching slippery surfaces. However, changing locomotion parameters to address slippage can be successful only on terrain with limited roughness and moderate slipperiness. Conversely, if very challenging environment is considered (e.g. crossing a river or walking on ice), the occurrence of slippage might result in unrecoverable loss of stability because any other footstep can be infeasible. A *short term strategy* is needed in these cases. At this extent, Takemura proposes to instantaneously add a force (at the occurrence of slippage) to have the ground reaction forces (GRFs) back in the friction cone. This approach has several shortcomings: (1) it is based on the idea that the normal is properly estimated; (2) the required force might not be necessarily realizable at the foot, since the ground reaction force is the result of the robot motion in interaction with the environment. More precisely, the GRFs can only be controlled to a limited extent (e.g. creating internal forces) in the null-space of the contact constraints. In addition to this (in static conditions) the maximum applicable total normal force is constrained by the robot weight.

To address the above limitations we propose a *short term* slip recovery strategy, which is built on top of a whole body controller [9, 10]. In essence the controller we use, is formulated as a QP problem where the goal is to realize a certain body wrench while optimizing for ground reaction forces (decision variables). We added inequality constraints to the problem to obtain forces that obey friction cones limits, additionally accounting for the fact that the ground can only push and not pull (unilateral constraints). The body wrench is the 6D vector of forces/moments coming from the body motion task that aims to track a specified trajectory for the CoM and the trunk orientation.

In this context, assuming a reliable low level tracking of the joint torques and no external disturbances, a slippage can only occur if the controller uses a *wrong* estimate of the surface normal or of the friction coefficient. In fact, this results in GRFs which are out of the real friction cones (cf. Sect. 4.1). Therefore, differently from Takemura, instead of striving to apply the correct GRFs (which might not be feasible), we propose to *correct* the estimate of the surface inclination and of

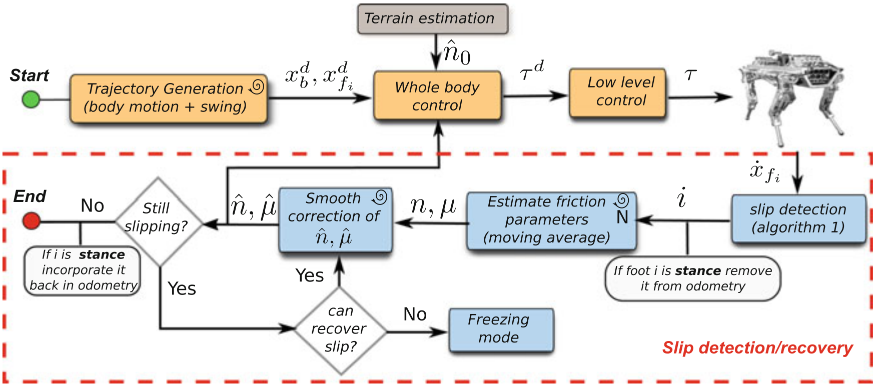

Terrain estimation
$\hat{n}_0$
Start
Trajectory Generation (body motion + swing)
$x_b^d, x_{f_i}^d$
Whole body control
τ^d
Low level control
τ
$\dot{x}_{f_i}$
End
No
Still slipping?
If i is **stance** incorporate it back in odometry
$\hat{n}, \hat{\mu}$
Smooth correction of $\hat{n}, \hat{\mu}$
n, μ
Estimate friction parameters (moving average)
N
i
slip detection (algorithm 1)
If foot i is **stance** remove it from odometry
Yes
Yes
can recover slip?
No
Freezing mode
Slip detection/recovery

Fig. 1 Slip recovery pipeline (*blue blocks*): detection, estimation and correction. The *yellow blocks* are: the body, feet trajectory generation and the high level control (whole body) and the low level one (torque)

the friction coefficient in order to allow the robot to apply forces which satisfy the friction limits.

More recently, in one of the online videos,[1] the BigDog robot demonstrated to successfully recover from slipping on ice. However, to date, no experimental results have been published and no details have been reported on the repeatability of the used approach.

Within the context of legged robots, the main contribution of this paper is a methodology to: (1) detect slippage, (2) on-line estimate the friction coefficient μ and the normal n of the surface making use of only proprioceptive information (torque measurement and encoder readings) (3) on-line recovery from slippage by smoothly accommodating (but in a short time interval) the value of the normal used in the optimization to the estimated one. The whole *slip recovery* pipeline is graphically summarized in Fig. 1. We have implemented the slip detection/estimation/recovery strategy for a model of HyQ [24], a 80 kg quadruped robot with point feet. HyQ is shown to perform a (statically stable) walk on a highly slippery (flat) surface and on a moderately slippery ramp.

The applicability of this approach is limited to torque controlled robots equipped with a high-level controller which optimizes for ground reaction forces. We will show simulations where the slip is detected only in one leg and when there are at least three legs in contact with the ground. A possible solution for the detection in the case of more than two legs slipping is only drafted.

This paper is structured as follows: Sect. 2 presents a robust way to detect slippage, followed by Sect. 3 that illustrates the on-line estimation of the friction parameters. Section 4 describes the implemented strategy to recover traction during slippage. Sections 5 and 6 contain the results of simulations with HyQ and the conclusions, respectively.

[1] Video available at http://www.youtube.com/watch?v=cNZPRsrwumQ.

2 Slip Detection

The approaches to address the problem of slip detection can be divided into two big groups: force based and kinematic based approaches. The force based, require the availability of a 6-axis force sensor which is usually located at the contact point (e.g. the foot-tip). If the friction coefficient is known in advance, the slippage could simply be detected by checking if the ratio of the tangential/normal component of the contact forces [20] is within the limit of static friction. When the friction coefficient μ is unknown, a possible strategy is to check the frequency content of the tangential contact force signal. As a matter of facts, in presence of slippage, a high frequency ripple in the force signal appears, due to the local stick-slip phenomena that occurs between the contacting surfaces. First Holweg [15] and more recently Palli [21], claimed that, after performing a Fourier analisis (FFT) of the higher harmonics of the force signal, it was possible to recognize the deformations which precedes the real slip. These approaches are of limited applicability to legged robots, because they need a high cost force/torque sensor to be attached to the foot tip. However, due to the repetitive impacts with the ground, in the long run, this can result in a damage of the sensor. Furthermore, during locomotion, the touchdown event can create discontinuities in the force signals and jeopardize the detection. As a matter of facts, it is not easy to measure the instant when the force oscillation, due to the touch-down, has settled down, in order to have a detection without false positives. Conversely, a detection strategy based on kinematics, it preferable in the context of legged robots where ground impacts are the order of the day.

A kinematic strategy can be implemented at the acceleration, velocity or position level. In [26] Takemura considered slippage as an impulse-like leg acceleration, and attached accelerometers to the lower-leg links to detect slippage. A drawback of this approach is that accelerometers are usually affected by noise.

Alternatively, slippage could be estimated at the position level, by checking the inter-distances between the stance feet. Indeed slippage of one (or more) feet can be detected if the mutual distance of the stance feet (which is set at the moment a new touchdown event occurs) changes within the duration of a single stance configuration. However, when traction is lost the resulting acceleration will create also a velocity difference among the stance feet. Being the position the integral of velocity this difference can be detected more promptly in velocity than in position. Thus, we propose to check the slippage at the velocity level. It is important to underline that the choice of the frame in which the feet velocities are compared, directly affects the *robustness* of the approach. The most intuitive way is to check if the Cartesian velocities of the stance feet are all *zero* in an inertial (*world* $\mathcal{W}$) frame. However, expressing the foot velocity in the world frame (${}_w\dot{\mathbf{x}}_f$), requires an estimation of the robot base linear velocity ${}_w\dot{\mathbf{x}}_b$:

$$0 \approxeq {}_w\dot{\mathbf{x}}_f = {}_w\dot{\mathbf{x}}_b + {}_w\mathbf{R}_b \underbrace{\left({}_b\dot{\mathbf{x}}_f + {}_b\boldsymbol{\omega}_b \times {}_b\mathbf{x}_f\right)}_{{}_b\dot{\tilde{\mathbf{x}}}_f} \tag{1}$$

where ${}_w\mathbf{R}_b \in \mathbb{R}^{3\times3}$ is the rotation matrix representing the orientation of the robot base, while ${}_b\boldsymbol{\omega}_b$ is its angular velocity. ${}_b\mathbf{x}_f$, ${}_b\dot{\mathbf{x}}_f$ are, respectively, the position and velocity of the foot expressed in the base frame ($\mathcal{B}$). The angular velocity ${}_b\omega_b$ can be measured with reasonable accuracy by an on-board IMU sensor, while the base linear velocity, as common practice in robotics, can be inferred through leg odometry or state estimation techniques [3]. Therefore, if we compare the feet velocities in the world frame, they are influenced by errors in the state estimation, which can result into false positives in the slip detection. A more robust approach would be to compare the feet velocities in the base frame (term ${}_b\dot{\tilde{\mathbf{x}}}_f$ in (1) which accounts also for the influence of the moving frame). The advantage of this, is that the kinematics is always accurate because it directly depends on direct sensor measurements (e.g. encoders, gyro). Differently from the world frame case, the stance feet velocities ${}_b\dot{\tilde{\mathbf{x}}}_f$ have to be equal in norm and direction (and opposite to the base linear velocity ${}_b\dot{\mathbf{x}}_b$). Thus, in a manner similar to what a car ABS braking system is doing [2], a fruitful strategy for slip detection is to compare the values of the norm $v_i = \|{}_b\dot{\tilde{\mathbf{x}}}_{f_i}\|$ of the velocities of the stance feet and discriminate the outlier with appropriate statistical tools. Henceforth, for the sake of brevity, we denote the norm with v and the associated Cartesian vector with $\mathbf{v}$.

One leg slip detection: Following, we present a pseudo-code implementation of the slip detection for one leg of a legged robot:

Algorithm 1 detectSlippageOneLeg()

```
for each $stance\_leg \quad i$ do
    $vel\_norm[i] \leftarrow |{}_b\dot{\mathbf{x}}_{f_i}|$
end for
$M \leftarrow median(vel\_norm)$
for each $stance\_leg \quad i$ do
    $legSlippingFlag[i] \leftarrow abs(vel\_norm[i] - median) > th$
end for
```

At each control loop the median of the norms of the stance feet velocities is computed. The median will have a value in between the velocities of the non slipping legs and the slipping one. The slipping leg will be the one with a velocity far from the median beyond a certain margin th, that can be tuned experimentally. During locomotion the detection algorithm is continuously checking, within the set of active stance legs, if there is any slippage. Whenever a slip is detected, if the leg was a stance leg, this should be removed from the set of stance legs accounted in the state estimation/odometry, while it can be incorporated back at the end of the slip (see Fig. 1). This is crucial to prevent the corruption of the state estimate.

Multiple leg slip detection: A more subtle situation is when two legs are slipping at the same time. In this case, it is hard to detect with the median approach, who are the slipping legs and who are the stance legs (we just know that they have pairwise different velocities). In this case, checking which of the feet velocities $\mathbf{v}_i$ are kinematically consistent with the base velocity ${}_b\dot{\mathbf{x}}_b$, could help us to discriminate the slipping legs. At this extent a short-time integration of the base linear acceleration (IMU) can be the only resort. It is known that integrating accelerometers is prone to drift but, for a limited time interval, the estimate should be accurate enough.

3 Surface Normal and Friction Coefficient Estimation

Once that the slip is in act, it is crucial to estimate the friction coefficient μ and the surface normal n in the early milliseconds of slippage, in order to be able to apply a corrective action as described in Sect. 4.

Remark: Along the paper, we will not make a distinction between static and dynamic friction.

Firstly, we make the following assumptions:

Assumption 1: The frictional properties of the surface around the foot are isotropic (coefficient of friction equal in all directions).

Assumption 2: when the leg starts to slip (it will start slow), it will cover a surface where the normal is uniformly constant.

Assumption 3: we assume no soft contact. Since the robot has point feet we can neglect the influence of the rotational friction about the normal direction (more complex than the linear one because it depends on the size of the contact area).

We can get useful insights for the estimation considering the following facts:

1. if unilateral constraints are always satisfied (e.g. the legs are always pushing on the ground and the feet are not detaching), the direction of the foot slip velocity **v** will always be tangent to the surface. This means the surface normal **n** forms a right angle with the velocity vector **v**. Furthermore, physics tells us that the normal should lie in the plane Π passing through the direction of the ground force **F** and of the velocity **v** (see Fig. 2(left)). These two facts allow us to easily determine the surface normal **n** by simple geometric computations:

$$\boldsymbol{\pi} = \frac{\mathbf{v} \times \mathbf{F}}{\|\mathbf{v} \times \mathbf{F}\|} \qquad\qquad \mathbf{n} = \frac{\boldsymbol{\pi} \times \mathbf{v}}{\|\boldsymbol{\pi} \times \mathbf{v}\|} \tag{2}$$

   where $\boldsymbol{\pi}$ is the normal to the plane Π.
2. during the slipping motion, the ground force **F** is always lying on one edge of the friction cone. Therefore, *while* the foot is slipping, the angular distance ϕ between **F** and **n** (see Fig. 2), can be an estimate of the real friction coefficient ($\mu = tan(\phi)$):

$$\mu = tan(\phi) = \frac{sin(\phi)}{cos(\phi)} = \frac{\|\mathbf{F} \times \mathbf{n}\|}{\mathbf{F} \cdot \mathbf{n}} \tag{3}$$

To obtain a noise-free estimation of the normal **n**, we compute a moving average on N samples, by using a parametric representation of the geodesic [12], while for

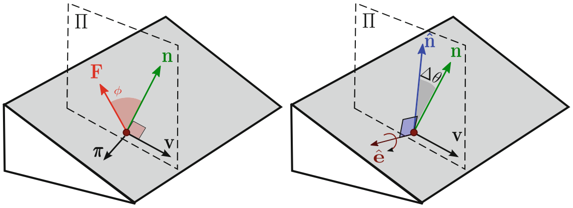


Fig. 2 *Left* vector definitions for slip detection (for a generic foot on a slope). The *red dot* is the foot location, Π is the plane where the ground reaction force **F** and the foot velocity vector **v** lie while **n** is the estimated surface normal. *Right* slip recovery definitions: $\hat{\mathbf{n}}$ is the actual normal used in the controller. $\hat{\mathbf{e}}$ is the axis of rotation to move $\hat{\mathbf{n}}$ towards **n** while $\Delta\theta$ is the correction angle

the friction coefficient μ we perform a moving average with linear weighting (last sample is weighted most).

Observation: the value of μ found with this approach, represents a "sample" of the friction coefficient in a certain *direction*. Depending on the way the friction constraints are implemented in the optimization (e.g. if the friction cone is approximated with an inscribed polyhedron to have linear constraints [25]) an appropriate scaling should be considered. For instance, in the case we approximate the cone with the inscribed pyramid, the estimated μ should be scaled by $1/\sqrt{2}$ which is the ratio between the edge of the inscribed pyramid and the diameter of the cone.

4 Slip Recovery

4.1 Dynamics of Slippage

To obtain insights to draft a strategy for slip recovery it is useful to understand the dynamics of slippage.

1D case: Let us consider the simple case of a mass standing on a plane with friction under the effect of a vertical force (Fig. 3(top)). We can model the contact as a set of tiny bristles [13]. If an external force is applied to the mass which has a tangential component F_{ext_t} the "bristles" at the contact interface start to deform (b) and the friction force with the plane F_f builds up until the breaking point $|\tilde{F}_f| = \mu|F_{ext_n}|$ is achieved, where the bristles start to slide over each other. From then onward they will apply a constant resistance force $\tilde{F}_f$ which is opposing the motion direction (c). The subsequent motion will depend on the mass dynamics ($mdv/dt = F_{ext_t} - \tilde{F}_f$) and any tangential component of F_{ext} will increase the kinetic energy of the body, increasing the slippage. If F_{ext_t} is removed (d), the accumulated momentum will keep the mass in motion but $\tilde{F}_f$ will decelerate it until $v = 0$.

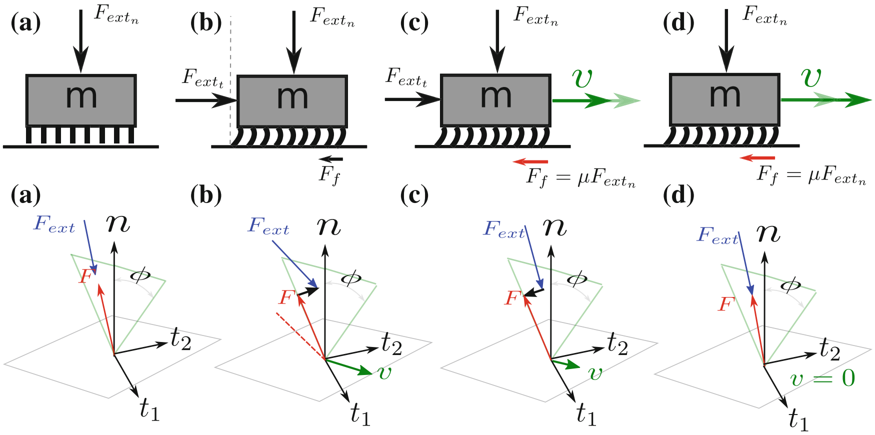


Fig. 3 Slip dynamics: (*top*) 1D case (*bottom*) 3D case

3D case: Consider now the case of a point foot on a frictional plane (Fig. 3(bottom)). In the situation (a), the ground reaction force **F** is able to balance the external force $\mathbf{F}_{ext}$ and the body is in equilibrium ($\mathbf{v} = 0$). If an external load is applied which would require a force which is out of the friction cone to be balanced (b), the ground will be able to balance only with a **F** which is constrained to lie on the boundary of the cone (satisfying the relationship $\|\tilde{\mathbf{F}}_t\| = \mu\|\mathbf{F}_n\|$). The foot will then start moving because there is a net force (black) accelerating it.

Now, as long as $v \neq 0$, **F** will stay on the cone boundary and be opposing the motion. However, if the external force is applied inside the friction cone (c), by the composition of vectors, the net force will have a decelerating component that will slow down the slipping motion until $v = 0$ and the grip will be recovered (d). In this situation $\mathbf{F}_{ext}$ will balance again **F** and the contact will be stable.

4.2 Smooth Correction of Friction Parameters

When slippage occurs some action should be undertaken. The detection phase, illustrated in Sect. 2, has provided the estimated values of **n** and μ. The goal of our *short term* slip recovery strategy is to make the actual surface inclination $\hat{\mathbf{n}}$ (e.g. coming from a terrain estimation algorithm) and the friction coefficient $\hat{\mu}$, which are used in the optimization, converge to the estimated ones. Once that $\hat{\mathbf{n}}$ is set to the appropriate direction (inside the cone), the slippage will naturally end-up after a short transient, because the tangential component of the GRF (see Sect. 4.1) will “make work” against the slipping motion and eventually stop it. To prevent step-wise torque discontinuities, we choose to perform the correction in a smooth fashion. The following recursive equations result in a smooth (1^{st} order) convergence of $\hat{\mathbf{n}}$ toward **n** and of $\hat{\mu}$ toward μ:

$$\begin{aligned} \Delta\theta(k) &= atan\left(\|\hat{\mathbf{n}} \times \mathbf{n}\|/(\hat{\mathbf{n}} \cdot \mathbf{n})\right) \\ \boldsymbol{\omega_\theta}(k) &= \hat{\mathbf{e}}(k) K_n \Delta\theta(k) \\ \hat{\mathbf{n}}(k+1) &= \mathbf{R}(\boldsymbol{\omega_\theta}(k)dt)\hat{\mathbf{n}}(k) \\ \hat{\mu}(k+1) &= K_\mu \mu + (1 - K_\mu)\hat{\mu} \end{aligned} \tag{4}$$

where $\Delta\theta \in \mathbb{R}$ is the angular error between $\hat{\mathbf{n}}$ and $\mathbf{n}$ at time k (see Fig. 2 (right)). $\hat{\mathbf{e}} \in \mathbb{R}^3$ is the instantaneous rotation axis perpendicular to both $\hat{\mathbf{n}}$ and $\mathbf{n}$, and $\mathbf{R}(.) \in \mathbb{R}^{3\times3}$ is the rotation matrix associated to the rotation vector $\boldsymbol{\omega_\theta} dt$, which is obtained by the Rodrigues' formula. K_n and K_w are scalar gains to set the convergence rates of $\hat{\mathbf{n}}$ and $\hat{\mu}$, respectively. dt is the control loop duration.

Comment: It is well known that for a legged robot static/dynamic stability is dependent on the relationship between the CoM/ZMP and the support polygon [27]. In the case the feet are standing on non-coplanar surfaces more elaborated computations should be carried out [4, 5]. In our work we assume that a stabilizing controller is available for the robot. The main goal of our approach is to eliminate slippage in a very short time interval (tens of ms) such that the support polygon does not suffer significant changes and hence the robot stability is not affected. An analysis of the maximum amount of slippage which is tolerable in the context of locomotion in order to preserve stability is out of the scope of this paper and will be part of future works.

4.3 *Freezing Mode*

If, during the slip recovery, the ground frictional force is not sufficient to reduce the slip velocity in a reasonable time (tens of ms) the slipping foot accumulates a significant position error (with respect to the desired set-point). This can likely result in significant degradation of the support polygon shape and possible loss of stability. In this case, the last resort is to stiffen all the joints in the actual configuration and make the robot behave like a "wooden chair" (*freezing mode* in Fig. 1). In such a way, a stable situation can be achieved even if all legs are slipping at the same time. Such a strategy is often successfully adopted by humans when slipping on ice.

5 Simulation Results

In this section we show the effectiveness of the proposed slip detecion/estimation/recovery strategies simulating a walking of the dynamic model of the quadruped HyQ [24] on very slippery surfaces. Namely, an ice slab with locally different frictional properties and a ramp. HyQ is 1 m long and weights 80 kg. Our simulation environment is composed of two software packages. The first, called *SL* [23], is a multi-process application that provides a low level joint controller, a customizable trajectory generator, and a simulator. The robot-specific software, namely the kinematics and dynamics engine, is implemented with *RobCoGen* which provides an optimized C++ implementation of kinematics and dynamics [11] based on spatial-vectors algebra and state-of-the-art numerical algorithms [7]. As far as contact forces are concerned, the SL simulator implements a simple spring damper contact model, together with a Coulomb model for friction. The simulation is based purely on rigid body dynamics, and as such it assumes ideal force sources at the actuated joints. To be consistent with a real implementation on the real robot, we estimate the ground reaction forces at the feet from torque measurements (HyQ is not currently equipped with foot sensors). Both the loop for the optimization and the rigid body simulation run at 1 kHz, which is the frequency of the low level controller in the real platform. The state (position/orientation) of the robot base is estimated through leg odometry. The terrain inclination (roll and pitch) is computed by fitting a plane through the stance feet in a least-square fashion. The evaluation is carried out after each touch-down event and this provides an initial estimate of the surface normal $\hat{\mathbf{n}}_0$ (see Fig. 1).

5.1 Ice Patches

Figure 4(left) shows a simulation of the robot walking on a slippery set of patches located on flat ground. The patches have friction coefficients ($\mu = 0.15 - 0.3$) comparable to the one of an ice-shoe contact [16]. Refer to [19], for different pairs of materials. The robot has point feet and, thus, the friction forces are lower compared a robot which has flat feet. Indeed foot-ground contact in humanoids is usually modeled with 4 contact points located at the foot edges and slippage occurs when all of

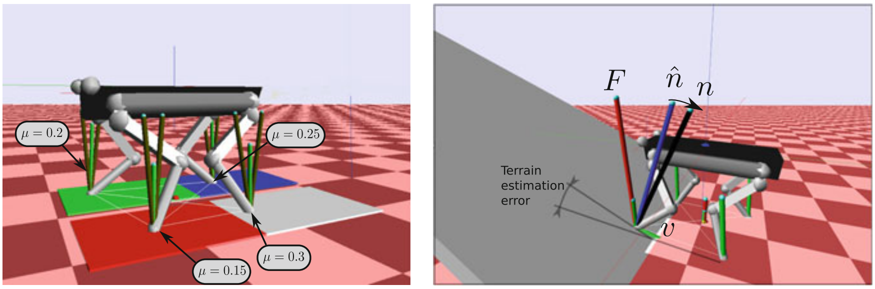
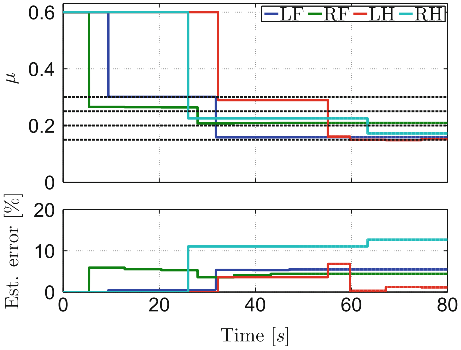


Fig. 4 Simulation screen-shots of the robot walking on ice patches (*left*) and on a ramp (*right*). In the *left plot*, olive *green lines* represent the cone boundaries while the ground reaction forces are depicted in *light green*. In the *right plot* $\hat{n}$ is converging toward n while the LF foot is slipping after the touch-down

Fig. 5 Friction coefficients estimation for the 4 legs in the ice patches simulation. *Upper plot solid lines* are the estimated μ, *dashed lines* are the ground truth. *Lower plot* are the percent estimation errors



them break the contact. For a point foot morphology, walking on ice is challenging and slip recovery becomes crucial for the success of the task.

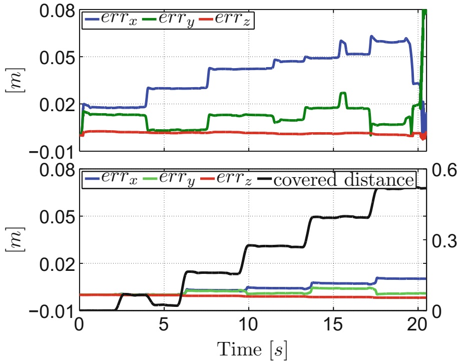
With the *ice patches* simulation we want to show the *robustness* of our algorithm in estimating the friction coefficients of the *different* surfaces for all the 4 feets. The blue/green patches have $\mu = 0.25, 0.2$ while the white/red $\mu = 0.3, 0.15$. They are all 75 cm long. Our online video[2] shows that, without any slip recovery strategy, the robot falls at the very beginning after a few steps. Conversely, with the slip recovery enabled, is able to traverse effectively all the patches, including the last ones which have lower friction coefficients. In Fig. 5 we show the plots of the friction coefficient estimates for the 4 legs starting from $\mu = 0.6$ which is the default value set at the beginning of the simulation, in the controller. LF, RF, LH and RH stand for **L**eft-**F**ront, **R**ight-**F**ront, **L**eft-**H**ind and **R**ight-**H**ind leg, respectively. For the estimate we used a moving average window of 4 samples. The percent error in the estimation is always below 13%.

The slip recovery is beneficial also to avoid the accumulation of big estimation errors in the leg odometry. From the same simulation data, the upper plot of Fig. 6, shows that slippage created an estimation errors in the X direction (before falling) of 18 cm out of 50 cm walked, while using the slip recovery (lower plot) the maximum error is below 1 cm for the same time window.

Observation: In the enclosed video, a little slip is always present at the touch-down, that in principle should not occur, because the friction coefficient has been properly identified after stepping on the surface. This is due to the actual implementation of the stance detection. When the swing foot touches the ground it must apply a force beyond a certain threshold, to trigger the stance and to start optimizing the force. This little force (before the trigger) is not optimized and it causes a little slippage, which however is immediately recovered.

[2] Video available at https://youtu.be/Hrwi9-411AM.

Fig. 6 Plot of the Cartesian components of the odometry error, without (*upper plot*) and with (*lower plot*) slip recovery. The *black line* is the covered distance (right ordinate)



5.2 Slippery Ramp

A transition from walking on flat terrain to a ramp (inclination 0.25 rad) is a good template to demonstrate the effectiveness of the algorithm in estimating the surface normal. Indeed, in the moment in which the robot is standing with only the front feet on the ramp, there is a big error (see Fig. 4 (right)) on the terrain inclination estimate. This results in a wrong estimation of the surface normal $\hat{\mathbf{n}}$ which is set perpendicular to the estimated plane. If the surface is slippery enough (we set $\mu = 0.5$) the front legs will slip and the slip recovery intervention is necessary to climb the ramp. In Fig. 7 we magnify one slip event for the LF leg after the *swing phase*. The slip is detected at time $t = 28$ ms, the estimation phase is shaded in red, while the correction phase is in blue. The slip transient ends at $t = 80$ ms. In the upper plots, we show the convergence of $\hat{\mathbf{n}}$ to the estimated value $\mathbf{n}$ while $\mathbf{n}_{gt}$ is the ground truth. The lower plot shows that the friction cone constraint is violated (in a strict sense) for the whole slippage situation (time interval t =28–80 ms). We underline that the torques of the stance legs (e.g. cf. the knee of RH in Fig. 7(bottom)), do not suffer from step-wise *discontinuity*, because of the smooth correction implemented for $\hat{n}$ and $\hat{\mu}$.

Differently from Fig. 7 which shows the LF foot slipping after a swing phase, Fig. 8 shows a slip occurring during the body motion where the LF leg is in stance. The upper plot shows the velocities (norm) v of the feet while the lower one plots boolean variables that tell which leg is slipping. Around $t = 170$ ms the LF foot starts to slip and this is visible looking at its velocity which significantly differs from the ones of the other feet. The picture also shows that from the point of detection, thanks to the recovery action the slip terminates (the norm of the velocity goes back to the values of the other feet velocities) in less than 40 ms.

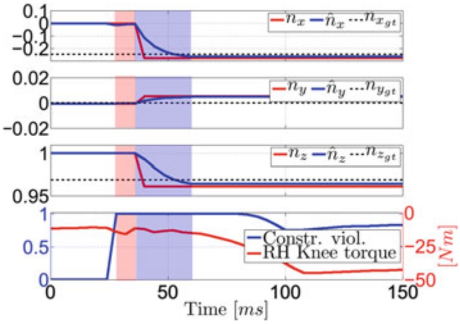
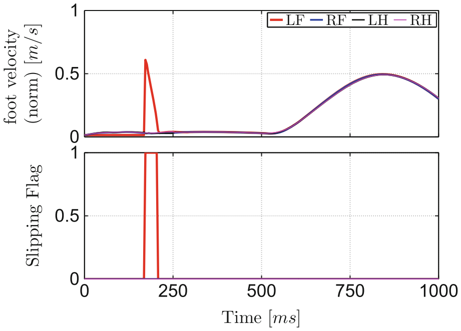


Fig. 7 Slip event for the LF foot at the touch-down (ramp simulation). (First 3 *upper plots*) Cartesian components of the surface normal where: $\hat{\mathbf{n}}$ is the actual value, $\mathbf{n}$ is the estimated value and $\mathbf{n}_{gt}$ is the ground truth (from simulation). *Lower plot* the *blue line* depicts the friction cone violation (in a [0–1] range) while the *red line* is the torque in the knee joint of the RH leg (stance). The estimation phase is shaded in *red*, while the correction phase is in *blue*

Fig. 8 Slip event for the LF leg during the body motion. *Upper* plot of the velocity (norm) v of the feet. *Lower* boolean flags which monitor which leg is slipping



6 Conclusions and Future Works

We presented a methodology to detect slippage and estimate the relevant friction parameters together with a *short term* strategy to recover from slippage *during* locomotion. The detection is based on kinematic measurement (plus the trunk angular velocity) and, in the context of legged robots, is more suitable than a force-based approach which involves the use of 6 axis force/torque sensors at the foot-tips. Having an idea of the friction properties of the terrain during locomotion can be also useful to set different level of “cautiousness”, selecting more or less conservative gaits according to the situation at hand. On the other hand, the recovery strategy (which was able to reduce slippage in less than 40 ms), was implemented at the force level. The idea behind the strategy was to correct the surface normal toward the estimated one resulting in GRFs which were back inside the real friction cone. The slip recovery strategy has been demonstrated to be essential for locomotion on very slippery surfaces, or in situations (ramp) where the terrain inclination was wrongly estimated.

In future works we plan to speed-up the recovery action by setting (in the optimization) constraints on the tangential component on the GRF in order to “help” the frictional force in decelerating the slipping foot. We are aware that with the actual implementation, the estimated friction coefficient can only *decrease*. Indeed, if the robot enters in a less slippery terrain after coming from a slippery one, it will keep the previous friction coefficient which will be too conservative.

In the future, we are planning to fuse the actual approach with semantic information coming from vision. according to the terrain the robot is traversing the purpose of vision is to provide a default value for the friction coefficient together with an estimate of its “difficulty”.

Finally, we plan to perform extensive experimental validation of the proposed approach on the real robot platform (HyQ). In particular we are planning to make it walk on slippery slopes and Teflon patches.

References

1. Abe, Y., da Silva, M., Popović, J.: Multiobjective control with frictional contacts. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 249–258 (2007)
2. Aly, A.a: An antilock-braking systems (ABS) control: a technical review. Intell. Control Autom. **02**(03), 186–195 (2011)
3. Bloesch, M., Hutter, M., Hoepflinger, M., Leutenegger, S., Gehring, C., Remy, C.D., Siegwart, R.: State estimation for legged robots-consistent fusion of leg kinematics and IMU. In: Robotics: Science and Systems (2012)
4. Bretl, T., Lall, S.: Testing static equilibrium for legged robots. IEEE Trans. Robot. **24**(4), 794–807 (2008)
5. Caron, S., Pham, Q.-c., Nakamura, Y.: Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics. In: Robotics: Science and Systems (2015)
6. Chilian, A., Hirschmüller, H., Görner, M.: Multisensor data fusion for robust pose estimation of a six-legged walking robot. In: IEEE International Conference on Intelligent Robots and Systems, pp. 2497–2504 (2011)
7. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, Boston (2008)
8. Feng, S., Xinjilefu, X., Huang, W., Atkeson, C.G.: 3D walking based on online optimization. In: 13th IEEE-RAS International Conference on Humanoid Robots (2013)
9. Focchi, M., Del Prete, A., Havoutis, I., Featherstone, R., Cald-well, D. G., Semini, C.: High-slope terrain locomotion for torque-controlled quadruped robots. Autonomous Robots **41**(1), 259–272. doi:10.1007/s10514-016-9573-1 (2017)
10. Focchi, M., Prete, A., Havoutis, I., Featherstone, R., Caldwell, D.G., Semini, C.: Ground reaction forces control for torque-controlled quadruped robots. In: IEEE International Conference on Intelligent Robots and Systems: Workshop on Whole-Body Control for Robots in the Real World (2014)
11. Frigerio, M., Buchli, J., Caldwell, D.G.: Code generation of algebraic quantities for robot controllers. In: IEEE International Conference on Intelligent Robots and Systems, pp. 2346–2351, October 2012
12. Gramkow, C.: On averaging rotations. J. Math. Imaging Vis. **15**(1–2), 7–16 (2001)
13. Haessig, D.a., Friedland, B.: On the modeling and simulation of friction. In: American Control Conference, pp. 1256–1261 (1990)

14. Herzog, A., Righetti, L., Grimminger, F., Pastor, P., Schaal, S.: Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 981–988. IEEE (2014)
15. Holweg, E., Hoeve, H., Jongkind, W., Marconi, L., Melchiorri, C., Bonivento, C.: Slip detection by tactile sensors: algorithms and experimental results. In: IEEE International Conference on Robotics and Automation, vol. 09, 3234–3239 (1996)
16. Izumi, I., Nakamura, T., Sack, R.L.: In: Snow Engineering: Recent Advances: Proceedings of the Third International Conference, Sendai, Japan, 26–31 May 1996. CRC Press (1997)
17. Klasing, K., Althoff, D., Wollherr, D., Buss, M.: Comparison of surface normal estimation methods for range sensing applications. In: IEEE International Conference on Robotics and Automation, pp. 3206–3211 (2009)
18. Kuindersma, S., Permenter, F., Tedrake, R., Bu, A.: An efficiently solvable quadratic program for stabilizing dynamic locomotion. In: IEEE International Conference on Robotics and Automation, pp. 2589–2594 (2014)
19. Lide, D.R.: CRC Handbook of Chemistry and Physics, vol. 53, 94th edn. CRC Press, Boca Raton (2013)
20. Melchiorri, C.: Slip detection and control using tactile and force sensors. IEEE/ASME Trans. Mechatronics **5**(3), 235–243 (2000)
21. Palli, G., Moriello, L., Scarcia, U., Melchiorri, C.: Development of an optoelectronic 6-axis force/torque sensor for robotic applications. Sensors Actuators A Phys. **220**, 333–346 (2014)
22. Rusu, R.B.: Semantic 3d object maps for everyday manipulation in human living environments. PhDThesis **24**(4), 345–348 (2010)
23. Schaal, S.: The SL simulation and real-time control software package. Technical Report, Accessed Aug 2015 at http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf (2006)
24. Semini, C., Tsagarakis, N.G., Guglielmino, E., Focchi, M., Cannella, F., Caldwell, D.G.: Design of HyQ - a hydraulically and electrically actuated quadruped robot. Proc. Instit. Mech. Eng. Part I J. Syst. Control Eng. **225**(6), 831–849 (2011)
25. Stewart, D., Trinkle, J.: An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In: IEEE International Conference on Robotics and Automation, vol. 1 (2000)
26. Takemura, H., Deguchi, M., Ueda, J., Matsumoto, Y., Ogasawara, T.: Slip-adaptive walk of quadruped robot. Robot. Auton. Syst. **53**(2), 124–141 (2005)
27. Vukobratović, M., Frank, A.A., Juricić, D.: On the stability of biped locomotion. IEEE Trans. Bio-Med. Eng. **17**(1), 25–36 (1970)

Effective Generation of Dynamically Balanced Locomotion with Multiple Non-coplanar Contacts

Nicolas Perrin, Darwin Lau and Vincent Padois

1 Introduction

Locomotion is a challenging task for humanoid robots, even within completely known environments. Although the dynamics of multibody systems are well understood, direct approaches to resolve the motion of all degrees of freedom for locomotion have remained computationally intractable due to the large number of degrees of freedom and nonlinear behaviour. To generate dynamically balanced locomotion trajectories efficiently, current methods are typically based on simplified models of the robot dynamics.

One of the most widespread model is the Inverted Pendulum Model (*IPM*) [10], where the mass of the robot is assumed to be concentrated at its center of mass (*CoM*). Although more accurate models are sometimes required [11], the simple IPM is suitable for many situations where the rotational inertial effects of the robot arms, legs and torso are negligible or can compensate for each other. An example of such a scenario includes walking at a moderately fast pace.

One of the main properties of multi-contact locomotion is that the robot's CoM acceleration depends only on the contact forces with the environment. Hence, an important objective for the generation of locomotion is to design a CoM trajectory that is dynamically balanced at any point in time. *Dynamically balanced* refers to the existence of contact forces that can produce the desired CoM acceleration while respecting the contact constraints, such as being within the friction cones.

N. Perrin (✉) · D. Lau · V. Padois
Institut des Systèmes Intelligents et de Robotique (ISIR), CNRS UMR 7222,
Université Pierre et Marie Curie, Paris, France
e-mail: perrin@isir.upmc.fr

D. Lau
e-mail: lau@isir.upmc.fr

V. Padois
e-mail: padois@isir.upmc.fr

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_12

When all the contacts are coplanar, a point of particular interest, sometimes called the Zero Moment Point (*ZMP*) [17], can be defined. Although this name has led to confusion about its physical nature [3], the term ZMP will be used in the remainder of the work. A system is considered dynamically balanced if the ZMP lies within the support region, i.e. the 2D convex hull of the contact points. In the IPM, the relationship between the CoM and ZMP dynamics is defined by nonlinear differential equations. However, if the vertical displacements of the CoM are set in advance, the relationship can be decoupled and expressed as linear differential equations. This substantially simplifies the problem into one that can be more efficiently and easily implemented for locomotion trajectory generation and gait control.

The trajectory generation problem has been studied through analytical approaches [4], and more recently, using convex optimization with constraints on the ZMP in discrete time [6, 7, 19, 20]. The efficiency of convex optimization solvers allows them to be used within model predictive control (*MPC*) schemes and with the potential of real-time implementation for reactive walking. However, the ZMP+MPC approach suffers from two main drawbacks: (1) the trajectory of the CoM height (z-direction) must be known or fixed in advance, and (2) the contact points must always be coplanar, making this approach unsuitable for walking in complex unstructured environments or if the arms are to also be used. Due to these two restrictions, the ZMP+MPC approach can be regarded as a 2D CoM planner that operates in the horizontal xy-plane only.

Several studies have looked into extending the concept of ZMP into 3D conditions, such as GZMP [5] and 3DZMP [9], to handle non-coplanar contact points. These criteria have been used in control algorithms to maintain the dynamical balance of a humanoid robot interacting with its environment. However, no locomotion trajectory generation algorithm has been designed based on these notions. In [8], a more general criterion than the ZMP was proposed to evaluate the balance of contacts during the motion of a legged robot. This criterion was used within a preview control algorithm with additional restrictions, for example, the vertical motion of the CoM must be approximately constant. Furthermore, a preliminary phase is needed to plan the inertia and gravity wrenches appropriately, which is a difficult problem.

In this paper, two simple and novel MPC approaches to solve for 3D locomotion with multiple non-coplanar contacts are presented. The 3D condition for dynamically balanced gait allows for non-coplanar multiple contacts and no restrictions on the CoM height trajectory. Using the proposed 3D dynamically balanced criterion, the first MPC formulation treats the criterion as an objective function, where the resulting non-convex MPC problem is solved using a sequence of alternating (convex) quadratic programs (*AQP*). The second formulation considers the criterion as non-convex constraints to the problem, and is solved through a succession of convex QCQPs. The results for simple locomotion scenarios show the promise in using the proposed 3D condition to generate the CoM trajectory within the control framework of robots with multiple contacts.

Although generalizations, such as allowing for multiple contacts, non-coplanar contacts and not predetermining height trajectory, inherently increase the problem complexity, care has been taken to maintain balance between the computational efficiency and limitations of the model. Indeed, our approach is more general than traditional simplified methods based on the IPM or ZMP, but thanks to our main assumption, namely, that the CoM lies inside all the contact friction cones, the number of variables that have to be taken into account is much smaller than with direct methods, such as [15]. The feature of the proposed MPC schemes is that they can be effectively and efficiently solved through a succession of convex optimization problems, when the problem formulation is of particular forms, such as bilinear problems or non-convex QCQPs.

The remainder of the paper is organised as follows: Sect. 2 formulates the 3D condition for dynamically balance. The two MPC formulations using the 3D condition as an objective or as constraints are formulated in Sects. 3 and 4, respectively. Section 5 presents a brief discussion on the two approaches. Finally, Sect. 6 concludes the paper and presents areas of future work.

2 Conditions for Dynamically Balanced Locomotion in 3D

Expressing wrenches with respect to the CoM of the robot $\mathbf{x}$, the Newton-Euler equations of motion for a multibody robot system in a fixed world frame can be written as

$$\mathbf{W}_x^{gravity} + \mathbf{W}_x^{contact} = \begin{bmatrix} \dot{\mathbf{L}} \\ M\ddot{\mathbf{x}} \end{bmatrix}, \tag{1}$$

where $\mathbf{W}_x^{gravity}$ is the gravity wrench in 6D vector notation [2], $\mathbf{W}_x^{contact}$ the sum of the contact wrenches, $\mathbf{L}$ is the angular momentum of the whole robot with respect to its CoM, and M the total mass of the robot. The gravity wrench is equal to the vector $\begin{bmatrix}\mathbf{0}^T & M\mathbf{g}^T\end{bmatrix}^T$, where $\mathbf{g}$ is the gravity vector, and

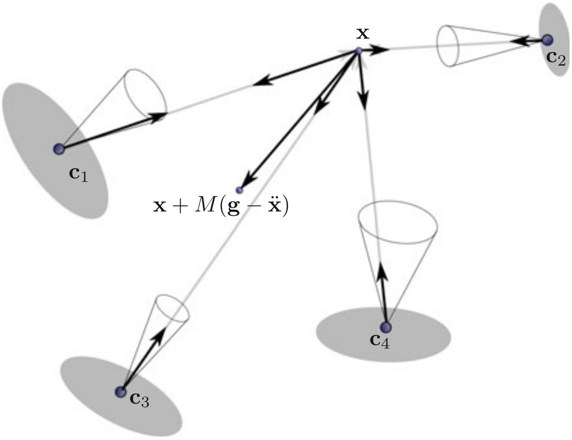
$$\mathbf{W}_x^{contact} = \sum_j^N \begin{bmatrix} (\mathbf{c}_j - \mathbf{x}) \times \mathbf{f}_j \\ \mathbf{f}_j \end{bmatrix}, \tag{2}$$

where the $\mathbf{c}_j$ and $\mathbf{f}_j$ are the location and force of contact j, respectively, and N is the total number of contacts. Substituting (2) into (1) results in

$$\sum_j \begin{bmatrix} (\mathbf{c}_j - \mathbf{x}) \times \mathbf{f}_j \\ \mathbf{f}_j \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{L}} \\ M(\ddot{\mathbf{x}} - \mathbf{g}) \end{bmatrix}. \tag{3}$$

The system can be regarded as *dynamically balanced* if (3) is satisfied. The goal of the problem of CoM trajectory generation is to compute $\ddot{\mathbf{x}}(t)$ such that the *dynamic wrench*, the right hand side term of (3), can be compensated by the sum of contact force wrenches. It is usually assumed that the contact forces belong to a cone (*friction cone*) spanning from the contact point.

Fig. 1 All the contact cones contain the direction towards the position of the robot CoM **x**. Choosing the force of contact j equal to $\alpha_j(\mathbf{c}_j - \mathbf{x})$, $\alpha_j \geq 0$, yields: $M(\mathbf{g} - \ddot{\mathbf{x}}) = \sum_j \alpha_j(\mathbf{c}_j - \mathbf{x}),\ \alpha_j \geq 0$



Generally, it is not trivial to compute the set, or even a reasonable subset, of the dynamic wrenches that can be compensated by a given set of contact cones [16]. However, by assuming that all contact cones contain the CoM **x**, as shown in Fig. 1, it is possible to choose the force of contact j to be

$$\mathbf{f}_j = \alpha_j(\mathbf{x} - \mathbf{c}_j)\ , \alpha_j \geq 0\ . \tag{4}$$

This assumption is equivalent to enforcing the constraint $\mathbf{x} - \mathbf{c}_j \in F_j$, where F_j is the friction cone for the contact point j. Given the contact point $\mathbf{c}_j$, normal vector to the contact surface $\hat{\mathbf{n}}_j$ and the coefficient of friction μ_j, this constraint can be expressed as a second order cone constraint or approximated by a set of linear constraints.

By this selection, substituting (4) into the criterion for dynamic balance (3) results in

$$\begin{bmatrix} \mathbf{0} \\ \sum \alpha_j(\mathbf{x} - \mathbf{c}_j) \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{L}} \\ M(\ddot{\mathbf{x}} - \mathbf{g}) \end{bmatrix}\ . \tag{5}$$

From (5), the system is dynamically balanced if:

1. The angular momentum $\dot{\mathbf{L}}$ of the whole robot with respect to its CoM is negligible.
2. $M(\mathbf{g} - \ddot{\mathbf{x}})$ can be expressed by a positive linear combination of the vectors $\mathbf{c}_j - \mathbf{x}\ , j = 1, \ldots, N$.

It is worth noting that the proposed condition (5) holds in 3D without the restriction that contacts are coplanar. Furthermore, although the assumption $\mathbf{x} - \mathbf{c}_j \in F_j$ limits range of scenarios in which the simplified conditions for dynamic balance (5) can be used, it is important to note that the restrictions are less limited and can be applied in a larger range of scenarios compared with Zero Moment Point (ZMP) conditions.

3 Model Predictive Control with Dynamically Balancing Objective

3.1 Problem Formulation: Non-convex Optimization Problem

The aim of this problem is to determine the CoM trajectory for the locomotion of a multi-limbed robot for a given sequence of contacts over a finite time horizon. Although the number of contacts between the robot and the environment can theoretically be infinite, without loss of generality it can be supposed that only a finite number of points N of the robot can be in contact with the environment. For example, the contact points for a rectangular foot can be represented by the 4 vertices of the foot sole. The positions of these time-varying contact points can be denoted by $\mathbf{c}_1(t), \mathbf{c}_2(t), \ldots, \mathbf{c}_N(t)$. A binary variable $\kappa_j(t) \in \{0, 1\}$ can be defined to denote whether $\mathbf{c}_j(t)$ is in contact ($\kappa_j(t) = 1$) or not in contact ($\kappa_j(t) = 0$) with the environment at time t. In this work, the contact trajectory $\mathbf{c}_j(t)$ and $\kappa_j(t)$ are predetermined.

In this proposed model predictive control (*MPC*) scheme, the *decision variables* are the CoM positions $\mathbf{x}(t)$ and the contact force multipliers $\alpha_j(t)\ \forall j \in \{1, \ldots, N\}$, over the finite time horizon T, where $t \in [t_0, t_0 + T]$. Discretizing the horizon at a time step of δt results in K discrete time instances, where $T = K\delta t$ and $t_i = t_0 + i \cdot \delta t$. Given the contact trajectory information $\mathbf{c}_j(t)\ \forall j \in \{1, \ldots, N\}$ and $\kappa_j(t)\ \forall j \in \{1, \ldots, N\}$, the *objective* is to satisfy the translational components of (5) by minimising

$$J_1 = \sum_{i=1}^{K} \left\| M(\mathbf{g} - \ddot{\mathbf{x}}(t_i)) - \sum_{j}^{N} \kappa_j(t_i)\alpha_j(t_i)(\mathbf{c}_j(t_i) - \mathbf{x}(t_i)) \right\|^2 .$$

The *constraints* for this problem are:

- Initial/current CoM position $\mathbf{x}(t_0) = \mathbf{x}_0$
- Initial/current CoM velocity $\dot{\mathbf{x}}(t_0) = \dot{\mathbf{x}}_0$
- Final CoM position at the end of the horizon $\mathbf{x}(t_K) = \mathbf{x}_f$
- Max acceleration $\|\ddot{\mathbf{x}}(t_i)\|_\infty \leq a_{max},\ i = 0, \ldots, K-1$

Note that any constraints on the CoM velocity $\dot{\mathbf{x}}$ and acceleration $\ddot{\mathbf{x}}$ can be expressed with respect to $\mathbf{x}$ by the linear relationships

$$\dot{\mathbf{x}}(t_i) = \frac{\mathbf{x}(t_i) - \mathbf{x}(t_{i-1})}{\delta t},\ i = 1, \ldots, K \tag{6}$$

$$\ddot{\mathbf{x}}(t_i) = \frac{\dot{\mathbf{x}}(t_{i+1}) - \dot{\mathbf{x}}(t_i)}{\delta t},\ i = 0, \ldots, K-1 . \tag{7}$$

As a result, all constraints are either linear equality or inequality constraints with respect to $\mathbf{x}$. Thus, the MPC optimization problem to determine the CoM trajectory over the finite time horizon can be expressed as

$$\begin{aligned}
\min_{(\mathbf{x}(t_i))_i,\left(\alpha_j(t_i)\right)_{i,j}} \quad & J_1\left((\mathbf{x}(t_i))_i\,,\,\left(\alpha_j(t_i)\right)_{i,j}\right) \\
\text{s.t.} \quad & \|\ddot{\mathbf{x}}(t_i)\|_\infty \leq a_{max}\,,\ i = 0, \ldots, K-1 \\
& \mathbf{x}(t_0) = \mathbf{x}_0 \\
& \dot{\mathbf{x}}(t_0) = \dot{\mathbf{x}}_0 \\
& \mathbf{x}(t_K) = \mathbf{x}_f\ .
\end{aligned} \tag{8}$$

Remark: As it is presented here, to implement the approach as a model predictive controller would require measuring the CoM velocity ($\dot{\mathbf{x}}_0$), which is not always easy. If the contact forces are accessible, one could estimate their total wrench and use it to obtain an approximation of $M(\mathbf{g} - \ddot{\mathbf{x}})$, which could potentially be used to make the approach more robust.

3.2 *An Algorithm Exploiting the Objective Function Structure*

On a computational level, the objective function J_1 is non-convex and it is therefore difficult to obtain the global optimum. Finding a local minimum could be relatively quick and provide good results, but in the next section we propose another approach that exploits the form of the objective function. The alternating convex optimization approach is inspired by [12], where successful results for a different application are obtained by using an alternating sequence of convex QPs (*AQPs*) instead of trying to solve head-on an optimization problem with a bilinear objective function. The structure of function J_1 can be expressed as

$$J_1 = \sum_i \|\epsilon_i\left(\mathbf{x}(t_i), (\alpha_j(t_i))_j\right)\|^2\ , \tag{9}$$

with

$$\epsilon_i = M(\mathbf{g} - \ddot{\mathbf{x}}(t_i)) - \sum_j^N \kappa_j(t_i)\alpha_j(t_i)(\mathbf{c}_j(t_i) - \mathbf{x}(t_i)).$$

The functions ϵ_i are bilinear in the variables $\mathbf{x}(t_i)$ and $\alpha_j(t_i)$. Therefore, if $\mathbf{x}(t_i)$ are fixed, the optimization problem (8) becomes a convex QP that can be efficiently solved. Similarly, if the $\alpha_j(t_i)$ variables are fixed, we obtain a convex QP in the $\mathbf{x}(t_i)$ variables. This property can be used to alternatively optimize the $\mathbf{x}(t_i)$ and $\alpha_j(t_i)$ variables, using the solution of each step as the fixed variables of the next step. This ensures that the objective J_1 decreases at each iteration of the algorithm until it converges.

For the first step, an initial guess was made for the CoM trajectory that is consistent with the constraints from (8), and the $\alpha_j(t_i)$ variables are optimized. The most direct motion from $\mathbf{x}_0$ to $\mathbf{x}_f$ is often a natural candidate for this initial guess, so in practice this method does not require any tuning at all.

3.3 Results

The RobOptim [14] framework was used to illustrate the use of the proposed MPC formulation (8) and alternating quadratic programs that exploit the bilinear form (9) on CoM generation for locomotion. RobOptim provides a convenient interface to try various optimization tools. One interesting aspect of the formulation of the proposed MPC is that the matrices describing the constraints are very sparse, with $O(m)$ non-zero elements where m is the total number of variables. Thus, we chose the IPOPT optimizer [18] that can exploit matrix sparsity. Three different scenarios are simulated and presented:

1. 3 step walk with two foot supports (coplanar contacts);
2. 3 step walk with two foot supports and one hand support (non-coplanar contacts);
3. Jump-step with flight phase.

The 4.5 s trajectory for scenario 1 can be defined by the following sequence as shown in Fig. 2:

- $0 \leq t < 0.6$ s: Robot initially begins in double support
- $0.6 \leq t \leq 1.2$ s: Step 1 moving the left foot
- $1.2 < t < 1.9$ s: Double support phase
- $1.9 \leq t \leq 2.5$ s: Step 2 moving the right foot
- $2.5 < t < 3.2$ s: Double support phase
- $3.2 \leq t \leq 3.8$ s: Step 3 moving the left foot
- $3.8 < t \leq 4.5$ s: Double support phase.

With a time horizon of 4.5 s and $\delta t = 0.1$ s, the MPC scheme consists of 45 discrete steps. Without loss of generality, it should be noted that the MPC in this setup solves for the entire trajectory, where in practice the time horizon would be

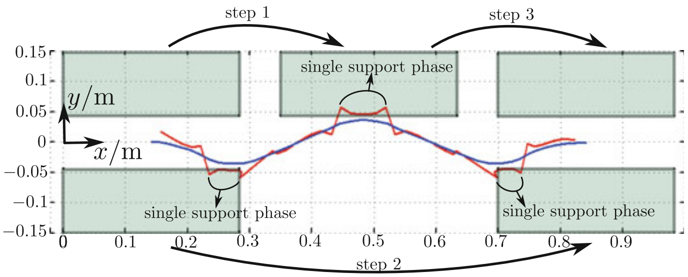


Fig. 2 The resulting trajectory from the MPC scheme (8) for scenario 1. In *blue*, the horizontal CoM trajectory produced by our algorithm in 95 ms. In *red*, the projected ZMP trajectory (10)

shorter to achieve better computational performance. The dimensions of the feet 10.45 cm × 28.3 cm and the total mass 36.66 kg correspond to the Romeo robot [1]. At $t = 0$ s and $t = 4.5$ s, the CoM is set to be in the middle between the feet at a height of 0.6 m, $\mathbf{x}(0) = [0.14 \ \ 0 \ \ 0.6]^T$ and $\mathbf{x}(4.5) = [0.85 \ \ 0 \ \ 0.6]^T$, respectively. The initial guess for the CoM trajectory follows a straight line from its initial to its final position. Figure 2 shows the CoM motion generated after two steps of the algorithm (i.e. one optimization of the $\alpha_j(t_i)$ variables and then one optimization of the CoM trajectory).

Figure 2 shows the resulting CoM trajectory (in blue) and the projection (in red)

$$\mathbf{x}_{zmp} = \lambda_{zmp}(\mathbf{x} + M(\mathbf{g} - \ddot{\mathbf{x}})), \tag{10}$$

where $\lambda_{zmp} > 0$ is a scaling factor such that $z_{zmp} = 0$. The projected point (10) is equivalent to the ZMP point for the scenario 1 in which all contact points are always coplanar. It can be observed that the projected ZMP lies within the foot during single support phases. In this scenario, the results are similar to that of the ZMP+MPC approach, since all contacts are coplanar on the ground plane. For this scenario, the optimization problem (8) with horizon of K steps consists of $11K$ decision variables, comprised of 3 for the CoM position and 8 for the two contact supports (four contact points per support) at each time step, and 9 + K constraints from (8). The trajectory was computed in 95 ms (on a 2.40GHz Intel(R) Core(TM) i7-4700MQ CPU) after solving two alternations of the AQP, and hence 4 convex QPs. Executing more alternations of the algorithm showed that the cost J_1 quickly reached small values, for example, more than 1000 times smaller than the initial value after 4 optimizations. Furthermore, it was observed that the algorithm had almost converged after the 2 first optimizations.

In the second scenario, an additional contact corresponding to the right hand of the robot on a wall is considered. The contact point is at a height of 0.6 m, and activated only during the second step. This example demonstrates the ability of the proposed 3D condition and MPC algorithm to handle non-coplanar contacts. The resulting trajectory in Fig. 3 shows how the proposed algorithm can handle this situation and generated a different CoM trajectory. It could be observed from the results that the additional hand contact point enabled the robot to avoid the sway motion to the left.

Finally, the third scenario illustrates the ability of the algorithm to determine the CoM height without restrictions unlike the classical ZMP+MPC approach. The steps were replaced by one jump defined by the following sequence of actions: at $t = 0.6$ s the left foot leaves the ground, and at $t = 2.0$ s the right foot leaves the ground. After a flight phase of 0.4 s, the left foot lands at $t = 2.4$ s, and then the right foot lands at $t = 3.8$ s. Remark: in this scenario the a_{max} bound must be at least equal to the norm of the gravitational acceleration. Figure 4 shows the CoM trajectory produced after 2 optimizations. This example clearly shows the benefits in relaxing the fixed height trajectory from the ZMP+MPC approach, such that the CoM trajectory generation is able to produce a jump motion in the z-direction.

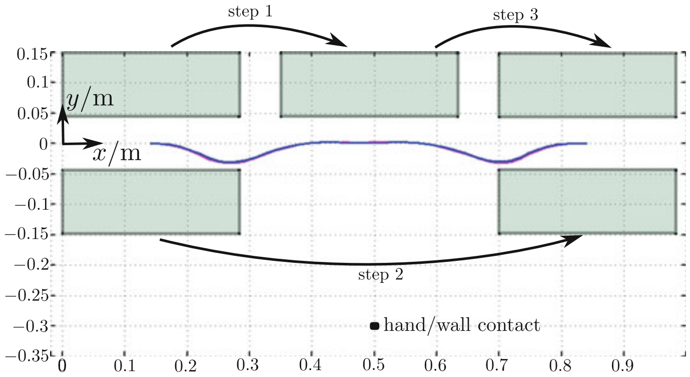


Fig. 3 The resulting trajectory from the MPC scheme (8) for scenario 2 with hand support. In *blue*, the horizontal CoM trajectory produced after 2 steps of the algorithm (in 96 ms). In *purple*, the trajectory produced after 6 steps of the algorithm (in 412 ms). These trajectories are almost exactly the same, which shows that the convergence is fast

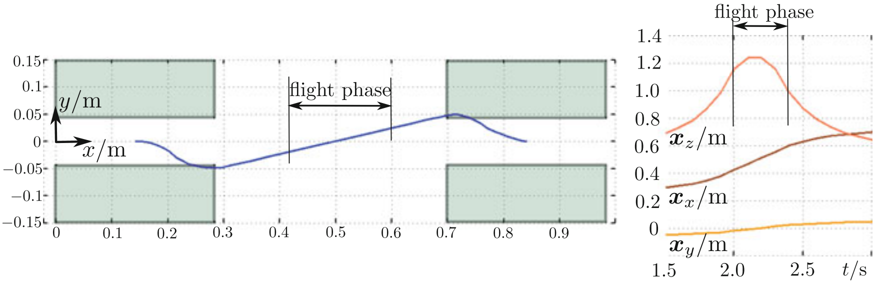


Fig. 4 *On the left* the horizontal CoM trajectory generated (in 176 ms) for the jump scenario. *On the right* the evolution of the CoM coordinates during the flight phase

4 Model Predictive Control with Dynamically Balancing Constraints

4.1 Problem Formulation: Non-convex Quadratically Constrained Quadratic Program

With the same aim of determining the CoM trajectory as Sect. 3, the problem in this section will be formulated by considering the dynamically balanced condition (5) as constraints over the time horizon. Furthermore, the objective to minimise is the tracking error for a reference CoM trajectory $\mathbf{x}_r(t)$. The reference CoM trajectory is pre-planned based on the desired motion. In this MPC problem, the *decision variables* are the CoM jerk vectors $\dddot{\mathbf{x}}(t_i)$, $\forall i \in \{1, \ldots, K\}$. The convex *objective* of the problem is to minimise

$$J_2 = \sum_{i=1}^{K} Q_1 \left\| \mathbf{x}(t_i) - \mathbf{x}_r(t_i) \right\|^2 + Q_2 \left\| \dddot{\mathbf{x}}(t_i) \right\|^2 \ ,$$

where Q_1 and Q_2 are the relative weights between the tracking and jerk terms, respectively. Note that the relationship between the CoM jerk $\dddot{\mathbf{x}}$ and the CoM position $\mathbf{x}$, velocity $\dot{\mathbf{x}}$ and acceleration $\ddot{\mathbf{x}}$ is linear [19] in the form

$$\begin{bmatrix} \mathbf{x}(t_{i+1}) \\ \dot{\mathbf{x}}(t_{i+1}) \\ \ddot{\mathbf{x}}(t_{i+1}) \end{bmatrix} = A \begin{bmatrix} \mathbf{x}(t_i) \\ \dot{\mathbf{x}}(t_i) \\ \ddot{\mathbf{x}}(t_i) \end{bmatrix} + B \dddot{\mathbf{x}}(t_i) \ .$$

As a result, the objective and all constraints can be expressed with respect to the CoM jerk. As presented in Sect. 2, the geometrical meaning of the constraint

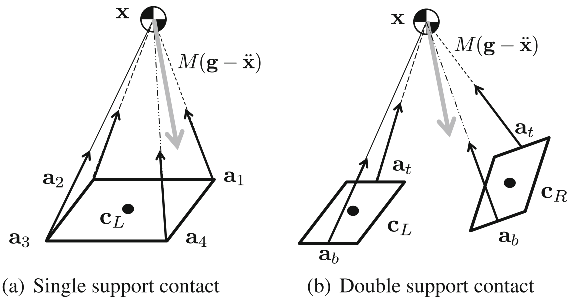
$$\sum \alpha_j (\mathbf{c}_j - \mathbf{x}) = M(\mathbf{g} - \ddot{\mathbf{x}}) \ , \alpha_j \geq 0 \tag{11}$$

is that the vector $M(\mathbf{g} - \ddot{\mathbf{x}})$ must lie within the positive cone of the vectors $(\mathbf{c}_j - \mathbf{x})$ $\forall j \in \{1, \ldots, N\}$. Such constraints will be derived in the following for the case of one and two supports in contact.

As shown in Fig. 5a, when the system is in single support it will be assumed that there are four contact points. The position of the vertices for the rectangular contact surface relative to the center of the contact $\mathbf{c}_j$ can be represented by $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ and $\mathbf{a_4}$. For the single support, the vector $M(\mathbf{g} - \ddot{\mathbf{x}})$ is inside the cone produced by the vectors $(\mathbf{c}_j + \mathbf{a}_1 - \mathbf{x}, \mathbf{c}_j + \mathbf{a}_2 - \mathbf{x}, \mathbf{c}_j + \mathbf{a}_3 - \mathbf{x}, \mathbf{c}_j + \mathbf{a}_4 - \mathbf{x}) = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4)$ if the following triple products are positive:

$$\begin{aligned} \mathbf{w}_1 \times \mathbf{w}_2 \cdot (\mathbf{g} - \ddot{\mathbf{x}}) \geq 0, \quad \mathbf{w}_2 \times \mathbf{w}_3 \cdot (\mathbf{g} - \ddot{\mathbf{x}}) \geq 0 \\ \mathbf{w}_3 \times \mathbf{w}_4 \cdot (\mathbf{g} - \ddot{\mathbf{x}}) \geq 0, \quad \mathbf{w}_4 \times \mathbf{w}_1 \cdot (\mathbf{g} - \ddot{\mathbf{x}}) \geq 0 \ . \end{aligned} \tag{12}$$

For the case of double support contacts, the expression of the convex cone would be non-trivial if each support was assumed to consist of 4 contact points. This is because the convex cone would be dependent on the locations of each support.



(a) Single support contact (b) Double support contact

Fig. 5 The constraints required to achieve dynamic balance for: **a** when one support is in contact with the environment $\kappa_L = 1, \kappa_R = 0$ and **b** when two supports are in contact $\kappa_L = 1, \kappa_R = 1$

Furthermore, this would increase the number of surfaces of the convex cone for the vector $M(\mathbf{g} - \ddot{\mathbf{x}})$ to be checked, hence increasing the number of constraints. However, by assuming that each contact support only has two contact points, at the top $\mathbf{a}_t$ and bottom $\mathbf{a}_b$, then the resulting convex cone shown in Fig. 5b only consists of four surfaces and is a strict subset of the 4 contact point convex hull regardless of the support location. For the double support, the vector $M(\mathbf{g} - \ddot{\mathbf{x}})$ is inside the cone produced by the vectors $(\mathbf{c}_L + \mathbf{a}_t - \mathbf{x}, \mathbf{c}_L + \mathbf{a}_b - \mathbf{x}, \mathbf{c}_R + \mathbf{a}_b - \mathbf{x}, \mathbf{c}_R + \mathbf{a}_t - \mathbf{x}) = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4)$ as with the conditions in (12). In a similar manner, constraint equations for extra scenarios with additional contacts could be considered. For both mathematical and practical considerations, it will be assumed that contacts do not overlap, such that the contact points in Fig. 5b can form a convex cone.

It can be observed that the constraints (12) for both single and double support are non-convex quadratic inequality constraints if the contact trajectories are known. As a result, the MPC optimization problem to determine the CoM trajectory over the finite time horizon can be expressed as

$$\begin{aligned} \min_{(\dddot{\mathbf{x}}(t_i))_i} \quad & \sum_{i=1}^{K} Q_1 \, \|\mathbf{x}(t_i) - \mathbf{x}_r(t_i)\|^2 + Q_2 \, \|\dddot{\mathbf{x}}(t_i)\|^2 \\ \text{s.t.} \quad & (12) \text{ for } \kappa_j = 1, \ \kappa_a = 0 \ \forall a \neq j \\ & (12) \text{ for } \kappa_L = 1, \kappa_R = 1 \, . \end{aligned} \tag{13}$$

The optimization problem (13) is a non-convex quadratically constrained quadratic program (*QCQP*) with a convex objective function.

4.2 Feasible Point Pursuit Successive Convex Approximation

On a computational level, the objective function J_2 has non-convex constraints and therefore it is non-trivial to obtain a feasible solution, let alone the global optimum. The feasible-point-pursuit successive convex approximation (*FPP-SCA*) [13] is an effective approach that solves the non-convex QCQP as a succession of QP convex approximations. Considering the non-convex QCQP of the standard form with the decision variable $\mathbf{u} \in \mathbb{R}^n$

$$\begin{aligned} \min_{\mathbf{u}} \quad & \mathbf{u}^T A_0 \mathbf{u} + \mathbf{q}_0^T \mathbf{u} \\ \text{s.t.} \quad & \mathbf{u}^T A_k \mathbf{u} + \mathbf{q}_k^T \mathbf{u} \leq b_k, \ k = 1, \ldots, L \, , \end{aligned} \tag{14}$$

where $A_0 \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix, if any of $A_k \in \mathbb{R}^{n \times n}$ from the L constraints are not positive semidefinite, then (14) is non-convex and the problem is NP-hard in general. By performing a *linear restriction* about any point $\mathbf{z}_i$, the FPP-SCA approach aims to solve the following problem

$$\begin{aligned} \min_{\mathbf{u},\mathbf{s}} \quad & \mathbf{u}^T A_0 \mathbf{u} + \mathbf{q}_0^T \mathbf{u} + \lambda \sum_k s_k^2 \\ \text{s.t.} \quad & \mathbf{u}^T A_k^+ \mathbf{u} + (2\mathbf{z}_i^T A_k^- + \mathbf{q}_k^T)\mathbf{u} \le b_k + \mathbf{z}_i^T A_k^- \mathbf{z}_i + s_k \;, \\ & s_k \ge 0, \;\; k = 1, \ldots, L \end{aligned} \tag{15}$$

where A_k^+ and A_k^- are the positive semidefinite and negative semidefinite matrices from the decomposition $A_k = A_k^+ + A_k^-$. The terms s_k are slack variables that represent the violation of the k-th quadratic constraint and $\lambda \gg 1$ is a constant that gives priority to minimise the constraint violation. The vector $\mathbf{z}_i \in \mathbb{R}^n$ is the initial guess vector at iteration i. In the FPP-SCA approach, the convex QCQP (15) is repeated, where the variable $\mathbf{z}_{i+1}$ is set as the optimal solution $\mathbf{u}_i^*$ at iteration i.

By using the FPP-SCA approach to solve the MPC problem (13) with horizon of K steps, at each iteration of FPP-SCA the convex QCQP problem consists of 7 K decision variables, comprised of 3 for the CoM jerk and 4 constraint slack variables at each time step, and 4 K quadratic constraints (12) at each time step. The initial guess $\mathbf{z}_0$ can be any arbitrary vector, and in the simulations the zero vector was chosen. By solving a succession of (15), it was shown in [13] and in the results of Sect. 4.3 that the algorithm converges within a few successions.

4.3 Results

To demonstrate the MPC formulation (13), the CoM jerk trajectories were determined using the FPP-SCA for the three following scenarios: scenario 1 from Sect. 3.3, scenario 2 from Sect. 3.3, and walking up 2 steps of a staircase with the same hand support as in scenario 2 from Sect. 3.3. Scenarios 1 and 2 allow for a direct comparison between the trajectories resulting from the two different MPC schemes. The number of successive QCQPs solved was set as 5, however it was observed that convergence typically happened in less than 3 successions. The reference trajectory for both scenarios was set to be simply the forward linear motion of travel at a height of 0.6 m. The values for objective function weights in (13) and (15) were set as $Q_1 = 1$, $Q_2 = 10^{-4}$ and $\lambda = 10^4$.

The trajectory for scenario 1 is shown in Fig. 6. From the projected ZMP points using (10) it is clear that the produced CoM trajectory satisfies the 3D dynamically balanced criterion. To maintain good tracking performance of the reference trajectory for $y = 0$, it is expected that the projected ZMP would be as close to the boundary of the single stance (SS) support region as possible. It can be observed that this MPC scheme generated very similar CoM trajectory results from Fig. 2.

As with Fig. 2, the natural artifact of left-right swaying motion can be observed in Fig. 6 due to the existence of single support instances. As such, a hand contact was included (scenario 2) during step 2 of the motion. From the resulting trajectories shown in Fig. 7 (on the left), the MPC scheme generated a very similar, in fact near identical, behaviour to that in Fig. 3. The extra hand contact allowed the CoM to

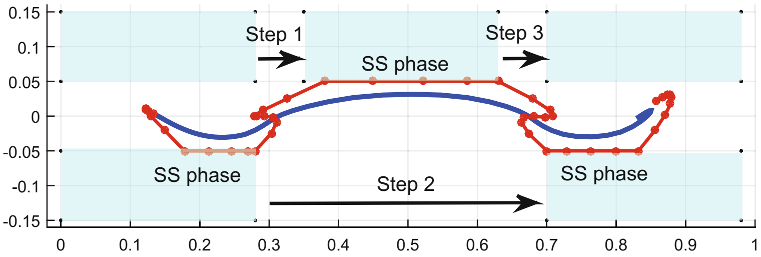


Fig. 6 The resulting trajectory from the MPC scheme (13) for scenario 1. In *blue*, the horizontal CoM trajectory and in *red*, the projected ZMP trajectory (10) since all contacts are coplanar

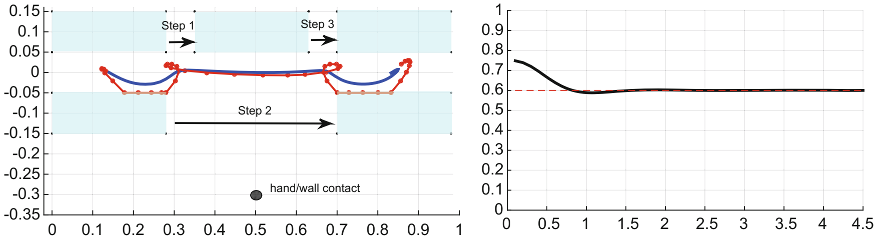


Fig. 7 *On the left* the resulting trajectory from the MPC scheme (13) for scenario 2 with hand support. In *blue*, the horizontal CoM trajectory and in *red*, the projected ZMP trajectory (10). It can be observed that the results are similar to that from Fig. 3. *On the right* the resulting z-direction trajectory from the MPC scheme (13) for scenarios 1 and 2. In *black*, the CoM trajectory and in *red*, the reference height trajectory. The results show that the trajectory successfully lowers from the initial height $z_0 = 0.75$ m to the reference height 0.6 m

better track the reference trajectory as there is no single support phase sway required during step 2. To show the tracking of the height trajectory, the initial height of the CoM was set to be 0.75 m. Figure 7 (on the right) shows that the MPC CoM planner was able to quickly converge to the desired reference height of 0.6 m. As such, it can be claimed that for this scenario, the FFP-SCA formulation was able to achieve both feasibility of the dynamically balancing constraints and good tracking performance.

Finally, scenario 3 demonstrates the ability for the proposed MPC formulation to solve more complex behaviours, such as walking up stairs while using the hand to hold onto the staircase rails. The steps of the stairs were set as 15 cm high and the reference height trajectory for the CoM was set to be 0.6 m above the surface of stair steps. Figure 8 (on the left) shows the x and y direction (top-down view) CoM trajectories determined by the MPC. It can be observed that the CoM results for the x and y directions are very similar to that of walking on a flat ground as shown in Fig. 7 (on the left). However, the resulting z trajectory shown in Fig. 8 (on the right) shows significant differences with scenario 2. In addition to satisfying the dynamically balanced constraints for the locomotion on the non-coplanar contacts,

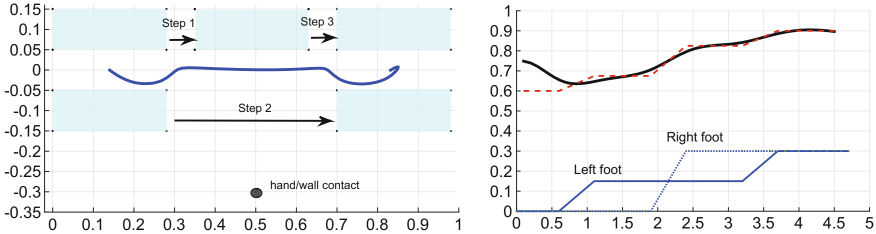


Fig. 8 *On the left* the resulting trajectory from the MPC scheme (13) for scenario 3 with hand support walking up a staircase. The horizontal CoM trajectory is shown in *blue*. It can be observed that the results are similar to that from Fig. 7. *On the right* the resulting z-direction trajectory from the MPC scheme (13) for scenario 3. In *black*, the CoM trajectory and in *red*, the reference height trajectory. The *blue lines* show the *left* and *right* foot trajectories indicating that the robot is walking up the staircase. The results show that the trajectory successfully tracks the desired height above the staircase platform

the desired height trajectory represented by the red line in Fig. 8 (on the right) was able to be tracked. This example shows the robustness of the formulation and the QCQP solver to various different scenarios.

5 Discussion

To demonstrate the advantages and use of the 3D model for dynamically balanced locomotion in Sect. 2, two different example ways to use the criterion for MPC generation of CoM trajectory were presented. Both approaches have a fundamentally common point: the generalisations to allow for 3D non-coplanar multiple contacts naturally result in non-convex problems. However, regardless of the use of the criterion as an objective or constraint, the nature of the criterion is that it is in a bilinear form that can be converted into a convex quadratic function by restricting some variables. Both MPC approaches take advantage of this and then solve a succession of convex optimization problems. This is important in the proposed MPC schemes to ensure that it is still possible to implement them on a robot in real-time. Comparing between the two MPC schemes, it can be observed that using the dynamically balancing criterion in the objective function results in a problem with more decision variables than treating it as a constraint ($11K$ vs. $7K$, where K is the horizon length). However, the number of constraints is significantly less in return ($9 + K$ vs. $4K$), and the resulting problem is only a QP rather than a $QCQP$. As a result, the approach from Sect. 3 is expected to be more computationally efficient than the one from Sect. 4. But the constraint MPC approach provides a stricter notion of feasibility to dynamically balanced locomotion, and is less concerned with optimality.

Finally, it is also worth noting that compared to the traditional ZMP+MPC approach, several restrictions have been removed, such as coplanar contacts and predetermined height trajectory. The interesting point is that if any of these are relaxed, the problem complexity is identical to that if all are relaxed. As such, the 3D formulation proposed relaxes many conditions from the ZMP+MPC approach while still maintaining a balance with the computational cost of the resulting method. As with the ZMP+MPC approach, the focus is typically more on generating dynamically balanced motion rather than optimal gait behaviour. Hence, the development of methods such as AQP and FPP-SCA provides the opportunity to generate feasible motion for more general locomotion scenarios in real-time control.

6 Conclusion and Future Work

We proposed a novel model for dynamically balanced trajectory generation, more general than the classical IPM+ZMP approach, but simple enough to enable fast computations of CoM trajectories through an iterative resolution of convex QPs or convex QCQPs. This claim is supported by the low number of decision variables and constraint equations shown in the problem analysis. The generalizations gained from the proposed model and MPC approach include the ability to allow for multiple non-coplanar contacts and not having to predefine the CoM height trajectory. The results of the two proposed MPC approaches support the belief that the proposed 3D model of dynamically balanced locomotion is a good candidate for real-time model predictive control for multi-contact locomotion.

In future work, we will focus on performing experiments on a real humanoid robot, and address the following points:

1. Both the AQP and FPP-SCA approaches are observed to work well in practice and converge quickly. However there is no mathematical guarantee on the optimality of the solution, hence better understanding and analysis of such methods on the particular structure of the proposed MPC formulations should be more precisely studied.
2. In the optimisation of the CoM trajectories, the contact locations $\mathbf{c}_i$ could also be optimized without losing the convex QP structure. This allows the potential to not only compute the CoM trajectory, but also optimised contact locations.

Acknowledgements The research presented in this paper was partially funded by the ROMEO2 project.

References

1. Aldebaran: The Romeo project. http://projetromeo.com/
2. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, Berlin (2008)
3. Goswami, A.: Postural stability of biped robots and the foot-rotation indicator (FRI) point. Int. J. Robot. Res. **18**(6), 523–533 (1999)

4. Harada, K., Kajita, S., Kaneko, K., Hirukawa, H.: An analytical method on real-time gait planning for a humanoid robot. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids'04), pp. 640–655 (2004)
5. Harada, K., Kajita, S., Kaneko, K., Hirukawa, H.: Dynamics and balance of a humanoid robot during manipulation tasks. IEEE Trans. Robot. **22**(3), 568–575 (2006)
6. Herdt, A., Perrin, N., Wieber, P.: Walking without thinking about it. In: IEEE International Conference on Intelligent Robots and Systems (IROS'10), pp. 190–195 (2010)
7. Herdt, A., Perrin, N., Wieber, P.: LMPC based online generation of more efficient walking motions. In: IEEE/RAS International Conference on Humanoid Robotics (Humanoids'12), pp. 390–395 (2012)
8. Hirukawa, H., Hattori, S., Harada, K., Kajita, S., Kaneko, K., Kanehiro, F., Fujiwara, K., Morisawa, M.: A universal stability criterion of the foot contact of legged robots - adios ZMP. In: IEEE International Conference on Robotics and Automation (ICRA'06), pp. 1976–1983 (2006)
9. Inomata, K., Uchimura, Y.: 3DZMP-based control of a humanoid robot with reaction forces at 3-dimensional contact points. In: IEEE International Workshop on Advanced Motion Control, pp. 402–407 (2010)
10. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., Hirukawa, H.: The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01), pp. 239–246 (2001)
11. Lee, S., Goswami, A.: Reaction Mass Pendulum (RMP): an explicit model for centroidal angular momentum of humanoid robots. In: IEEE International Conference on Robotics and Automation (ICRA'07), pp. 4667–4672 (2007)
12. Majumdar, A., Ahmadi, A.A., Tedrake, R.: Control design along trajectories with sums of squares programming. In: IEEE International Conference on Robotics and Automation (ICRA'13), pp. 4054–4061 (2013)
13. Mehanna, O., Huang, K., Gopalakrishnan, B., Konar, A., Sidiropoulos, N.D.: Feasible point pursuit and successive approximation of non-convex QCQPs. IEEE Signal Process. Lett. **22**(7), 804–808 (2015)
14. Moulard, T., Lamiraux, F., Bouyarmane, K., Yoshida, E., et al.: RobOptim: an optimization framework for robotics. In: The Robotics and Mechatronics Conference (ROBOMEC'13) (2013)
15. Posa, M., Cantu, C., Tedrake, R.: A direct method for trajectory optimization of rigid bodies through contact. Int. J. Robot. Res. **33**(1), 69–81 (2014)
16. Trinkle, J.C., Pang, J.S., Sudarsky, S., Lo, G.: On dynamic multi-rigid-body contact problems with coulomb friction. ZAMM-J. Appl. Math. Mech./Zeitschrift für Angewandte Mathematik und Mechanik **77**(4), 267–279 (1997)
17. Vukobratović, M., Borovac, B.: Zero-moment point–thirty five years of its life. Int. J. Humanoid Robot. **1**(1), 157–173 (2004)
18. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
19. Wieber, P.: Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids'06), pp. 137–142 (2006)
20. Wieber, P.: Viability and predictive control for safe locomotion. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08), pp. 1103–1108 (2008)

The Yoyo-Man

Jean-Paul Laumond, Mehdi Benallegue, Justin Carpentier and Alain Berthoz

1 Introduction: Legs Versus Wheels?

Goal oriented motion is a distinguished character of living beings. A stone does not move by itself. Within the living systems, displacement is what makes the difference between plants and animals. Animals make use of fins in the water and wings in the air. On land, apart from exceptions as crawling snakes, most of the animals are equipped with legs. Legged locomotion is based on rotating articulated limbs. The rotation of the limbs around the contact points on the ground transfers the body from a position to another one. Rotation then appears as a solution to translate an articulated body. If nature applies this principle to legged animals, it is surprising that it does not push this principle until the wheel discovery. Wheel has been invented and developed by humans.[1] Our cars are equipped with wheels and not with legs.

The magic of the wheel is to transform a rotational motion into a translational one as soon as the wheel touches the ground. In this paper we intend to reveal the presence of a virtual wheel as condensing all the apparent complexity of the bipedal locomotion.

[1] The statement has to be nuanced: rotating engines exist at molecular scale and some insects are able to shape objects as spheres to move them.

J.-P. Laumond (✉) · M. Benallegue · J. Carpentier
LAAS CNRS, Univ. de Toulouse, UPS, 7 Avenue du Colonel Roche,
31400 Toulouse, France
e-mail: jpl@laas.fr

M. Benallegue
e-mail: mehdi.benallegue@laas.fr

J. Carpentier
e-mail: justin.carpentier@laas.fr

A. Berthoz
Collège de France - 11, Place Marcelin Berthelot, 75231 Paris Cedex 5, France
e-mail: alain.berthoz@college-de-france.fr

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_13

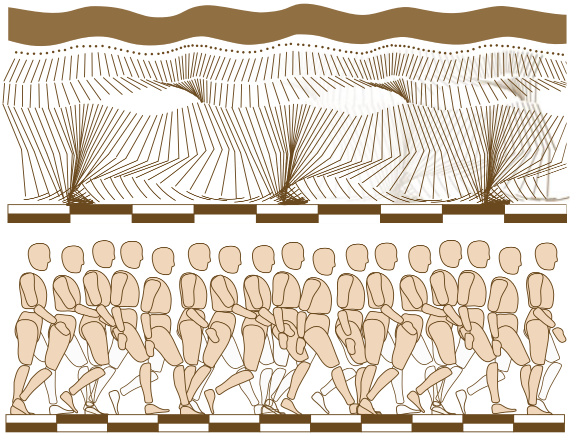


Fig. 1 From a chronophotographic image by E.J. Marey: Walking is a complex process involving the actuation and the motion coordination of many body segments. What does motion capture reveal on the underlying synergies?

The motivation is twofold. From a biomechanics and neuroscience perspective we want to explore the synergies of human locomotion: how the walking body reveals motion invariants beyond the well-known arm-leg coordination (Fig. 1)? From a robotics perspective [32] we seek to fill the gap between two opposite approaches of humanoid locomotion control. The most robust one is based on the control of the center of pressure between the feet and the ground allowing humanoid robots to walk on rough terrains. The second approach is based on clever mechanical designs that take advantage from the gravity. In the latter case the locomotion is much less energy consuming; however it is very fragile with respect to the ground perturbations.

The Yoyo-Man project intends to contribute to new mechanical and control designs for bipedal walkers inspired both by a better understanding of human walking and by the current research on passive walkers.

Figure 2 illustrates the rationale underlying the project. The rationale is twofold. From a mechanical perspective, a wheel rotating at the extremity of a string (i.e., a yoyo) induces its own translation as soon as it touches the ground. Legs are made of three rotating segments (foot, shank and thigh). A first question is addressed in Sect. 3: is there a locomotion geometric reference center to describe the motion of the foot independently from the motion of the shank and the thigh? On the other hand, from a neuroscience perspective, it is known that humans stabilize their head while walking. The second question we address in Sect. 4 is the following one: is there some mechanical benefit to equip passive walkers with a stabilized head on the top of them, i.e. a locomotion control center?

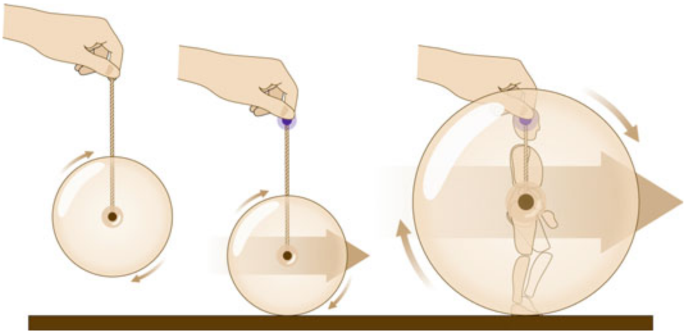


Fig. 2 **The Yoyo-Man**: The hand controls the height of the rotating wheel. The wheel translates as soon as it touches the ground. The Yoyo-man is a human walker model made of the geometric center of a virtual rotating wheel together with a control center located at the head

2 Origins of the Rationale

2.1 Mechanical Basics of Bipedal Walking

Anthropomorphic systems are made of a tree of articulated rigid bodies linked together by rotational joints. This is true for all humanoid robots. This is also true for human at first glance, if we neglect mechanical scapula or kneecap subtleties. Joint positions define the system posture. The system configuration is made of all the joints together with the three placement parameters that give respectively the position and the orientation of the system on the ground. From a control viewpoint, muscles or motors operates in the posture space. There is no direct control of the three placement parameters. In that sense, humans and humanoid robots are underactuated systems. What is called locomotion is the process that modifies the posture of the system in such a way the reaction forces with the ground induce the variation of placement parameters.

Bipedal walking is a cyclic process sequencing two phases: single support when only one foot is touching the ground and double support when both feet are touching the ground. This physical description holds for all bipedal walking systems. The cycle of locomotion is then made of four phases after which it starts again from (almost) the same starting posture. The stability of the locomotion is reflected by the attractiveness of a periodic orbit called limit cycle. It is captured by the so-called Poincaré map [16]. In our context, the Poincaré map is the intersection of the orbit of the periodic walking motion with the posture space at a same instant of the cycle, e.g., when the left foot touches the ground (Fig. 3).

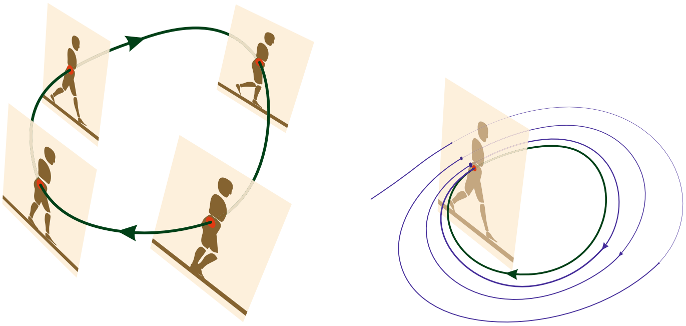


Fig. 3 **Locomotion cycle**: Locomotion is a cyclic process sequencing the same postures alternatively (*left*). The stability of the underlying dynamical system is captured by the Poincaré map (*right*)

2.2 Basics in Humanoid Robot Control

At each phase of the cycle the pressure applied by the surface of feet on the ground may be concentrated onto a single point: the center of pressure. When both the ground and the feet surfaces are flat, the center of pressure coincides with the so-called zero moment point (ZMP) introduced in [30]. As soon as the ZMP remains within the support surface, the system does not fall.

The property of the ZMP is at the origin of a popular locomotion control scheme. The ZMP and the center of mass (CoM) are linked together by nonlinear equations. The control of the CoM is easily derived from the control of the posture. So, in theory, it is possible to control the placement of the ZMP within the surface support. However the nonlinearities linking CoM and ZMP variables make the problem computationally challenging. Under some hypothesis the equations are linear and the problem becomes easier. This is the case when the center of mass remains at the same altitude. Maintaining the CoM at the same altitude is made possible thanks to the redundancy of the anthropomorphic body. The hypothesis is at the origin of the cart-table model introduced in [19] (Fig. 4). The foundations of such control schemes are based on the knowledge of the foot steps to be performed. The literature refers to the so-called preview control [31]: locomotion consists in planning the foot placement in advance.

Passive walkers are designed from a completely different control perspective [9]. They are minimally actuated. The mechanical design is devised to take advantage of the gravity and to convert potential energy into kinetic energy. In its simplest version, the passive walker is made of two articulated legs connected to the hip [10]. It can be modeled as a compass whose gaits induced a motion of the hip that is the same as the motion of the center of a rimless wheel. At that stage, it is noticeable that the

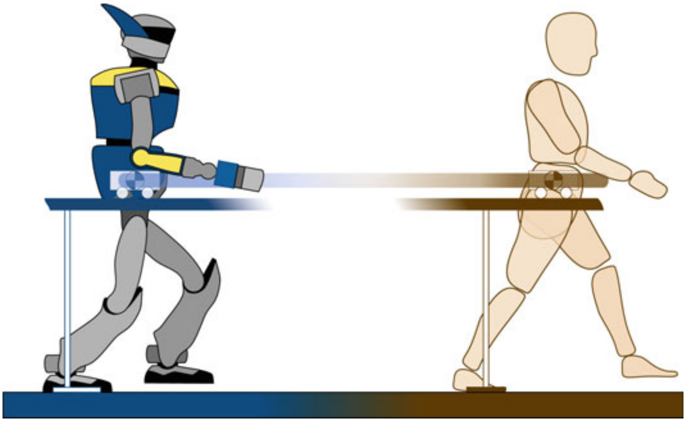


Fig. 4 **Cart-table**: The cart-table model works under the hypothesis that the CoM moves on a horizontal plane. The hypothesis can be applied to control the locomotion of humanoid robots (*left*). Figure 1 suggests it does not hold for humans (*right*)

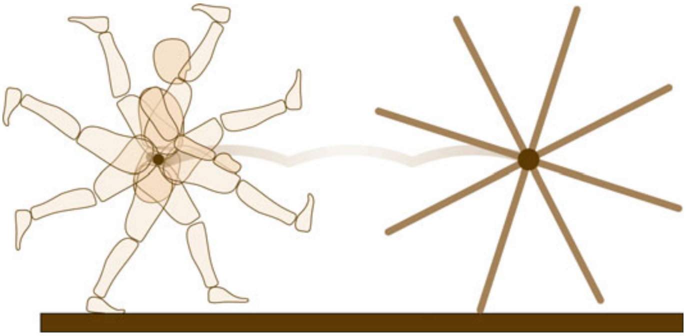


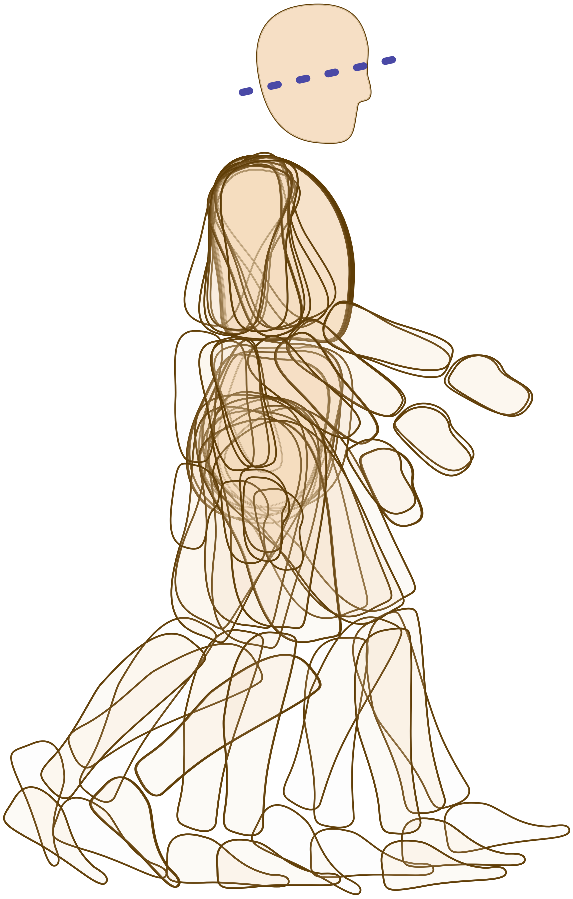
Fig. 5 **Rimless wheel**: At a first glance, the center of a rolling rimless wheel roughly accounts for the motion the hip

motion of the center of a rimless wheel seems to be a rather good approximation of the hip motion in human walking (Fig. 5). The analogy is part of the Yoyo-Man project rational.

2.3 Neurophysiology Basics in Human Walking

Neurophysiologists have observed that humans and animals stabilize their head when moving (see an illustration in Fig. 6). Head stabilization facilitates the fusion of visual and vestibular information. It offers a consistent egocentric reference frame for motion perception for locomotion [23]. In the Yoyo-Man project we argue that

Fig. 6 Sketch of the superimposition of walker positions in different phases of the cycle. The superimposition is achieved so that the head is in the same position. The head is stabilized to keep constant orientation displayed by the *dotted blue line*. (Inspired by a drawing in [23])



head stabilization also contribute mechanically to the balance when walking. In depth description of the sensory cognitive benefits of head stabilization and preliminary results about its mechanical advantages are presented in Sect. 4.

Do humans plan their steps in advance? Sometimes, they obviously do, when the ground is too uneven. However most of the time, they walk without thinking, i.e. without consciousness of any planning phase computing in advance where they have to place their feet. How does walking in the street differ from walking on a mountain path? On the top part of the pavement depicted by Fig. 7, we have to anticipate what stones will next be used for stepping. On the other hand, walking on the pavement at the bottom part of the figure does not require any anticipation of the foot placements. In which context do we start watching our steps? Sect. 4.4 addresses the question by introducing the notion of ground texture.

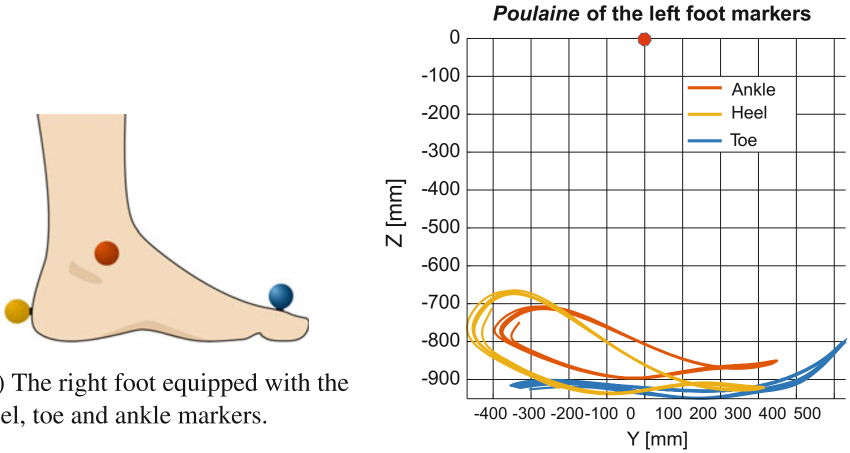
Fig. 7 Pavement in Roma: two textured grounds. In the bottom part, we walk without thinking, in the upper part, one has to watch his/her steps

3 In Search of a Geometric Center for the Yoyo-Man

This section brings to light the geometrical similarity between the rimless wheel and the human body during walking (Fig. 5). While rolling on the floor, the center of the rimless wheel describes a sequence of circle arcs whose radius correspond to the stand beam. From a local viewpoint, this statement can be rephrased as follows: the contact point describes an arc of circle around the center of the rimless wheel during each supporting phase. In the case of human body, does there exist such a link between the foot touching the ground and some point that plays the role of the center of some rimless wheel? As far as we know, this question has never been addressed in human motion modeling. At first glance, the articulation point between the thighbone and the pelvis, i.e. the hip center, would be a good candidate to play the role of the locomotion geometric center. This is not the case. In this section, we show both that the proposed rimless wheel model holds for human walkers, and that the center of the rimless wheel is the center of mass (CoM) of the walking body.

3.1 Experimental Setup

The experimental setup is based on an existing motion database used in [22]. It is composed of 12 participants (5 women and 7 men, 32.8 ± 5.9 years old, 1.71 ± 0.09 m, 65.3 ± 10.1 kg) who have been asked to walk straight at three different speeds three times each: natural, slow, and fast walking speed. Subjects were equipped with 41 reflective markers, with a standard markers placement allowing to compute the center of mass trajectory by means of anthropomorphic tables [12]. Finally, the segmentation of gait into simple et double support phases was achieved by using the methodology described in [15].



(a) The right foot equipped with the heel, toe and ankle markers.

(b) *Poulaines* of the foot markers.

Fig. 8 Illustration of the right foot equipped with the heel, toe and ankle markers and *Poulaines* of those markers along 8 steps. None of the *poulaines* describes a circular path relatively to the pelvis center

In our study, we are interested by natural locomotion. So, from the database we extracted the trials dealing with natural velocity. The the total number of analyzed trajectories is $12 \times 3 = 36$.

3.2 *Identification of the Foot-CoM Relationship*

Poulaine[2] is a French word designating the trajectory of the anatomic feet markers (e.g. ankle, heel, toe) relatively to the geometric center of the pelvis and expressed in the world frame. For instance, Fig. 8 illustrates the *poulaines* of the heel, toe and ankle markers respectively.

At the first sight, none of the aforementioned anatomic markers describes a circular trajectory relatively to the pelvis center.[3] At most, some *poulaines* have a temporally (i.e. during a short period) a constant curvature, but not during all the stance phase. Our approach consists in moving the reference frame from the hip joint center to the CoM. We then show that a particular convex combination of the heel, ankle and toe markers of the stance leg describes a circular trajectory whose center is very close to the center of mass itself.

Choosing the CoM as the center of the reference frame and considering a convex combination of the toe, ankle and heel markers are supported by the following

[2] We did not find the exact translation of this word in English.

[3] In biomechanics, the pelvis center is considered as the root node from which the body segment tree is built.

rationale. Firstly, the shift from the root marker to the center of mass allows us not to consider one precise segment (i.e. the root) but to take into account the overall movement of the human body. Secondly, by choosing a convex combination of the three aforementioned markers, we ensure that this particular point has an almost zero velocity during the stance phase.[4] It can therefore be treated as the pivot point of the rimless wheel.

3.3 Methodology

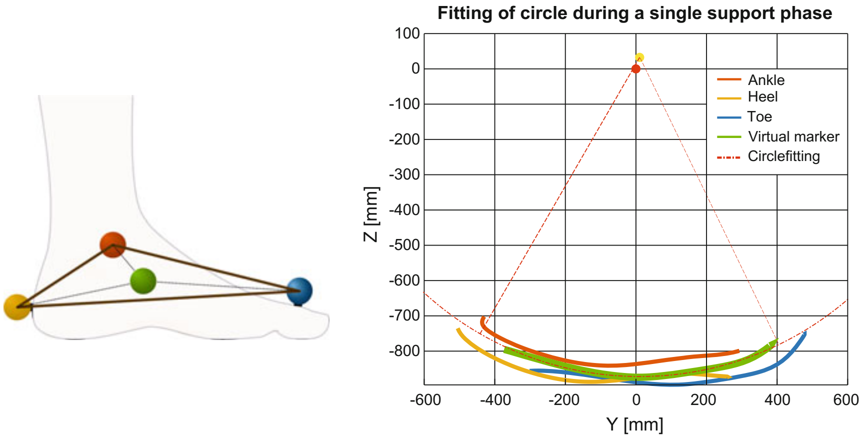
Each walking trial is composed of 10 steps. We divided each of these trials into phases of single and double support phases. Then we introduce a virtual marker at the convex combination of toe, ankle and heel markers by selecting a particular convex combination for each subject, we fitted in the least-square sense the best circle passing through this virtual marker during 85% of the single support phase. On average, the root mean square error of the fitting part was around 2.5 mm. Figure 9 illustrates the procedure by showing the fitted circle having a center (yellow marker) very close to the CoM (red marker) and passing on average by the convex combination (in green). The other curves correspond to the anatomic markers of the foot, the hip joint center and the pelvis center.

3.4 Results

For each subject, we computed the covariance matrix of the set of circle center positions relative to either the center of mass or the hip joint center. From the inverse of both covariance matrices, we define two distance metrics centered on the mean position of the circle centers and relative to the both reference points: the center of mass and the hip joint center. At the end, we obtained two dimensionless distances which discriminate if the two reference points belong to the circle center distributions or not.

Figure 10 summarizes the study over the 12 subjects. For the two metrics, the bar errors plotted at the top of each orange or blue boxes of Fig. 10 corresponds to the confidence interval $[-1; 1]$. While the height of the boxes corresponds to the dimensionless distance between either the center of mass or the hip joint center and the circle center distributions. We can remark that for all subjects, the CoM lives in the confidence interval of the circle center distributions. It is never the case concerning the hip joints center. Those observations allow us to conclude as following: first, there exists a similarity between the rimless wheel and humans during nominal walking

[4] It is worth to mention at this stage that, due to the rolling of the foot on the ground, there is no zero velocity point which is fixed in the feet during the stance phase.



(a) The virtual marker as a convex combination of the anatomic foot markers.

(b) The virtual and anatomic marker trajectories and the fitted circle.

Fig. 9 The virtual marker location and its trajectory relative to the CoM. The virtual marker (i.e. the convex combination of heel, toe and ankle markers) follows a circle whose center (*yellow point*) is close to the CoM (*red point*)

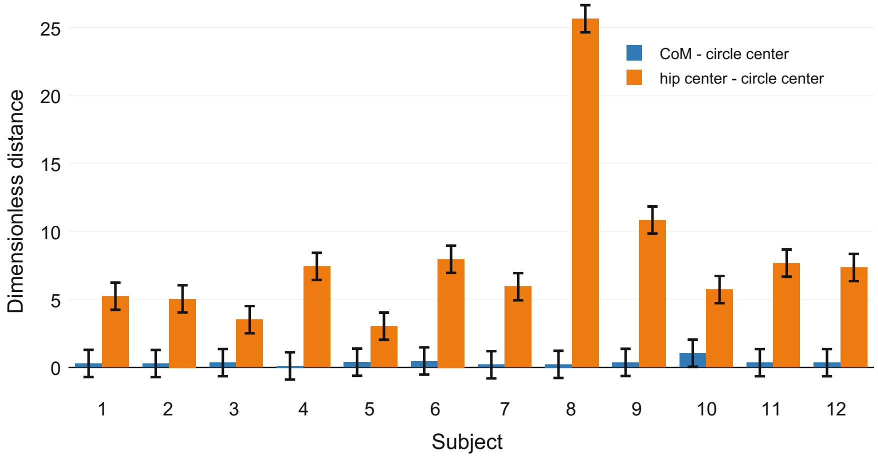


Fig. 10 Dimensionless distance between the fitted circle centers and the CoM or the hip joint center. For all subjects, the center of mass belongs to the distribution of circle centers. This is not true in the case of the hip joint center

gait and second, the center of this rimless does not correspond to the geometric pivot center (i.e. the hip joint center) but rather to the center of mass itself.

Finally, it is worth mentioning that our results hold only in the case of nominal gaits (i.e. walking gait with natural comfort velocity). Indeed, in the case of slow or fast walking velocities, we found that there is no convex combination of markers belonging to the stance foot which has a circular path. Some other studies have been focused on formulating a generic model describing the center of mass trajectory for a large class of walking speeds [17]. Nonetheless, the proposed model overestimates the vertical displacement of the center of mass while it fits well lateral motions.

4 In Search of a Control Center for the Yoyo-Man

4.1 A Convenient Center of Control

One important property of the human steady gait dynamics is that it takes profit from the natural *passive* dynamics of the body. The passive dynamics is the dynamics of the body when no actuation is present, the robot is then subject only to gravity, external forces and passive elasticity and friction of the joints. The body morphology (especially the hip and knee joints [10]) allows the emergence of most prominent features of walking dynamics. The benefits of this structure is to enable the generation of walking motion with high energy efficiency and low control frequency [1]. Furthermore, the control of steady gait has been investigated to suggest that it happens in a very low level of the brain, in a spinal level, consisting in a combination of a simple rhythm generator and reflexes to external perturbations [11]. The steady gait seems to require minimal muscular efforts and cognitive involvements: we walk without thinking about it.

However, as we said earlier, neurophysiologists have observed that humans stabilize actively their head when moving, including walking on flat surfaces. By stabilization, we mean that the head tilt is controlled to remain relatively constant compared to other limbs of the body. Head stabilization is a task prone to dissipate energy since it works almost always against the motion. So why do humans stabilize their head?

The head carries most of the sensory organs, and specifically the visuo-vestibular system, responsible for a great part of balance estimation, spatial localization and motion perception. It can be understood then that stabilizing the head facilitates the fusion of visual and vestibular information. Recent studies show also that head stabilization improves the accuracy of estimation of vertical direction by vestibular-like inertial sensor [14]. Head stabilization improves perturbation detection and safety supervision. Moreover, head tilt conservation offers a consistent and stable egocentric reference frame for perception and generation of motion in general [6] and locomotion in particular [18, 23].

These explanations fit with clinical observations on humans. The unsteadiness and the loss of balance resulting from head-neck system sensorimotor disturbances have been widely documented [7, 20, 25, 29]. It has even been suggested that the impairments in the neck somatosensory inputs and sensorimotor control are as important for balance as a lower-limb proprioception loss following a knee or an ankle injury [27].

Therefore, we can consider that the head is the convenient center of locomotion control: even when we happen to walk without thinking, it offers a comfortable frame with stable dynamics and there takes place the perception, the cognition and the generation of gait.

However, the head is a relatively rigid limb, representing 7% of the total mass of the body, and occupies the top 12% of its height. That means a non-negligible deal of the inertia lies in there. Therefore, head stabilization which actively modifies the motion of the head, should have a noticeable impact on the dynamics of the gait. This effect may be negative, perturbing the walking dynamics and requiring the rest of the body to compensate for it. Alternatively it can be part of the desired dynamics, enhancing balance and improving coordination. In few words: does the head-stabilization by itself contribute to war effort against falling?

4.2 The Model of Steady-Gait Head-Body Dynamics

Based on mechanical concepts from passive robot walkers [8, 9], we introduce a walking simulation scheme where two simple walking mechanical models are then compared. These models include improvements to classical compass-like walkers, by adding torso, interleg actuation, spring-damper at the feet, and rough terrains.

Figure 11 illustrates our mechanical model. It operates in the sagittal plane. It is made of five articulated rigid bodies: two bodies for the (knee-free) legs, one body for the torso, one for the neck and one for the head. Note that the neck is modeled as an articulated body and not as a simple joint. This setting reflects the property of the head-neck system to have two centers of rotation in the sagittal plane: one at the base of the neck and the other at ear level [28]. The mass distribution and the limb lengths are anthropometric (e.g., [3]). In the first of our two models, the walker has a rigid neck and tends to stabilize the torso upright. In the second one the neck is modeled as a limb of two joints and the walker tends to maintain the head direction constant. Both walker models are inspired by the mechanical design of passive walking robots [9].

Indeed, we do not aim at modeling perfectly the human gait. Up to now, only simple dynamical models allow to reproduce locomotion gaits [21]. Dynamical modeling of human walking is out of reach of all current simulators. Nevertheless the energy efficiency of these robots, the low-frequency of their control and their natural limit-cycle dynamics are common characteristics with human locomotion [1, 16].

Detailed technical description of the models is presented in [5].

4.3 Estimating Balance: Ground Textures and MFPTs

Due to difference in control, the whole body dynamics of the walkers is different. However, both dynamics are balanced on flat surface and converge to a stable limit

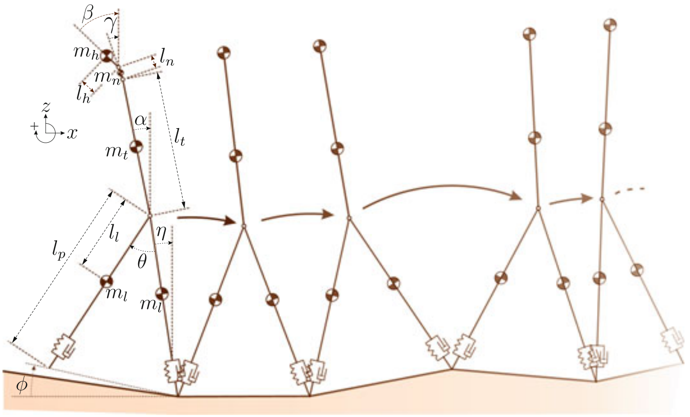


Fig. 11 A representation of the models we simulate. The A model is the same structure subject to the constraints $\alpha = \beta = \gamma$. The B model has stabilized neck joints. The rough terrain is modeled with a slope change at each step

cycle. Therefore both walkers can walk indefinitely on flat surface without falling. However, the difference between the dynamics should lead to a difference in balance performances. This difference should appear in the presence of external perturbations. In our context the perturbation we study is ground *texture*, because it is still today a challenging problem, especially for passive-dynamics walkers.

A textured ground is a ground for which the unevenness follows a probability distribution. For our 2D walker, we model it by changing the ground inclination at each step, following a centered Gaussian law. The standard deviation of the probability distribution define the degree of ground unevenness (see Fig. 11).

Byl and Tedrake [8] present a metrics which is particularly suitable for limit cycle walkers on uneven ground. This metrics is derived from classical analysis of metastable systems and is called Mean First Passage Time (MFPT). Limit-cycle walking is then considered as a metastable system, and MFPT is the mean number of steps the walker makes before falling. This metrics has the property to explore all the reachable dynamics of the walker subject to perturbations, to take into account the repetitive property of ground texture, and to provide a synthetic estimator easy to comprehend intuitively. However, the computation of MFPTs may be time consuming using a naive approach, especially for good performance walkers. To solve this problem, we developed then an optimized algorithm to compute MFPT in reasonable time for complex walking systems [4].

We computed then MFPTs for both models on several ground textures. And we present the results hereinafter.

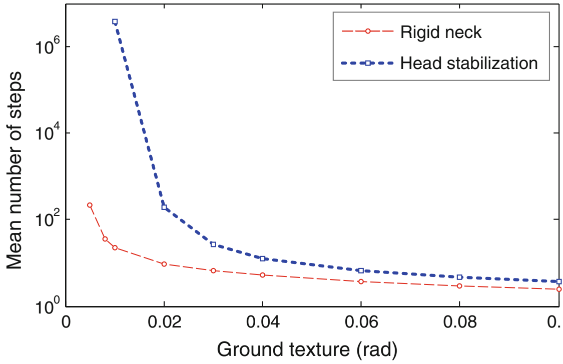


Fig. 12 Mean number of steps with an ideal orientation sensor. Mean number of steps of the walker models equipped with an ideal orientation sensor on different textures of the ground. By texture we mean the standard deviation of the ground slope. MFPTs are displayed in logarithmic scale. For higher ground roughness, MRPT of both models drops such that they need to change their walking control: watching their step becomes necessary

4.4 Results

On flat terrain, and for both control models, it has not been possible to find an upper bound on MFPTs (see Fig. 12). However walker performances greatly differ as soon as a slight texture change appears. The phenomenon can be seen from the example of 0.01 rad standard deviation. In this case, MFPT of the rigid neck model is 23 steps, while head stabilization guarantees MFPT of more than 3 million steps! This performance improvement persists as the ground texture increases, even if the difference declines. This is purely due to mechanical effects, i.e. to the contribution of the head motion to the balance of the gait.

These results may be seen differently. The head stabilization curve of Fig. 12 can be seen as a shift to the right for the rigid neck curve. In other words, head stabilization enables to increase significantly the range of ground textures the walker can handle with the same balance performances.

As this level we may conclude that head stabilization may improve substantially the dynamic balance of walking systems. Head stabilization is an heuristic answer to the question of taking advantage of the head mobility during walking. Indeed, while it is likely not the optimal control of the neck regarding balance, it is a very simple control that produces a complex behavior with significant benefits. Additional explanations for the origin of this effect, including its impact on energy consumption can be found in [5].

5 Conclusions

The preliminary results presented in this paper supports the intuition that bipedal walking can be understood as a wheel rotating around a fixed point (the CoM) while being controlled by a stabilized mass on top of it (Fig. 2). What we introduced as the Yoyo-Man model then appears as a promising route to explore both to elucidate the synergies of the human locomotion and to design new mechanical and control architectures for humanoid robots. Here are the current research directions we are exploring:

- First, we have seen that the rotating rimless wheel model is a rather good model of human locomotion as soon as the center of the wheel is located at the center of mass, and surprisingly not at the joint between the hip and the thighbone. The result holds in the sagittal plane. However the model of the foot we have introduced from the three markers on the heel, the toe and the ankle, does not account for the continuous roll of the feet on the ground. To overcome these limitations, a deeper observation of the CoM motion in the 3-dimensional space deserves to be pursued.
- Second, we have shown that a simple walking compass equipped with a stabilized articulated mass on top of it is more robust to ground perturbations than a compass equipped with the same but non-articulated mass. The result opens new perspectives in the design of humanoid robots based on passive dynamic principles [2]. Why not equipping future humanoid robots with controlled articulated heads?
- Third, after the contribution of the head stabilization in sensing [13], the mechanical contribution of the head stabilization to bipedal walking enhances the role of the head in anthropomorphic action control. Furthermore the head yaw angle anticipates body yaw (shoulder and trunk) and shift in locomotor trajectory [18, 26]. This behavior has been successfully implemented to steer a humanoid robot by its head [24]. However the implementation remains based on a classical preview control of the ZMP. The Yoyo-Man intends to truly “walk without thinking”. It challenges us to devise new locomotion controllers that would be free of any step anticipation and even free of contact force sensors.

Acknowledgements We deeply thank Armel Crétual and Anne-Hélène Olivier from M2S lab, university of Rennes 2, France, for providing the database of captured walking motion.
The work is supported by the European Research Council (ERC) through the Actanthrope project (ERC-ADG 340050) and by the European project KOROIBOT FP7-ICT-2013-10/611909.

References

1. Alexander, R.M.: Walking made simple. Science **308**(5718), 58–59 (2005)
2. Anderson, S., Wisse, M., Atkeson, C., Hodgins, J.K., Zeglin, G., Moyer, B.: Powered bipeds based on passive dynamic principles. In: 5th IEEE-RAS International Conference on Humanoid Robots, pp. 110–116 (2005)
3. Armstrong, H.G.: Anthropometry and mass distribution for human analogues. Technical report, Aerosp. Med. Res. Lab Wright-Patterson AFB Ohio (1988)

4. Benallegue, M., Laumond, J.-P.: Metastability for high-dimensional walking systems on stochastically rough terrain. In: Robotics Science and Systems (2013)
5. Benallegue, M., Laumond, J.-P., Berthoz, A.: A Head-neck-system to Walk Without Thinking (2015). https://hal.archives-ouvertes.fr/hal-01136826
6. Berthoz, A.: The Brain's Sense of Movement. Harvard University Press, Massachusetts (2002)
7. Bove, M., Courtine, G., Schieppati, M.: Neck muscle vibration and spatial orientation during stepping in place in humans. J. Neurophys. **88**(5), 223–241 (2002)
8. Byl, K., Tedrake, R.: Metastable walking machines. Int. J. Robot. Res. **28**(8), 1040–1064 (2009)
9. Collins, S., Ruina, A., Tedrake, R., Wisse, M.: Efficient bipedal robots based on passive-dynamic walkers. Science **307**(5712), 1082–1085 (2005)
10. Collins, S.H., Wisse, M., Ruina, A.: A three-dimensional passive-dynamic walking robot with two legs and knees. Int. J. Robot. Res. **20**, 607–615 (2001)
11. Dietz, V.: Spinal cord pattern generators for locomotion. Clin. Neurophysiol. **114**, 1379–1389 (2003). doi:10.1016/S1388-2457(03)00120-2
12. Dumas, R., Cheze, L., Verriest, J.-P.: Adjustments to mcconville et al. and young et al. body segment inertial parameters. J. Biomechan. **40**(3), 543–553 (2007)
13. Farkhatdinov, I., Hayward, V., Berthoz, A.: On the benefits of head stabilization with a view to control balance and locomotion in humanoids. In: Humanoids 2011, pp. 147–152 (2011)
14. Farkhatdinov, I., Michalska, H., Berthoz, A., Hayward, V.: Modeling verticality estimation during locomotion. In: Romansy 19 - Robot Design, Dynamics and Control CISM International Centre for Mechanical Sciences, vol. 544, pp. 359–366 (2013)
15. Fusco, N., Crétual, A.: Instantaneous treadmill speed determination using subject's kinematic data. Gait Posture **28**(4), 663–667 (2008)
16. Goswami, A., Espiau, B., Keramane, A.: Limit cycles in a passive compass Gait Biped and Passivity-Mimicking control laws. Auton. Robots **4**, 273–286 (1997)
17. Hayot, C., Sakka, S., Fohanno, V., Lacouture, P.: Biomechanical modeling of the 3d center of mass trajectory during walking. In: Movement and Sport Sciences-Science and Motricité (2013)
18. Hicheur, H., Vieilledent, S., Berthoz, A.: Head motion in humans alternating between straight and curved walking path: combination of stabilizing and anticipatory orienting mechanisms. Neurosci. Lett. **383**, 87–92 (2005). doi:10.1016/j.neulet.2005.03.046
19. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: IEEE International Conference on Robotics and Automation, vol. 2, pp. 1620–1626 (2003). doi:10.1109/ROBOT.2003.1241826
20. Lajoie, Y., Teasdale, N., Cole, J.D., Burnett, M., Bard, C., Fleury, M., Forget, R., Paillard, J., Lamarre, Y.: Gait of a deafferented subject without large myelinated sensory fibers below the neck. Neurology **47**(1), 10915 (1996). ISSN 0028-3878. http://www.ncbi.nlm.nih.gov/pubmed/8710062
21. Mombaur, K.: Using optimization to create self-stable human-like running. Robotica **27**(3), 321–330 (2009)
22. Olivier, A.-H., Kulpa, R., Pettré, J., Cretual, A.: A step-by-step modeling, analysis and annotation of locomotion. Comput. Anim. Virtual Worlds (2011). https://hal.inria.fr/inria-00536608
23. Pozzo, T., Berthoz, A., Lefort, L.: Head stabilization during various locomotor tasks in humans. Exper. Brain Res. **82**, 97–106 (1990)
24. Sreenivasa, M.N., Soueres, P., Laumond, J.-P., Berthoz, A.: Steering a humanoid robot by its head. In: IROS, pp. 4451–4456 (2009)
25. Stokell, R., Yu, A., Williams, K., Treleaven, J.: Dynamic and functional balance tasks in subjects with persistent whiplash: a pilot trial. Manual therapy **16**(4), 3948 (2011). ISSN 1532-2769. 10.1016/j.math.2011.01.012. http://www.ncbi.nlm.nih.gov/pubmed/21367648
26. Raphan, T., Imai, T., Moore, S.T., Cohen, B.: Interaction of the body, head and eyes during walking and turning. Exper. Brain Res. **136**, 1–18 (2001)
27. Treleaven, J.: Sensorimotor disturbances in neck disorders affecting postural stability, head and eye movement control. Manual therapy **13**(1), 211 (2008). ISSN 1532-2769. 10.1016/j.math.2007.06.003. http://www.ncbi.nlm.nih.gov/pubmed/17702636

28. Viviani, P., Berthoz, A.: Dynamics of the head-neck system in response to small perturbations: analysis and modeling in the frequency domain. Biolog. Cybern. **19**(1), 1937 (1975). ISSN 0340-1200. http://www.ncbi.nlm.nih.gov/pubmed/1191717
29. Vuillerme, N., Pinsault, N., Vaillant, J.: Postural control during quiet standing following cervical muscular fatigue: effects of changes in sensory inputs. Neurosci. Lett. **378**(3), 1359 (2005). ISSN 0304-3940. 10.1016/j.neulet.2004.12.024. http://www.ncbi.nlm.nih.gov/pubmed/15781146
30. Vukobratović, M.: On the stability of anthropomorphic systems. Math. Biosci. **15**(1–2), 1–37 (1972)
31. Wieber, P.-B.: Viability and Predictive Control for Safe Locomotion. In: IEEE-RSJ International Conference on Intelligent Robots and Systems, Nice, France (2008)
32. Wieber, P.-B., Kuindersma, S., Tedrake, R.: Handbook of Robotics, 2nd edn., Chapter Modeling and Control of Legged Robots. Springer, Heidelberg (2015)

Quantifying and Optimizing Robustness of Bipedal Walking Gaits on Rough Terrain

Cenk Oguz Saglam and Katie Byl

1 Introduction

Quantifying robustness of legged locomotion is essential toward developing more capable robots with legs. In this work, we study underactuated biped walking models. For such systems, various sources of disturbance can be introduced for robustness analysis. While keeping the methods generic, this paper focuses on two-legged locomotion and studies *stability on rough terrain*, or equivalently, *robustness to terrain disturbance*.

An intuitive and capacious approach is to use two levels for controlling bipedal locomotion. Fixed low-level controllers are blind to environmental information, such as the terrain estimation. Given environment and state information, the high-level control problem defines a policy to choose the right low-level controller at each step. Our previous work has always assumed a fixed set of low-level gait controllers exist and focused on the high-level control design [1]; in this work, we finally address the more fundamental issue of tuning a particular gait (low-level controller) itself.

For optimization of low-level control for stability, quantification is a critical step. In many approaches to biped walking control, stability is conservatively defined as a binary metric based on maintaining the zero-moment point (ZMP) strictly within a support polygon, to avoid rotation of the stance foot and ensure not falling [2]. However, robust, dynamic, fast, agile, and energy efficient human walking exploits underactuation by foot rolling. For such robots, including point-feet walkers, local stability of a particular limit cycle is studied by investigating deviations from the trajectories (gait sensitivity norm [3], H_∞ cost [4], and L2 gain [5]), or the speed of

C.O. Saglam (✉) · K. Byl
Electrical and Computer Engineering Department, University of California,
Santa Barbara, CA 93106, USA
e-mail: saglam@ece.ucsb.edu
URL: http://robotics.ece.ucsb.edu/

K. Byl
e-mail: katiebyl@ece.ucsb.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_14

convergence back after such deviations (using Floquet theory [6, 7]). The L2 gain calculation in [5] was successfully extended and implemented on a real robot in [8]. Alternatively, the largest single-event terrain disturbance was maximized in [9] and trajectories were optimized to replicate human-walking data in [10].

Another approach to robustness quantification begins by stochastic modeling of the disturbances and (conservatively) defining what a failure is, e.g., slippage, scuffing, stance foot rotation, or a combination of such events. After discretizing the disturbance and state sets by meshing, step-to-step dynamics are studied to treat the system as a Markov Chain. Then, the likelihood of failure can be easily quantified by calculating the expected number of steps before falling, or mean first-passage time (MFPT) [11]. Optimizing a low-level controller for MFPT was previously impractical due to high computation time of MFPT for a given controller. However, our work over the years now allows us to estimate this number very quickly, and in turn, various low-level controllers can be optimized and benchmarked.

The rest of this paper is organized as follows. The walker model we study and the terrain model we employ are presented in Sect. 2. We then present two low-level control schemes in Sect. 3: (1) A hybrid zero dynamics strategy, with trajectories based on Bézier polynomials and joint-tracking via PD control as suggested in [12], and (2) sliding mode control with time-invariant piece-wise constant joint references adopted in [1]. Section 4 shows the discretization of the dynamics. Tools for generating and working on a Markov chain are presented in Sect. 5. Section 6 gives results, including both performance benchmarks, using the MFPT metric, and also tables documenting the optimal control parameters found using our algorithm. The latter is of particular use to anyone wishing to repeat and build upon our methods. Finally, Sect. 7 gives conclusions and discusses future work.

2 Model

2.1 The Biped

The planar 5-link biped with point feet and rigid links illustrated in Fig. 1 is adopted as the walker model in this paper. The ankles have no torque, so the model is underactuated. The ten dimensional state of the robot is given by $x := [q \ ; \ \dot{q}]$, where $q := [q_1 \ \ q_2 \ \ q_3 \ \ q_4 \ \ q_5]^T$ is the vector of angles shown in the figure.

When only one of the legs is in contact with the ground, the robot is in the single support phase, which has continuous dynamics. Using the Lagrangian approach, the dynamics can be derived as

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Bu, \tag{1}$$

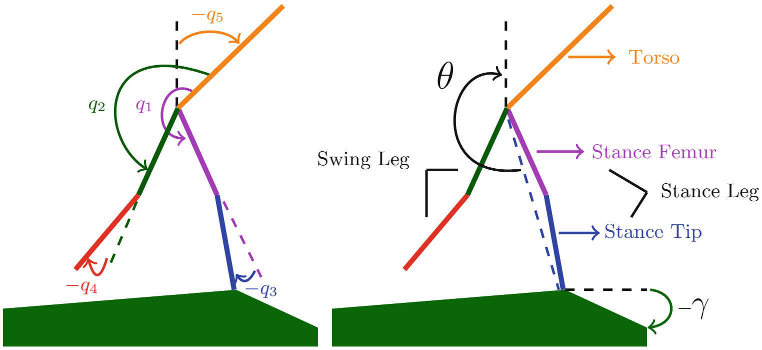


Fig. 1 Illustration of the five-link robot with symmetric legs. As will be explained, θ is called the phase variable

where u is the input. Equation (1) can be equivalently expressed as

$$\dot{x} = \begin{bmatrix} \dot{q} \\ -D^{-1}(C\dot{q} + G) \end{bmatrix} + \begin{bmatrix} 0 \\ D^{-1}B \end{bmatrix} u =: f(x) + g(x)u. \tag{2}$$

On the other hand, if both legs are contacting the ground, then the robot is in its double support phase, which can be approximated as an impact map given by

$$x^+ = \Delta(x^-), \tag{3}$$

where x^- and x^+ are the states just before and after the impact respectively. Conservation of energy and the principle of virtual work give the mapping Δ [13, 14].

A step consists of a single support phase and an impact event. Since walking consists of steps in sequence, it has hybrid dynamics. For a step to be successful, certain "validity conditions" must be satisfied, which are listed next. After impact, the former stance leg must lift from ground with no further interaction with the ground until the next impact. Also, the swing foot must have progressed past the stance foot before the impact of the next step. Only the feet should contact the ground. Furthermore, the force on stance tip during the swing phase and the force on the swing tip at impact should satisfy the no-slip constraint given by

$$F_{friction} = F_{normal}\, \mu_s > |F_{transversal}|. \tag{4}$$

If validity conditions are not met, the step is labeled as unsuccessful and the system is modeled as transitioning to an absorbing failure state. This is a conservative model because in reality violating these conditions does not necessarily mean failure.

2.2 The Terrain

In this paper we assume the terrain ahead of the robot is a constant slope until an impact. So each step experiences a slope and the terrain is angular. As shown in Fig. 1, we denote the slope by γ. This terrain assumption captures the fact that to calculate the pre-impact state, the terrain for each step can simply be *interpreted* as a ramp with the appropriate slope.

An alternative and perhaps more common choice is modeling the rough terrain with varying heights like stairs. Both models of rough terrain are equally complex, valid, and important for this paper's purpose and combining the two is a topic of future work. What they both do not consider is the possibility of various intermediate "bumps" that might cause tripping.

3 Control Scheme

This section summarizes two low-level controller strategies that are used to demonstrate the applicability of our method.

1. Hybrid Zero Dynamics Using Proportional-Derivative Control and Bézier Polynomials

The hybrid zero dynamics (HZD) controller framework provides stable walking motions on flat ground. We summarize some key points here and refer interested reader to [12] for details.

While forming trajectories, instead of time, the HZD framework uses a phase variable denoted by θ. Since it is an internal-clock, phase needs to be monotonic through the step. As the phase variable, we use θ drawn in Fig. 1, which corresponds to $\theta = cq$ with $c = [-1\ 0\ -1/2\ 0\ -1]$. Second, since there are only four actuators, four angles to be controlled need to be chosen, which are denoted by h_0. Controlling the relative (internal) angles means $h_0 := [q_1\ q_2\ q_3\ q_4]^T$. Then h_0 is in the form of $h_0 = H_0 q$, where $H_0 = [I_4\ 0]$.

Let $h_d(\theta)$ be the references for h_0. Then the tracking error is given by

$$h(q) := h_0(q) - h_d(\theta) = H_0 q - h_d(cq). \tag{5}$$

Taking the first derivative with respect to time reveals

$$\dot{h} = \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x} f(x) =: \mathcal{L}_f h, \tag{6}$$

where we used the fact that $\frac{\partial h}{\partial x} g(x) = 0$. Then, the second derivative of tracking error with respect to time is given by

$$\ddot{h} = \mathcal{L}_f^2 h + \mathcal{L}_g \mathcal{L}_f h\, u. \tag{7}$$

Substituting the linearizing controller structure

$$u(x) = (\mathcal{L}_g \mathcal{L}_f h)^{-1}(-\mathcal{L}_f^2 h + v) \tag{8}$$

to (7) yields

$$\ddot{h} = v. \tag{9}$$

To force h (and $\dot{h}$) to zero, a simple PD controller given by

$$v = -K_P y - K_D \dot{y} \tag{10}$$

can be employed, where K_P and K_D are the proportional and derivative gains, respectively.

As suggested in [12], we use Bézier polynomials to form the reference (h_d). Let θ^+ and θ^- be the phase at the beginning and end of limit cycle walking on flat terrain respectively. An internal clock which ticks from 0 to 1 during this limit cycle can be defined by

$$\tau(q) := \frac{\theta(q) - \theta^+}{\theta^- - \theta^+}. \tag{11}$$

Then, the Bézier curves are in the form of

$$b_i(\tau) = \sum_{k=0}^{M} \alpha_k^i \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k}, \tag{12}$$

where M is the degree and α_k^i are the coefficients. Then, the reference trajectory is determined as

$$h_d(\theta) := \begin{bmatrix} b_1(\tau) \\ b_2(\tau) \\ b_3(\tau) \\ b_4(\tau) \end{bmatrix}. \tag{13}$$

Choosing $M = 6$ yields $(6 + 1) \times 4 = 28$ α_k^i parameters to optimize. However, for hybrid invariance, $h = \dot{h} = 0$ just before an impact on flat terrain should imply $h = \dot{h} = 0$ after the impact. This constraint eliminates $2 \times 4 = 8$ of the parameters as explained in [12]. In total, $20 + 2 = 22$ parameters must be chosen, including the PD controller gains.

2. Sliding Mode Control with Time-Invariant Piece-Wise Constant References

The second controller strategy of this paper is adopting sliding mode control (SMC) to track piece-wise constant references [1, 15].

As in the HZD control, let h_0 denote the four variables to control. As a result of our experience in previous work [16], we proceed with

$$h_0 := [\theta_2 \;\; q_3 \;\; q_4 \;\; q_5]^T, \tag{14}$$

where $\theta_2 := q_2 + q_5$ is an absolute angle. Equivalently we can write $h_0 = H_0 q$, where

$$H_0 = \begin{bmatrix} 0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix}. \tag{15}$$

Substituting the control input

$$u = (H_0 D^{-1} B)^{-1}(v + H_0 D^{-1}(C\dot{q} + G)), \tag{16}$$

into (1) yields

$$\ddot{h}_0 = v. \tag{17}$$

We then design v such that h_0 acts as desired (h_d). The tracking error is again given by $h = h_0 - h_d$ and the generalized error is defined as

$$\sigma_i = \dot{h}_i + h_i/\tau_i \quad i = \{1, 2, 3, 4\}, \tag{18}$$

where τ_is are time constants for each dimension of h. Note that when the generalized error is driven to zero, i.e. $\sigma_i = 0$, we have

$$0 = \dot{h}_i + h_i/\tau_i. \tag{19}$$

The solution to this equation is given by

$$h_i(t) = h_i(t_0)\; exp(-(t - t_0)/\tau_i), \tag{20}$$

which drives h_i to 0 exponentially fast. Next, v in (17) is chosen to be

$$v_i = -k_i|\sigma_i|^{2\alpha_i - 1} sign(\sigma_i), \quad i = \{1, 2, 3, 4\}, \tag{21}$$

where $k_i > 0$ and $0.5 < \alpha_i < 1$ are called the convergence coefficient and convergence exponent respectively. Note that if we had $\alpha_i = 1$, this would simply be a standard PD controller. Then, τ_i and k_i are analogous to the proportional gain and derivative time of a PD controller. However, $0.5 < \alpha_i < 1$ ensures finite time convergence. For further reading on SMC please refer to [17]. Note that SMC has $4 \times 3 = 12$ parameters to be optimized.

For faster optimization, it is preferable to have fewer parameters to optimize. Motivated by simplicity, we use references in the form of

$$h_d = \begin{cases} [\theta_2^{ref1}\ q_3^{ref}\ q_4^{ref1}\ q_5^{ref}]^T, & \theta_1 := q_1 + q_5 > \pi, \\ [\theta_2^{ref2}\ q_3^{ref}\ q_4^{ref2}\ q_5^{ref}]^T, & \text{otherwise.} \end{cases} \tag{22}$$

Note that the references are piecewise constant and time-invariant. What makes this reference structure appealing is the fact that there are only 6 parameters to optimize. So, in total, $12 + 6 = 18$ parameters are optimized.

4 Discretization

4.1 Discretization of the Dynamics

The impacts when a foot comes into contact with the ground provide a natural discretization of the robot motion. For the terrain profile described in Sect. 2.2, using (2) and (3) the step-to-step dynamics can be written as

$$x[n+1] = \rho(x[n], \gamma[n], \zeta[n]), \tag{23}$$

where $x[n]$ is the state of the robot and $\gamma[n]$ is the slope ahead at step n.

4.2 Discretization of the Slope Set

Our method requires a finite slope set Γ, which we typically set as

$$\Gamma = \left\{ \gamma^\circ \ : \ \frac{\gamma}{d_\gamma} \in \mathbb{Z},\ -20 \le \gamma \le 20 \right\}, \tag{24}$$

where d_γ is a parameter determining the slope set density. The range needs to be wide enough so that the robot is *not* able to walk at the boundaries of the randomness set.

4.3 Meshing Reachable State Space

There are two key goals in meshing. First, while the actual state x might be any value in the 10 dimensional state space, the reachable state space for system is a lower dimensional manifold once we implement particular low-level control options and allow only terrain height variation as a perturbation source. The meshed set of states, X, needs to well cover the (reachable) part of state space the robot can visit. This set should be dense enough for accuracy while not having “too many” elements for computational efficiency. Second, we want to learn what the step-to-step transition mapping, $\rho(x, \gamma, \zeta)$, is for all $x \in X$ and $\gamma \in \Gamma$. Next, an initial mesh, X_i, should be chosen. In this study, we use an initial mesh consisting of only two points. One of these points (x_1) represents all (conservatively defined) failure states, no matter how the robot failed, e.g. a foot slipped, or the torso touched the ground. The other point (x_2) should be in the controllable subspace. In other words, it should be in the basin of attraction for controller ζ.

Then, our algorithm explores the reachable state space deterministically. We initially start by a queue of "unexplored" states, $\overline{X} = \{x \in X_i \ : \ x \neq x_1\}$, which corresponds to all the states that are not simulated yet for all possible terrains. Then we start the following iteration: As long as there is a state $x \in \overline{X}$, simulate to find all possible $\rho(x, \gamma, \zeta)$ and remove x from $\overline{X}$. For the points newly found, check their distance to other states in X. If the minimum such distance exceeds some threshold, a new point is then added to X and $\overline{X}$.

A crucial question is how to set the (threshold) distance metric so that the resulting X has a small number of states while accurately covering the reachable state space? The standardized (normalized) Euclidean distance turns out to be extremely useful, because it dynamically adjusts the weights for each dimension. The distance of a vector $\bar{x}$ from X is calculated as

$$d(\bar{x}, X) := \min_{x \in X} \left\{ \sum_i \left(\frac{\bar{x}_i - x_i}{r_i} \right)^2 \right\}, \tag{25}$$

where r_i is the standard deviation of i^{th} dimension of all existing points in set X. In addition, the closest point in X to $\bar{x}$ is given by

$$c(\bar{x}, X) := \operatorname*{argmin}_{x \in X} \left\{ \sum_i \left(\frac{\bar{x}_i - x_i}{r_i} \right)^2 \right\}. \tag{26}$$

We are now ready to present the pseudocode in Algorithm 1. Two important tricks to make the algorithm run faster are as follows. First, the slope set allows a natural cluster analysis. We can classify states using the inter-leg angle they possess. So, the distance comparison for a new point can be made only with the points that are associated with the same (preceding) slope. This might result in more points in the final mesh, but it speeds up the meshing and later calculations significantly. Secondly, consider a state x. We can simulate $\rho(x, -20^\circ, \zeta)$ just once and then extract $\rho(x, \gamma, \zeta)$ for all $\gamma \in \Gamma$. The reason is, for example, in order robot to experience an impact at -20°, it has to pass through all possible (less steep) impact points in the slope set.

While meshing the whole 10D state space is infeasible, this algorithm is able to avoid the curse of dimensionality because the reachable state space is actually a quasi-2D manifold [18]. As a result, the meshing can be done with a relatively small number of iteration steps.

Algorithm 1 Meshing algorithm

```
Input: Controller $\zeta$, Initial set of states $X_i$, Slope set $\Gamma$ and threshold distance $d_{thr}$
Output: Final set of states $X$, and state-transition map
$\overline{X} \leftarrow X_i$ (except $x_1$)
$X \leftarrow X_i$
while $\overline{X}$ is non-empty do
    $\overline{X}_2 \leftarrow \overline{X}$
    empty $\overline{X}$
    for each state $\overline{x} \in \overline{X}_2$ do
        for each slope $\gamma \in \Gamma$ do
            Simulate a single step to get the final state $x$, when initial state is $\overline{x}$, slope ahead is $\gamma$, and controller $\zeta$ is used. Store this information in the state-transition map.
            if robot did not fall and $d(x, X) > d_{thr}$ then
                add $x$ to $\overline{X}$
                add $x$ to $X$
            end if
        end for
    end for
end while
```

5 Metastable Markov Chains

5.1 Obtaining a Markov Chain

To obtain a finite state machine representation of the system, we need to approximate the dynamics for points $\rho(x, \gamma, \zeta) \notin X$. The most elementary approach is 0'th order approximation given by

$$x[n+1] \approx c(\rho(x[n], \gamma[n], \zeta[n]), X), \tag{27}$$

where $c(\bar{x}, X)$ is the closest point $x \in X$ to $\bar{x}$ for the employed distance metric. Then the deterministic state transition matrix can be written as

$$T^d_{ij}(\gamma, \zeta) = \begin{cases} 1, & \text{if } x_j = c(\rho(x_i, \gamma, \zeta), X) \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

The nearest-neighbor approximation in (27) appears to work well in practice. More sophisticated approximations result in transition matrices not just having one or zero elements, but also fractional values in between [19], and it does not provide much increase in accuracy to the authors' experience.

A Markov Chain can be represented by a stochastic state-transition matrix T defined as

$$T_{ij} := Pr(x[n+1] = x_j \mid x[n] = x_i). \tag{29}$$

To calculate this matrix, the first thing we need to do is assume a distribution over slope set, noted by

$$P_\Gamma(\gamma) = Pr(\gamma[n] = \gamma). \tag{30}$$

In this paper, we assume a normal distribution for P_Γ, with mean μ_γ, and standard deviation σ_γ. After distributing γ values, T can be calculated as

$$T(\zeta) = \sum_{\gamma \in \Gamma} P_\Gamma(\gamma)\ T^d(\gamma, \zeta). \tag{31}$$

As we make d_{thr} and d_γ smaller, we have more accurate representations of the full dynamics at the expense of higher numbers of states in the final mesh.

5.2 Expected Number of Steps Before Failure

This section serves as a summary on how we estimate the expected number of steps before failure, or mean first-passage time (MFPT). For details, we invite interested reader to [20].

The eigenvalues of T cannot have magnitude larger than one. However, the largest eigenvalue is equal to 1 because we model failure as absorbing. Also, the second largest eigenvalue, denoted by λ_2, is non-negative and real.

No matter what the initial condition is, if the robot does not fall within several steps, then the probability density function for the system converges to its *metastable distribution*. Starting with this distribution, with $1 - \lambda_2$ probability the walker is going to fall on the next step, otherwise the probability distribution does not change. Then, the probability of taking n steps only, equivalently falling at the nth step is simply

$$Pr(x[n] = x_1,\ x[n-1] \neq x_1) = \lambda_2^{n-1}(1 - \lambda_2). \tag{32}$$

For $\lambda_2 < 1$, realize that as $n \rightarrow \infty$, the right hand side goes to zero, i.e., the system will eventually fail. Note that we also count the step which ended up in failure as a step. An intuitive check is to consider failing at the first step (taking 1 step only). When $n = 1$ is substituted, we get $1 - \lambda_2$ as expected. Then, the average number of steps can be then calculated as

$$\begin{aligned} MFPT &= E[FPT] \\ &= \sum_{n=1}^{\infty} n\ Pr(x[n] = x_1,\ x[n-1] \neq x_1) \\ &= \sum_{n=1}^{\infty} n \lambda_2^{n-1}(1 - \lambda_2) = \frac{1}{1 - \lambda_2}, \end{aligned} \tag{33}$$

where we used the fact that $\lambda_2 < 1$. As a result, MFPT can then be calculated using

$$M = \begin{cases} \infty & \lambda_2 = 1 \\ \frac{1}{1-\lambda_2} & \lambda_2 < 1. \end{cases} \tag{34}$$

Note that being stable corresponds to $\lambda_2 = 1$, but we will introduce enough roughness so that we always have $\lambda_2 < 1$. This is achieved with a wide-enough slope set and high enough σ_γ.

We would like to mention that instead of steps, expected distance before failure can alternatively be calculated as explained in [21]. However, as listed later in the following section, we did not observe high variances in step width values. So, number of steps is a good indicator of how much the robot travels.

6 Results

Unless stated otherwise, we use the "minimize" function from [22], which is based on fminsearch function, in MATLAB to optimize. At every iteration the reachable state space for given controller parameters is meshed and the corresponding Markov chain is obtained to calculate the expected number of steps as explained in the previous sections. In this paper we optimize for $\mu_\gamma = 0°$, i.e., zero average slope. However, we optimize control for each of a range of particular values of σ_γ. If it is too small, then the MFPT turns out to be very large, which may not be calculated due to numeric software capabilities. Using MATLAB, we can calculate MFPT values up to around 10^{14} reliably. On the other hand, σ_γ should not be too large, otherwise it may not be as easy to differentiate different controllers' performance with all controllers performing "badly" as σ_γ gets large enough. Appropriate range for σ_γ can be easily decided by calculating MFPT for different values with a single mesh. Once we decide on σ_γ, we pick d_γ. $d_\gamma = \sigma_\gamma/2$ is the rule of thumb we apply in this paper. Just like d_γ, d_{thr} can be made smaller for higher accuracy in the expense of higher computation time. Whether d_γ and d_{thr} are small enough can be checked after the optimization by using smaller values and confirming MFPT estimation does not change much. For the d_γ and d_{thr} values listed later in this section, each cost computation (iteration) took around a minute. Typically, a couple of hours are enough to optimize controller parameters.

1. Hybrid Zero Dynamics Using Proportional-Derivative Control and Bézier Polynomials

For the HZD scheme, the base controller, ζ^1_{Base}, is obtained by assuming flat terrain, fixing speed to be 0.8 m/s and minimizing energy usage as in [12]. To obtain ζ^1_{COT}, we remove the speed constraint and optimize for cost of transport (COT) given by

$$COT = \frac{W}{mgd}, \tag{35}$$

where m is the mass, g is the gravitational constant, and d is the distance traveled. In this paper we use a conservative definition of "energy spent" by regarding negative work is also done by the robot, i.e., $W = |W_{positive}| + |W_{negative}|$.

Both of these controllers assume flat terrain, i.e., $\sigma_\gamma = 0°$, in optimization. In addition, the HZD framework shows how to obtain the trajectories only, but not the controller gains. So, we just picked $K_P = 100$ and $K_D = 10$, which works on flat terrain. To obtain ζ^1_{MFPT}, we used the "patternsearch" algorithm in MATLAB to optimize for MFPT with $\sigma_\gamma = 1°$, $d_\gamma = 0.5$ and $d_{thr} = 0.3$. Table 1 lists the parameters for each controller.

Table 1 Parameters for the first controller scheme in radians

	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}
K_P	100	100	169.2681
α^1_0	3.6151	3.6151	3.6037
α^1_1	3.6413	3.6475	3.5957
α^1_2	3.3894	3.4675	3.3948
α^1_3	3.2884	3.2884	3.2914
α^1_4	3.1135	3.1135	3.1136
α^1_5	3.1708	3.1708	3.1701
α^1_6	3.0349	3.0349	3.0448
α^2_0	3.0349	3.0349	3.0448
α^2_1	2.9006	2.9081	2.9259
α^2_2	2.9544	3.4544	3.0162
α^2_3	3.5470	3.0939	3.5302
α^2_4	3.5186	3.5186	3.5255
α^2_5	3.6851	3.6929	3.7298
α^2_6	3.6151	3.6151	3.6037
	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}
K_D	10	10	30.0166
α^3_0	−0.4162	−0.3693	−0.4113
α^3_1	−0.6657	−0.6079	−0.6018
α^3_2	−0.3732	0.0124	−0.3126
α^3_3	−0.3728	−0.6501	−0.3444
α^3_4	−0.2359	−0.1880	−0.2366
α^3_5	−0.3780	−0.3819	−0.3478
α^3_6	−0.3200	−0.3141	−0.3221
α^4_0	−0.3200	−0.3141	−0.3221
α^4_1	−0.2484	−0.2285	−0.2856
α^4_2	−0.3690	−0.7323	−0.3664
α^4_3	−1.1041	−0.1932	−1.1005
α^4_4	−0.3973	−0.3817	−0.3834
α^4_5	−0.4260	−0.5139	−0.5082
α^4_6	−0.4162	−0.3693	−0.4113

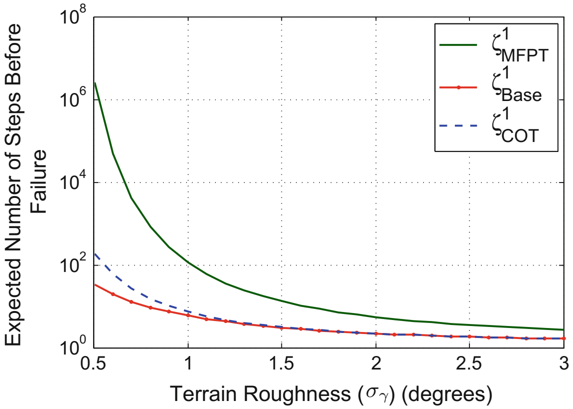


Fig. 2 Average number of steps before falling calculated using (33) versus σ_γ for the first controller scheme. Slopes ahead of the robot are assumed to be normally distributed with $\mu_\gamma = 0°$

Table 2 Estimation of MFPT for First Controller Scheme with $\mu_\gamma = 0°$ and $\sigma_\gamma = 2°$

	ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}
Estimation using (33)	2.2085	2.2049	5.5206
Monte Carlo simulation	2.1511	2.2487	6.1290

We compare the stability of each controller versus the roughness of the terrain in Fig. 2. Noting the logarithmic y-axis, we immediately notice the huge improvement in stability by optimizing with the suggested method.

We note that Monte Carlo simulations are not a computationally practical means of verifying MFPT when it is very high, which has motivated our methodology throughout. However, we present a Monte Carlo study in Table 2 for $\sigma_\gamma = 2°$, where MFPT is small. To obtain the second row in this table, we simulated 10 thousand times. To allow the robot to "forget" the initial condition, we omit the first step, i.e., we only consider cases where it took more than a single step and do not count that first step.

2. Sliding Mode Control with Time-Invariant Piece-Wise Constant References

We start optimizing the second controller scheme with the hand-tuned parameters taken from [16], which we refer to with ζ^2_{Base}. We first optimize for Cost of Transport (COT) of the limit cycle gait on flat terrain to obtain ζ^2_{COT}. We then optimize for MFPT with $\sigma_\gamma = 2°$, $d_\gamma = 1$ and $d_{thr} = 1$. This results with controller ζ^2_{MFPT}. The parameters for each controller are given in Table 3.

Figure 3 compares the stability of each controller versus the roughness of the terrain. Again noting the logarithmic y-axis, the suggested method provides a dramatic increase in the stability, just like in Fig. 2.

Table 4 presents the Monte Carlo study obtained assuming $\sigma_\gamma = 5°$. Just like in Table 2, we omit the first step to allow the simulation to "forget" the initial condition.

Table 3 Parameters for the second controller scheme

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
θ_2^{ref1}	225°	190.5977°	224.9460°
θ_2^{ref2}	204°	200.92392°	203.7358°
q_3^{ref}	0°	$-0.0008°$	$-0.0169°$
q_4^{ref1}	$-60°$	$-19.6094°$	$-60.0042°$
q_4^{ref2}	$-21°$	$-13.4718°$	$-24.0150°$
θ_5^{ref}	0°	$-0.0003°$	0.0040°
k_1	50	49.1126	40.3791
k_2	100	84.2092	96.4343
k_3	75	83.1357	77.1343
k_4	10	7.5848	15.7245
	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
α_1	0.7	0.7977	0.7003
α_2	0.7	0.6063	0.6954
α_3	0.7	0.6838	0.6991
α_4	0.7	0.4873	0.7001
τ_1	0.1	0.1354	0.0920
τ_2	0.1	0.0997	0.0905
τ_3	0.05	0.0679	0.0632
τ_4	0.2	0.1690	0.1918

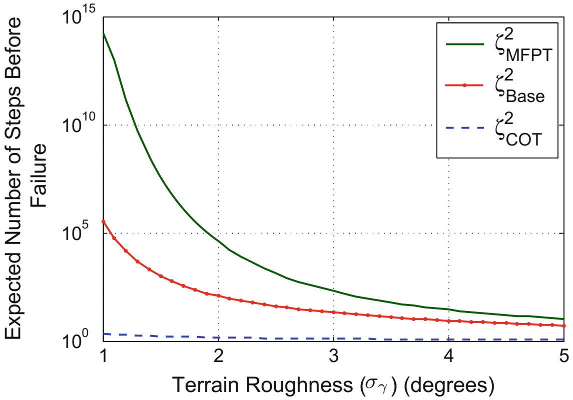


Fig. 3 Average number of steps before falling calculated using (33) versus σ_γ for the second controller scheme. Slopes ahead of the robot are assumed to be normally distributed with $\mu_\gamma = 0°$. Note that both the range of σ_γ and the y-axis scaling are different from Fig. 2

Table 4 Estimation of MFPT for Second Controller Scheme with $\mu_\gamma = 0°$ and $\sigma_\gamma = 5°$

	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
Estimation using (33)	5.1766	1.1470	10.6433
Monte Carlo simulation	5.0738	1.5716	10.4813

Table 5 Comparison of controller schemes for $\mu_\gamma = 0°$

		ζ^1_{Base}	ζ^1_{COT}	ζ^1_{MFPT}	ζ^2_{Base}	ζ^2_{COT}	ζ^2_{MFPT}
$\sigma_\gamma = 1°$ (Stochastic Terrain)	MFPT	5.9	7.3	113.1	3.2×10^5	2.2	1.6×10^{14}
$\sigma_\gamma = 0°$ (Flat Terrain)	Step width	0.413	0.43	0.4	0.456	0.388	0.439
	Speed	0.8	0.516	0.649	0.752	0.513	0.928
	COT	0.187	0.069	0.143	0.869	0.225	0.93

3. Comparison

We first note that all six controllers are stable on flat ground ($\sigma_\gamma = 0°$), because they all exhibit stable limit cycles. However, as Table 5 shows, there is a huge difference between ζ^2_{MFPT} and any of the HZD controllers. Comparing the results in Figs. 2 and 3 also emphasizes this dramatic difference. So, we conclude that the second controller scheme is much more capable in terms of stability. One of the main goals of this paper is to illustrate this benchmarking capability.

We note that many parameters of ζ^2_{Base} and ζ^2_{MFPT} in Table 3 are very close. We suspect that we only find local minimums. Indeed, starting with different initial conditions yields different final gaits.

A major problem in the first controller scheme, we believe, is the fact that reference is designed only for flat terrain. For example, the controller does not really know what to do when $\theta > \theta^+$ (or $\tau > 1$). This is because Bézier polynomials are designed for $0 \le \tau(q) \le 1$, and they quickly deviate outside this range. As a result, ζ^1_{Base} cannot take more than several steps on inclined terrain with a slope of $-1°$. We discovered an easy fix to the problem by adopting the following policy: If $\tau(q) > 0.95$, then do not apply any torque. With this update, the controller can still walk on flat terrain. In addition, it seems to be stable on $-9°$ *degree sloped terrain*! However, we did not present the result with this policy because it ends up with a low MFPT for $\mu_\gamma = 0°$. The reason is, it works very badly on uphill slopes. The fact that turning the controller off greatly helps when going downhill shows the need for a better reference parametrization to keep controller on at all times. Reference [21] presents an attempt to achieve this goal.

7 Conclusions and Future Work

In this work, we present a methodology for optimizing a low-level control scheme and of benchmarking final performance on rough terrain using the MFPT metric for reliability. We apply the approach to two particular control schemes as a motivating example; however, the approach is designed to provide a systematic means of optimizing and benchmarking any of a wide variety of control strategies, not only for walking systems but also for other dynamic systems subject to stochastic environments, more generally.

As mentioned in the previous section, we end up with a local minimum for the second controller scheme. We aim to find the global solution in a future study. The sensitivity of our stability metric to model mismatch is another important future work topic.

Reference [21] is a work that builds on this paper. It presents a controller scheme that is more capable than the two studied in this paper. It also shows that we can also optimize under constraints, e.g., for desired speed, step width, or ground clearance. Furthermore, by designing multiple controllers for different mean slopes, it demonstrates how to increase stability dramatically. Finally, we may use cost functions that incorporate other performance metrics also, similar to [23]. For example, a practical goal is to increase stability while decreasing energy consumption, balancing the two aims as desired.

Acknowledgements This work is supported in part by NSF (CAREER) award 1255018 and by the Institute for Collaborative Biotechnologies through grant W911NF-09-0001 from the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

1. Saglam, C.O., Byl, K.: Robust policies via meshing for metastable rough terrain walking. In: Proceedings of Robotics: Science and Systems, Berkeley, USA (2014)
2. Vukobratovic, M., Borovac, B.: Zero-moment point - thirty five years of its life. Int. J. Humanoid Robot. **1**(01), 157–173 (2004)
3. Hobbelen, D., Wisse, M.: A disturbance rejection measure for limit cycle walkers: the gait sensitivity norm. IEEE Trans. Robot. **23**, 1213–1224 (2007)
4. Morimioto, J., Zeglin, G., Atkeson, C.: Minimax differential dynamic programming: application to a biped walking robot. In: SICE 2003 Annual Conference, vol. 3, pp. 2310–2315, August 2003
5. Dai, H., Tedrake, R.: L2-gain optimization for robust bipedal walking on unknown terrain. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 3116–3123. IEEE (2013)
6. Hurmuzlu, Y., Basdogan, C.: On the measurement of dynamic stability of human locomotion. J. Biomech. Eng. **116**, 30–36 (1994)
7. McGeer, T.: Passive dynamic walking. Int. J. Robot. Res. **9**, 62–82 (1990)
8. Griffin, B., Grizzle, J.: Walking gait optimization for accommodation of unknown terrain height variations. In: American Control Conference (ACC) (2015)
9. Pratt, J., Chew, C.-M., Torres, A., Dilworth, P., Pratt, G.: Virtual model control: an intuitive approach for bipedal locomotion. Int. J. Robot. Res. **20**, 129–143 (2001)
10. Ames, A.D.: First steps toward automatically generating bipedal robotic walking from human data. In: Kozowski, K. (ed.) Robot Motion and Control 2011. Lecture Notes in Control and Information Sciences, vol. 422, pp. 89–116. springer, london (2012)
11. Byl, K., Tedrake, R.: Metastable walking machines. Int. J. Robot. Res. **28**, 1040–1064 (2009)

12. Westervelt, E., Grizzle, J.W., Koditschek, D.: Hybrid zero dynamics of planar biped walkers. IEEE Trans. Autom. Control **48**, 42–56 (2003)
13. Westervelt, E., Chevallereau, C., Morris, B., Grizzle, J., Ho Choi, J.: Feedback Control of Dynamic Bipedal Robot Locomotion. Automation and Control Engineering, vol. 26. CRC Press, Boca Raton (2007)
14. Hurmuzlu, Y., Marghitu, D.: Rigid body collisions of planar kinematic chains with multiple contact points. Int. J. Robot. Res. **13**, 82–92 (1994)
15. Tzafestas, S.G., Krikochoritis, T.E., Tzafestas, C.S.: Robust sliding-mode control of nine-link biped robot walking. J. Intell. Robot. Syst. **20**(2–4), 375–402 (1997)
16. Saglam, C.O., Byl, K.: Switching policies for metastable walking. In: Proceedings of IEEE Conference on Decision and Control (CDC), pp. 977–983, December 2013
17. Sabanovic, A., Ohnishi, K.: Motion Control Systems. Wiley, Singapore (2011)
18. Saglam, C.O., Byl, K.: Stability and gait transition of the five-link biped on stochastically rough terrain using a discrete set of sliding mode controllers. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 5675–5682, May 2013
19. Abbeel, P.: Discretization. In: CS 287: Advanced Robotics Lecture Notes (2012). Accessed: 2015-05-06
20. Saglam, C.O., Byl, K.: Metastable Markov chains. In: IEEE Conference on Decision and Control (CDC), December 2014
21. Saglam, C.O.: Tractable Quantification of Metastability for Robust Bipedal Locomotion. Ph.D. thesis, University of California, Santa Barbara, June 2015
22. Oldenhuis, R.: Minimize - File Exchange - MATLAB Central
23. Saglam, C.O., Byl, K.: Quantifying the trade-offs between stability versus energy use for underactuated biped walking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2014)

A Probabilistic Framework for Semi-autonomous Robots Based on Interaction Primitives with Phase Estimation

Guilherme Maeda, Gerhard Neumann, Marco Ewerton, Rudolf Lioutikov and Jan Peters

1 Introduction

Assistive and collaborative robots must have the ability to physically interact with the human, safely and synergistically. However, pre-programming a robot for a large number of tasks is not only tedious, but unrealistic, especially if tasks are added or changed constantly. Moreover, conventional programming methods do not address semi-autonomous robots—robots whose actions depend on the actions of a human partner. Nevertheless, once deployed, for example in a domestic or small industrial environment, a semi-autonomous robot must be easy to program, without requiring the need of a dedicated expert. For this reason, this paper proposes the use of interaction learning, a data-driven approach based on the use of imitation learning [17] for learning tasks that involve human-robot interaction.

Amongst the several challenges posed by interaction learning, this paper focuses on two intrinsically related problems. First, the problem of estimating the phase of the human movement, that is, the progress or the stage of the execution of the human trajectory under an intermittent stream of position data. This is a problem of practical

G. Maeda (✉) · M. Ewerton · R. Lioutikov · J. Peters
Intelligent Autonomous Systems Group, TU Darmstadt, Darmstadt, Germany
e-mail: maeda@ias.tu-darmstadt.de

M. Ewerton
e-mail: ewerton@ias.tu-darmstadt.de

R. Lioutikov
e-mail: lioutikov@ias.tu-darmstadt.de

G. Neumann
Lincoln Centre for Autonomous Systems, University of Lincoln,
Lincoln, UK
e-mail: gneumann@lincoln.ac.uk

J. Peters
Max Planck Institute for Intelligent Systems, Tuebingen, Germany
e-mail: peters@ias.tu-darmstadt.de

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_15

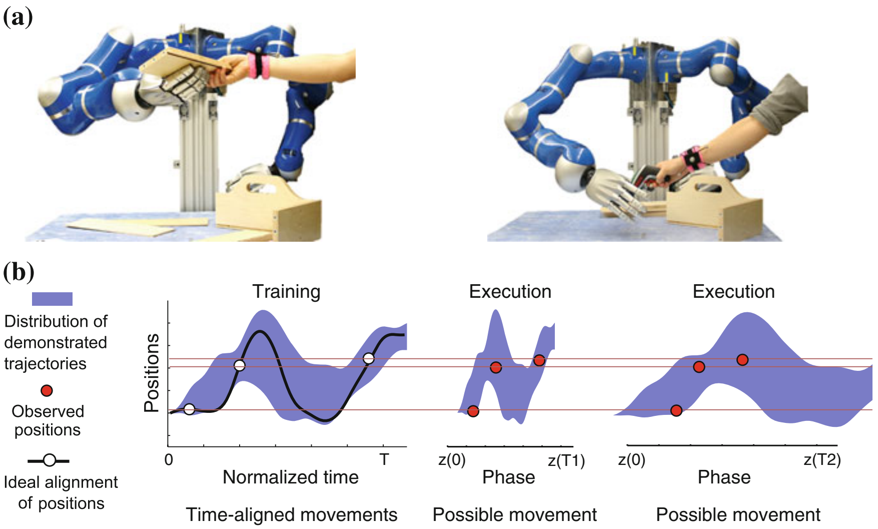


Fig. 1 Collaborative and assistive robots must address both action recognition and movement coordination based on human observations. **a** A robot coworker must recognize the intention of the human before deciding which action to take. **b** Observing the human movement through corrupted (e.g. occluded, sparse, intermittent) position data, poses the problem of identifying the correct phase of the movement

importance since the majority of motion capture systems available, such as marker tracking and depth cameras, rely on planned spaces and well positioned cameras; requirements that are incompatible with most of the already existing collaborative environments of interest (e.g. in a hospital, at home) where occlusions are prone to occur. Second, based on this assessment, we address the problem of recognizing the human action and generating the corresponding movement of the robot assistant. As illustrated in Fig. 1a, by observing the movement of the human, a semi-autonomous robot must decide if it should hand over a plate, or hold a screwdriver. The human, however, may execute movements at different unobserved speeds, and position measurements may be corrupted by occlusions, which cause the problem of temporally aligning sparse position observations with the interaction model. Figure 1b illustrates such a problem where the same sequence of three observed positions may fit two models that are identically spatially, but have different phases of execution. Such an ambiguity hinders the adaptation of the robot movement.

The contribution of this paper is a probabilistic framework for interaction learning with movement primitives that allows a robot to react faster by estimating the phase of the human, and to associate the outcome of the estimation to address different tasks. As the algorithm relies on Probabilistic Movement Primitives [15] for human-robot interaction, the method will also be referred to as Interaction ProMPs. An Interaction ProMP provides a model that correlates the weights that parameterize the trajectories of a human and a robot when executing a task in collaboration. The

Interaction ProMP is conditioned on the observations of the human and the robot is controlled based on a posterior distribution over robot trajectories.

This paper consolidates our recent efforts in different aspects of semi-autonomous robots. It leverages on the representation of movements with ProMPs, our developments in the context of human-robot interaction [1, 13], and the ability to address multiple tasks [7, 13]. While our previous interaction models were explicitly time-dependent, here, we introduce a phase-dependent method. Section 2 emphasizes the most relevant works in phase and time representations and briefly addresses related works in other aspects of the framework.[1] Section 3 describes the proposed method with a brief background on ProMPs, followed by Interaction ProMPs, phase estimation, and action recognition. Finally, Sect. 4 provides experiments and discussions on the application of the method in an assembly scenario.

2 Related Work

Dynamical Movement Primitives [8], or simply DMPs, have been known to address temporal variations with a phase variable. The phase variable is used to govern the spread of a fixed number of basis functions that encode parameters of a forcing function. ProMPs use the concept of phases in the same manner, with the difference that the basis functions are used to encode positions. This difference is fundamental for Interaction Primitives since estimating the forcing function of the human is nontrivial in practice, while positions can be often measured directly [13].

Recently, a modified form of DMPs where the rate of phase change is related to the speed of movement has been presented [20]. The method uses Reinforcement Learning and Iterative Learning Control to speed up the execution of a robot's movement without violating pre-defined constraints such as carrying a glass full of liquid without spilling it. A similar form of iterative learning was used to learn the time mapping between demonstrated trajectories and a reference trajectory [19]. With their approach, a robot was able to perform a surgical task of knot-tie faster than the human demonstrator.

Dynamic Time Warping (DTW) [16] has been used in robotics applications for temporally aligning trajectories. For example, as part of an algorithm that estimates the optimal, hidden trajectory provided by multiple expert demonstrations [4]. Although DTW has been shown suitable for off-line processing of data, its online application can be hard to achieve in practice due to exhaustive systematic search. A different approach is to explicitly encode the time of demonstrations such as in [3], where the structure of the model intrinsically generates smooth temporal solutions. The measurement or estimation of velocity, for example, by differentiation of a consistent stream of positions, removes the ambiguity of Fig. 1b and allows for the realization of online algorithms that cope with very fast dynamics [9, 10]. Such methods,

[1] The interested reader is referred to our previous works for additional and detailed literature review in respect to their corresponding contributions.

however, rely on a planned environment free from occlusions and fast tracking capabilities; requirements difficult to achieve in environments where semi-autonomous robots are expected to make their biggest impact, such as in small factories, hospitals and home care facilities. A limitation of ProMPs in relation to representations based on multiple reference frames such as the Dynamical Systems [3], and forcing functions as in DMPs, is that ProMPs only operate within the demonstrated set of demonstrations.

Several methods to learn time-independent models by imitation have been proposed. For example, Hidden Markov Models (HMM) and Gaussian Mixture Regression (GMR) have been used to learn and reproduce demonstrated gestures [2] where each hidden state corresponds to a Gaussian over positions and velocities, locally encoding variation and correlation. In [5], a method to reactively adapt trajectories of a motion planner due to changes in the environment was proposed by measuring the progress of a task with a dynamic phase variable. While this method is suited for cases where the goal is known from a planned trajectory—the phase is estimated from the distance to the goal—a semi-autonomous robot is not provided with such information: the goal must be inferred from the observation of the human movement, which in turn requires an estimate of the phase.

This paper shares similar challenges faced in [21] where the robot trajectory had to be adapted according to the observation of the human partner during handovers. In [21] the authors encoded the demonstrations in a tree-structured database as a hierarchy of clusters, which then poses the problem of searching matching trajectories given partial observations. The use of a probabilistic approach in the present work allows us to address the search for a matching trajectory simply computing the likelihoods of various models given the observed trajectories.

Several other works have addressed the action recognition problem. Graphical models, in particular, have been widely used. In human-robot interaction, HMMs have been used hierarchically to represent states and to trigger low-level primitives [12]. HMMs were also applied to predict the positions of a coworker in an assembly line for tool delivery [18] while in [11], Conditional Random Fields were used to predict the possible actions of a human. The prediction of the movement of human coworkers was addressed in [14] with a mixture model. The cited methods address the generation of the corresponding robot movement as an independent step, either by pre-programming suitable actions [11], or by using motion planners [14]. In contrast, Interaction ProMPs intrinsically correlate the action of the human with the movement of the robot such that action recognition and movement generation are provided by the same model.

3 Probabilistic Movement Primitives for Human-Robot Interaction

This section introduces ProMPs for a single degree-of-freedom (DoF) from which the multi-DoF ProMP will follow naturally. In human-robot interaction, the use of ProMPs consists on the use of the multi-DoF case where some of the DoFs are given by a tracked human interacting with a semi-autonomous robot. This section finishes by introducing phase estimation, which also provides means to recognize human actions in multiple-task scenarios.

3.1 Probabilistic Movement Primitives on a Single Degree-of-Freedom

For each time step t a position is represented by $\boldsymbol{y}_t$ and a trajectory of T time steps as a smooth sequence $\boldsymbol{y}_{1:T}$. A parameterization of $\boldsymbol{y}_{1:T}$ in a lower dimensional weight space can be achieved by linear regression on time-dependent Gaussian basis functions ψ_t,

$$\boldsymbol{y}_t = \psi_t^T \boldsymbol{w} + \epsilon_y, \tag{1}$$

$$p(\boldsymbol{y}_{1:T}|\boldsymbol{w}) = \prod_1^T \mathcal{N}(\boldsymbol{y}_t|\psi_t^T \boldsymbol{w}, \boldsymbol{\Sigma}_y), \tag{2}$$

where $\boldsymbol{\epsilon}_y \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_y)$ is zero-mean i.i.d. Gaussian noise and $\boldsymbol{w} \in \mathbb{R}^N$ is a weight vector that encodes the trajectory. The number of Gaussian bases N is often much lower than the number of trajectory time steps. The number of basis is a design parameter that must be matched with the desired amount of detail to be preserved during the encoding of the trajectory. In the particular case of the experiments here reported, trajectories have an average time of 3 s, sampled at 50 Hz. The dimensionality is decreased from $3 \times 50 = 150$ samples to a weight vector of length $N = 20$.

Assume M trajectories are obtained via demonstrations; their parameterization leading to a set of weight vectors $\boldsymbol{W} = \{\boldsymbol{w}_1, \ ... \ \boldsymbol{w}_i, \ ... \ \boldsymbol{w}_M\}$ (the subscript i as in $\boldsymbol{w}_i$ will be used to indicate a particular demonstration when relevant, and will be omitted otherwise). Define $\boldsymbol{\theta}$ as a parameter to govern the distribution of $\boldsymbol{W}$ such that $\boldsymbol{w} \sim p(\boldsymbol{w}; \boldsymbol{\theta})$. From the training data we model $p(\boldsymbol{w}; \boldsymbol{\theta})$ as a Gaussian with mean $\boldsymbol{\mu}_w \in \mathbb{R}^N$ and covariance $\boldsymbol{\Sigma}_w \in \mathbb{R}^{N \times N}$, that is $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$. This model allows us to sample from the demonstrated distribution over positions with

$$p(\boldsymbol{y}_t; \boldsymbol{\theta}) = \int p(\boldsymbol{y}_t|\boldsymbol{w}) p(\boldsymbol{w}; \boldsymbol{\theta}) d\boldsymbol{w} = \mathcal{N}(\boldsymbol{y}_t|\psi_t^T \boldsymbol{\mu}_w, \psi_t^T \boldsymbol{\Sigma}_w \psi_t + \boldsymbol{\Sigma}_y). \tag{3}$$

The Gaussian assumption is restrictive in two ways. First, the training data must be time-aligned, for example by DTW; second, only one type of interaction pattern—or collaborative task—can be encoded within a single Gaussian (mixture of models were used to address the latter problem in an unsupervised fashion [7]).

3.2 Correlating Human and Robot Movements with Interaction ProMPs

Interaction ProMPs model the correlation of multiple DoFs of multiple agents. Let us define the state vector as a concatenation of the P number of observed DoFs of the human, followed by the Q number of DoFs of the robot

$$\boldsymbol{y}_t = [\ \boldsymbol{y}^H_{1,t}, \dots \boldsymbol{y}^H_{P,t},\ \boldsymbol{y}^R_{1,t}, \dots \boldsymbol{y}^R_{Q,t}\ ]^T,$$

where the upper scripts $(\cdot)^H$ and $(\cdot)^R$ refer to the human and robot DoFs, respectively. Similar to the single DoF case, all DoF's trajectories are parameterized as weights such that

$$p(\boldsymbol{y}_t|\bar{\boldsymbol{w}}) = \mathcal{N}(\boldsymbol{y}_t|\mathbf{H}_t^T \bar{\boldsymbol{w}}, \boldsymbol{\Sigma}_y), \tag{4}$$

where $\mathbf{H}_t^T = \text{diag}((\psi_t^T)_1, \dots, (\psi_t^T)_P, (\psi_t^T)_{P+1}, \dots, (\psi_t^T)_{P+Q})$ has $P + Q$ diagonal entries. Each collaborative demonstration now provides $P + Q$ training trajectories. The weight vector $\bar{\boldsymbol{w}}_i$ of the i-th demonstration is

$$\bar{\boldsymbol{w}}_i = [\ (\boldsymbol{w}_1^H)^T, \dots, (\boldsymbol{w}_P^H)^T,\ (\boldsymbol{w}_1^R)^T, \dots, (\boldsymbol{w}_Q^R)^T\ ]^T, \tag{5}$$

from which a normal distribution from a set of M demonstrations $\bar{\boldsymbol{W}} = \{\bar{\boldsymbol{w}}_1, \dots \bar{\boldsymbol{w}}_M\}$ with $\boldsymbol{\mu}_w \in \mathbb{R}^{(P+Q)N}$ and $\boldsymbol{\Sigma}_w \in \mathbb{R}^{(P+Q)N \times (P+Q)N}$ can be computed.

The fundamental operation for semi-autonomy is to compute a posterior probability distribution of the weights (now encoding both human and robot) $\bar{\boldsymbol{w}} \sim \mathcal{N}(\boldsymbol{\mu}_w^{new}, \boldsymbol{\Sigma}_w^{new})$ conditioned on a sparse (e.g. due to motion capture occlusion) sequence of observed positions of the human $\boldsymbol{y}^*$ measured within the interval $[t, t']$. This operation can be computed with

$$\begin{aligned} \boldsymbol{\mu}_w^{new} &= \boldsymbol{\mu}_w + \boldsymbol{K}(\boldsymbol{y}^*_{t,t'} - \mathbf{H}^T_{t,t'}\boldsymbol{\mu}_w), \\ \boldsymbol{\Sigma}_w^{new} &= \boldsymbol{\Sigma}_w - \boldsymbol{K}(\mathbf{H}^T_{t,t'}\boldsymbol{\Sigma}_w), \end{aligned} \tag{6}$$

where $\boldsymbol{K} = \boldsymbol{\Sigma}_w \mathbf{H}^T_{t,t'}(\boldsymbol{\Sigma}_y^* + \mathbf{H}^T_{t,t'}\boldsymbol{\Sigma}_w \mathbf{H}_{t,t'})^{-1}$ and $\boldsymbol{\Sigma}_y^*$ is the measurement noise. The upper-script $(\cdot)^{new}$ is used for values after the update and the subscript $(\cdot)_{t,t'}$ is used to indicate the unvenly spaced interval between t and t'. The observation matrix $\mathbf{H}^T_{t,t'}$ is obtained by concatenating the bases at the corresponding observation steps, where the Q unobserved states of the robot are represented by zero entries in the diagonal. Thus, for a each time step t,

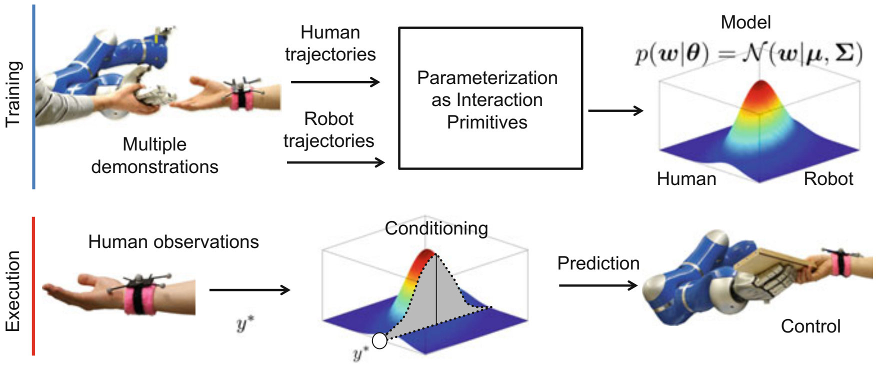

Training
Multiple demonstrations
Human trajectories
Robot trajectories
Parameterization as Interaction Primitives
Model
$p(\boldsymbol{w}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$
Human
Robot
Execution
Human observations
y^*
Conditioning
y^*
Prediction
Control

Fig. 2 The workflow of Interaction ProMP on a single task where the distribution of human-robot parameterized trajectories is abstracted as a bivariate Gaussian. The conditioning step is shown as the slicing of the distribution at the observation of the human position. In the real case, this distribution is multivariate

$$\mathbf{H}_t^T = \left[\begin{array}{ccc:ccc} (\psi_t^T)_1 & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & (\psi_t^T)_P & \mathbf{0} & \dots & \mathbf{0} \\ \hdashline \mathbf{0} & \dots & \mathbf{0} & \mathbf{0}_{P+1} & \dots & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0}_{P+Q} \end{array}\right]. \tag{7}$$

Trajectory distributions that predict human and robot movements are obtained by integrating out the weights of the posterior distribution

$$p(\boldsymbol{y}_{1:T}; \boldsymbol{\theta}^{new}) = \int p(\boldsymbol{y}_{1:T}|\bar{\boldsymbol{w}})p(\bar{\boldsymbol{w}}; \boldsymbol{\theta}^{new})d\bar{\boldsymbol{w}}. \tag{8}$$

Figure 2 summarizes the workflow of the Interaction ProMP. During the training phase, imitation learning is used to learn the parameter $\boldsymbol{\theta}$. In the figure, the distribution modelled by $\boldsymbol{\theta}$ is abstracted as a bivariate Gaussian where each of the two dimensions are given by the distribution over the weights of the human and robot trajectories. During execution, the assistive trajectory of the robot is predicted by integrating out the weights of the posterior distribution $p(\bar{\boldsymbol{w}}; \boldsymbol{\theta}^{new})$. The operation of conditioning is illustrated by the slicing of the prior, at the current observation of the position of the human $\boldsymbol{y}_t^*$.

3.3 Estimating Phases and Actions of Multiple Tasks

Previous works [7, 13] have only addressed spatial variability, but not temporal variability of demonstrated movements. However, when demonstrating the same task multiple times, a human demonstrator will inevitably execute movements at different speeds, thus changing the phase at which events occur. Previously, this problem has been alleviated by introducing an additional pre-processing step on the training data for time-alignment based on a variant of DTW. Back to Fig. 1b, the aligned model is shown as the distribution of trajectories indexed by the normalized time.

Time-alignment ensures that the weights of each demonstration can be regressed using the same feature $\boldsymbol{\psi}_{1:T}$. As a consequence, during execution, the conditioning (6) can only be used when the phase of the human demonstrator coincides with the phase encoded by the time-aligned model, which is unrealistic in practice. In [7, 13] we avoided this problem by conditioning only at the last position of the human movement, since for this particular case, the corresponding basis function is known to be $\boldsymbol{\psi}_T$. For any other time step t, the association between $\boldsymbol{y}_t^*$ and the basis $\boldsymbol{\psi}_t$ is unknown when the human presents temporal variability and the velocity is either unobserved or computation from derivatives impractical due to sparsity of position measurements.

We propose incorporating the temporal variance as part of the model by learning a distribution over phases from the multiple demonstrations. This enriched model not only eliminates the need for time-alignment, but also opens the possibility for faster robot behaviour as the conditioning (6) can be applied before the end of the human movement. Initially, we replace the original time indexes of the basis functions with a phase variable $z(t)$. Define T_{nom} as a nominal trajectory duration (e.g. the average final time of the demonstrations) from which the weights of all demonstrations are regressed to obtain the parameters of the distribution $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$; the Gaussian bases are spread over the nominal duration $\boldsymbol{\psi}_{1:T_{nom}}$. Assuming that each of the i-th demonstrations has a constant rate of change, define a temporal scaling factor

$$\alpha_i = T_{nom}/T_i. \tag{9}$$

The single scaling factor α_i means that observations (the three red circles in Fig. 1b) are “stretched” or “compressed” at the same rate in the temporal direction. Although simple, our experiments have shown that this assumption holds in practice for simple, short stroke movements typical of handovers (see [6] for problems where multiple phases are addressed). Thus, a trajectory of duration T can be computed relative to the phase

$$p(\boldsymbol{y}_{1:T}|\boldsymbol{w}) = \prod_{1}^{T} \mathcal{N}(\boldsymbol{y}(z_t)|[\boldsymbol{\psi}(z_t)]^T \boldsymbol{w}, \boldsymbol{\Sigma}_y), \quad z_t = \alpha t. \tag{10}$$

Given the sparse partial sequence of human position observations, a posterior probability distribution over phases is given as

$$p(\alpha|\boldsymbol{y}^*_{t,t'}, \boldsymbol{\theta}) \propto p(\boldsymbol{y}^*_{t,t'}|\alpha, \boldsymbol{\theta})p(\alpha). \tag{11}$$

For simplicity, we assume the prior $p(\alpha)$ as a univariate Gaussian distribution, obtained from the M demonstrations, $\alpha \sim \mathcal{N}(\mu_\alpha, \sigma^2_\alpha)$. For a specific α value the likelihood is

$$\begin{aligned} p(\boldsymbol{y}^*_{t,t'}|\alpha, \boldsymbol{\theta}) &= \int p(\boldsymbol{y}^*_{t,t'}|\bar{\boldsymbol{w}}, \alpha)p(\bar{\boldsymbol{w}})d\bar{\boldsymbol{w}} \\ &= \mathcal{N}(\boldsymbol{y}^*_{t:t'}|[\mathbf{A}(z_{t:t'})]^T \boldsymbol{\mu}_w, [\mathbf{A}(z_{t,t'})]^T \boldsymbol{\Sigma}_w [\mathbf{A}(z_{t,t'})] + \boldsymbol{\Sigma}^*_y), \end{aligned} \tag{12}$$

where

$$[\mathbf{A}(z_{t,t'})]^T = \begin{bmatrix} [\boldsymbol{\psi}(z_{t,t'})]^T_1 & \dots & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & [\boldsymbol{\psi}(z_{t,t'})]^T_P \end{bmatrix}, \tag{13}$$

is the matrix of basis functions of the observed positions of the human, which corresponds to the observed entries of the full matrix $\mathbf{H}$ in (7), however, now indexed by the phase $z_t = \alpha t$. Given the observations $\boldsymbol{y}^*_{t,t'}$, the likelihood of each sampled α candidate is computed with (12), and the most probable scaling value

$$\alpha^* = \arg\max_\alpha p(\alpha|\boldsymbol{y}^*_{t,t'}, \boldsymbol{\theta}) \tag{14}$$

is selected. Intuitively, the effect of different phases is to stretch or compress the temporal axis of the prior (unconditioned) distribution proportionally to α. The method then compares which scaling value generates the model with the highest probability given the observation $\boldsymbol{y}^*_{t,t'}$. Once the most probable scaling value is found, its associated observation matrix $\mathbf{H}(z_{1:T})$ can be used in (6) to condition and predict the trajectories of both human and robot. To efficiently estimate the phase during execution, one approach is to sample a number of values of α from the prior $p(\alpha)$ and precompute and store, for each of them, the associated matrix of basis functions $\mathbf{A}(z_{1:T})$ and $\mathbf{H}(z_{1:T})$ beforehand.

In a multi-task scenario, we can now address the recognition of the task given the positions $\boldsymbol{y}^*_{t,t'}$. Assume a K number of collaborative tasks are encoded by independently trained Interaction ProMPs represented by the parameter $\boldsymbol{\theta}_k$. For each task k, the most probable α^*_k and likelihood must be stored. By noting that each task is represented by its own parameter $\boldsymbol{\theta}_k$, the most probable task is given by re-using the likelihoods of the set $\{\alpha^*_k, \boldsymbol{\theta}_k\}$, already computed in (14). The task recognition is given by

$$k^* = \arg\max_k p(k|\boldsymbol{y}^*_{t,t'}), \tag{15}$$

where $p(k|\boldsymbol{y}^*_{t,t'}) \propto p(\boldsymbol{y}^*_{t,t'}|\alpha^*_k, \boldsymbol{\theta}_k)p(k)$ with $p(k)$ being a prior probability distribution of the task and $p(\boldsymbol{y}^*_{t,t'}|\alpha^*_k, \boldsymbol{\theta}_k)$ was previously obtained in (12). The two optimizations in (14) and (15) lead to an algorithm that scales linearly in the number of sampled α's and in the number of tasks.

4 Experiments with a Semi-autonomous Robot

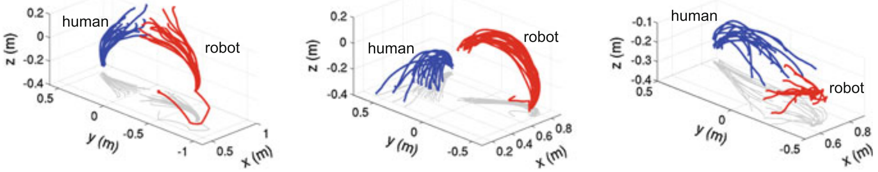
Collaborative assembly experiments were conducted using a 7-DoF lightweight arm equipped with a 5-finger hand. In all experiments, the wrist of the human was tracked by motion capture, providing XYZ Cartesian coordinates. The joint encoder trajectories of the robot were recorded by kinesthetic teaching. For each demonstration, the set of human-robot measurements were paired and stored with a sampling rate of 50 Hz.

4.1 A Multi-task Semi-autonomous Robot Coworker

As it was shown in Fig. 1a, we applied our method on a multi-task scenario where the robot plays the role of a coworker that helps a human assembling a toolbox. This scenario was previously proposed in [7] where the training data was time-aligned. As a consequence, the conditioning could only be applied at the end of the movement. Here, the robot can predict the collaborative trajectory before the human finishes moving, leading to a faster robot response. Moreover, the effort spent in pre-processing training data was considerably decreased as no time-alignment was needed.

The assembly consists of three different collaborative interactions. In one of them, the human extends his hand to receive a plate. The robot fetches a plate from a stand and gives it to the human by bringing it close to his hand. In a second interaction, the human fetches the screwdriver and the robot grasps and gives a screw to the human as a pre-emptive collaborator would do. The third type of interaction consists of the robot receiving a screwdriver such that the human coworker can have both hands free (the same primitive representing this interaction is also used to give the screwdriver back to the human). Each interaction of plate handover, screw handover and holding the screwdriver was demonstrated 15, 20, and 13 times, respectively. The trajectories obtained from the demonstrations are shown in Fig. 3 in the Cartesian space. Note, however, that the interaction primitives were trained on the Cartesian coordinates of the wrist with the joint coordinates of the robot.

To make a direct comparison between the previous method and the present method, the same training data presented in [7] was used. The durations of each demonstration was, however, randomly modified as we noticed the original data did not present sufficient variability of phases (in our initial tests the correct phase could be reasonably estimated with only 2–3 α samples). The randomization acts as a surrogate for different demonstrators with different speeds of execution. The original time-aligned demonstrations for one of the tasks can be seen in Fig. 4a as the several gray curves.



(a) Handing over a plate

(b) Handing over a screw

(c) Holding the screw driver

Fig. 3 Demonstrations of the three different interactions and their respective trajectories

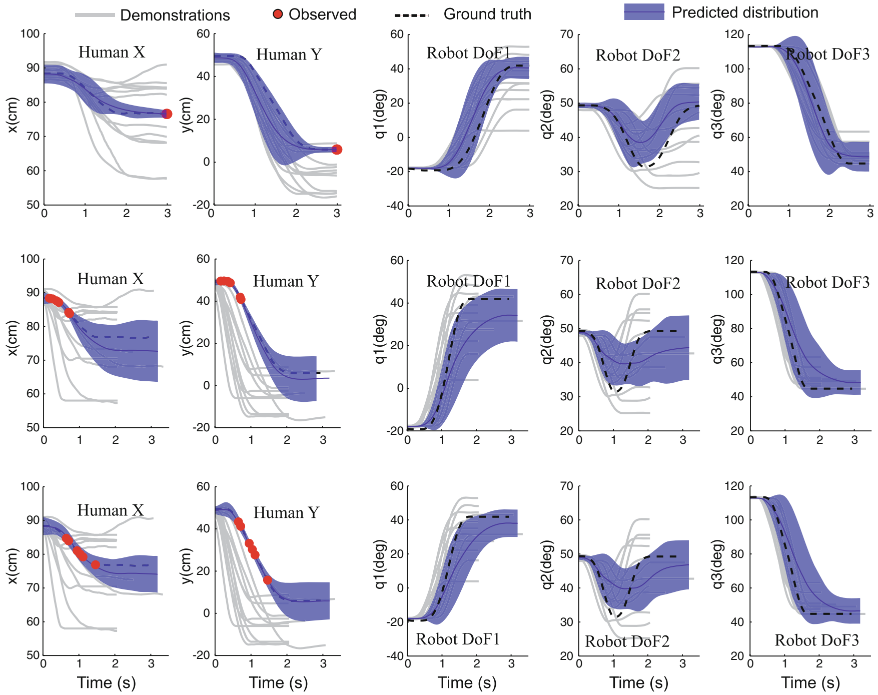


Fig. 4 Prediction of the distribution over trajectories of the human and the robot for the task of handing over a screw driver. **a** The previous method with time-aligned data without phase estimation and therefore conditioned only a the last observation of the human position. The phase estimation method where five measurements randomly spaced are taken up to 25% of the total duration of the human movement in (**b**) and 50% of the total duration of the human movement in (**c**)

Using leave-one-out cross-validation (LOOCV) the figure shows that the uncertainty of the human position collapses at the end, when the measurement is made on the test data. The posterior distribution, shown as the blue patch with ± two-standard deviations, predicts the robot final joint positions with an average error of 2.1 ± 6.7 degrees. Figure 4b, c shows the proposed method with phase-estimation where the training data includes various phases. In Fig. 4b, the observation represents 25% of the total trajectory length. The final positional error was of 6.8 ± 11.3 degrees.

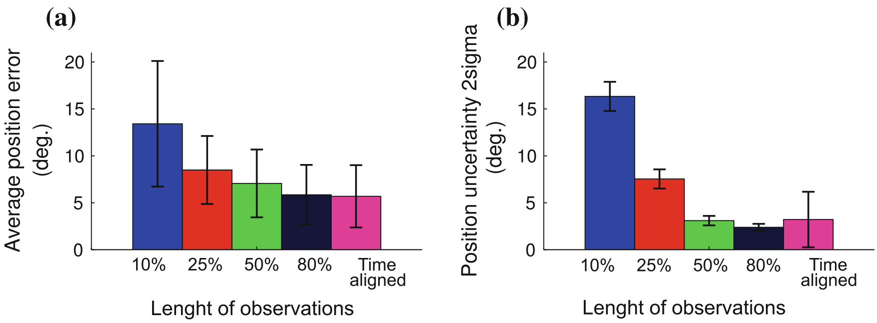


Fig. 5 Leave-one-out cross-validation over the whole training dataset of plate handovers. The final position errors (**a**) and the uncertainty ($\pm 2\sigma$) (**b**) of the predictions are shown. The predictions are made when 10, 25, 50, and 80 % of the trajectory are observed and compared with the time-aligned case

In (c), 50% of the trajectory was observed and the error decreased to 2.0 ± 5.7 degrees, roughly achieving the same accuracy as the time-aligned case shown in (a). The error was computed by averaging the RMS final position error over the 7 joint positions of the arm.

In Fig. 5a, each bar represents the final position error as the average over the 7 joints of the robot with LOOCV over the whole data set of demonstrations. The figure shows the cases when 10, 25, 50, and 80 % of the trajectory were observed. For each observation, 25 samples of the phase parameter α were used. We compared those results with the original method [7] when the prediction is made based on the final measurement, indicated by the bar labelled “Time aligned”. When 80 % of the trajectory was observed, the prediction provided the same accuracy of the time-aligned method.

From the same LOOCV test, the uncertainty at the final position (that is, the width of the blue patch previously shown in Fig. 4 at the end of the trajectory), was also quantified. These results are shown at the right plot Fig. 5b. Note that when 80% of observations were provided, a trajectory with less uncertainty than the time-aligned case can be predicted. This results from the fact that, with phase estimation, the covariance matrix is updated multiple times while in the time-aligned case only one single update is made at the end of the movement.

Interaction ProMPs also provide the ability for the robot to spatially coordinate its movement according to the movement of the human. A practical application is the handover of an object at different positions as shown in Fig. 6. In the left picture, the robot first receives the screwdriver from the human. In the right picture, the human extends his hand in a different location and the robot then delivers the tool back. The trajectory of the robot was inferred by conditioning the Interaction ProMP during the first second of the movement of the human. We have previously quantified the accuracy of the prediction in our setup achieving positional errors of less than 3 cm [13].

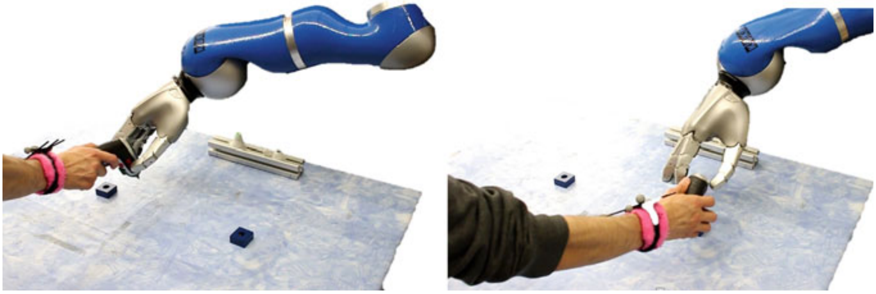


Fig. 6 Handover and return of a screwdriver at different positions, obtained by conditioning the Interaction ProMP on the positions of the wrist marker

With phase estimation, the robot reaction time for the handover of the screwdriver shown in Fig. 6 decreased on average by 2 s, a reduction of 25% of the task duration in relation to the original time-aligned case. Our preliminary evaluations on the assembly scenario was carried out by sampling 25 values of α phases for each of the three tasks, thus requiring 75 (25 samples × 3 tasks) calls to the computation of the probabilities (11) while the human moves his arm. The whole process, including the final prediction of the full trajectory with (8), observed during the first second of the human movement took in average 0.20 s using Matlab code on a conventional laptop (Core i7, 1.7 GHz). To control the robot, only the mean of the posterior distribution over trajectories for each joint of the robot was used, and tracked by the standard, compliant joint controller provided by the robot manufacturer.[2] A video of this experiment can be watched in http://youtu.be/4qDFv02xlNo.

4.2 *Discussion of the Experiment and Limitations*

In practice, there is an upper limit on the number of tasks and sampled α's that can be supported. This limit can be empirically evaluated as the total time required to compute the probability of all sampled alphas, for all tasks, which must be less than the duration of the human movement. Otherwise, it is faster to predict the robot trajectories based on the final measurement of the human, as it was done in previous works. This limit depends on the efficiency of the implementation and the duration of the human movement.

Since the experiments aimed at exposing the adaptation of the robot movement solely by means of the Interaction ProMPs, no direct feedback tracking of the marker on the human wrist was made. The interaction primitive framework may potentially benefit when used in combination with a feedback controller that tracks the markers

[2] Although not used in this paper, the ProMP framework also provides means to compute the feedback controller and the interested reader is referred to [15].

directly. Note, however, that it is not possible to completely replace an interaction primitive by a tracking controller. A feedback controller does not provide the flexibility and richness of the trajectories that can be encoded in a primitive learned from human demonstrations.

A system that allows for reliable estimation of velocity (or a constant stream of position) can greatly simplify the estimation of the phase, and under the assumption of a constant rate α, make the problem readily solvable. On the other hand, the nondisruptive deployment of semi-autonomous robots in the field must cope with occluded and sparse position measurements, often provided by low-cost sensors such as Kinect cameras, which requires algorithms that are capable of estimating the phase from such data. In the short term we envision a self-contained setup that uses a Kinect camera as a replacement of the optical marker tracking system that was used during the experiments.

5 Conclusions

This paper presented a method suited for collaborative and assistive robots whose movements must be coordinated with the trajectories of a human partner moving at different speeds. This goal was achieved by augmenting the previous framework of Interaction ProMPs with a prior model of the phases of the human movement, obtained from demonstrated trajectories. The encoding of phases enriches the model by allowing the alignment of the observations of the human in relation to the interaction model, under an intermittent positional stream of data. We experimentally evaluated our method in an application where the robot acts as a coworker in a factory. Phase estimation allowed our robot to predict the trajectories of both interacting agents before the human finishes the movement, resulting in a faster interaction. The duration of a handover task could be decreased by 25% while using the same robot commands (same speed of the robot movement).

A future application of the method is to use the estimated phase of the human to adapt the velocity of the robot. A slowly moving human suggests that the robot should also move slowly, as an indication that a delicate task is being executed. Conversely, if the human is moving fast, the robot should also move fast as its partner may want to finish the task quickly.

Acknowledgements The research leading to these results has received funding from the European Community's Seventh Framework Programmes (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand) and (FP7-ICT-2009-6) under grant agreement 270327 (ComPLACS); and from the project BIMROB of the Forum für interdisziplinäre Forschung (FiF) of the TU Darmstadt.

References

1. Ben Amor, H., Neumann, G., Kamthe, S., Kroemer, O., Peters, J.: Interaction primitives for human-robot cooperation tasks. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2014)
2. Calinon, S., Sauser, E.L., Billard, A.G., Caldwell, D.G.: Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2671–2676 (2010)
3. Calinon, S., Li, Z., Alizadeh, T., Tsagarakis, N.G., Caldwell, D.G.: Statistical dynamical systems for skills acquisition in humanoids. In: Proceedings of the IEEE/RAS International Conference on Humanoids Robots (HUMANOIDS), pp. 323–329 (2012)
4. Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: Proceedings of the 25th International Conference on Machine Learning (ICML), pp. 144–151. ACM (2008)
5. Englert, P., Toussaint, M.: Reactive phase and task space adaptation for robust motion execution. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 109–116 (2014)
6. Ewerton, M., Maeda, G., Peters, J., Neumann, G.: Learning motor skills from partially observed movements executed at different speeds. In: Accepted: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2015)
7. Ewerton, M., Neumann, G., Lioutikov, R., Ben Amor, H., Peters, J., Maeda, G.: Learning multiple collaborative tasks with a mixture of interaction primitives. In: Proceedings of the International Conference on Robotics and Automation (ICRA), pp. 1535–1542 (2015)
8. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural Comput. **25**(2), 328–373 (2013)
9. Kim, S., Gribovskaya, E., Billard, A.: Learning motion dynamics to catch a moving object. In: Proceedings of the IEEE/RAS International Conference on Humanoids Robots (HUMANOIDS), pp. 106–111 (2010)
10. Kim, S., Shukla, A., Billard, A.: Catching objects in flight. IEEE Transactions on Robotics (TRO) **30** (2014)
11. Koppula, H.S., Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. In: Robotics: Science and Systems (2013)
12. Lee, D., Ott, C., Nakamura, Y.: Mimetic communication model with compliant physical contact in human-humanoid interaction. Int. J. Robot. Res. **29**(13), 1684–1704 (2010)
13. Maeda, G., Ewerton, M., Lioutikov, R., Ben Amor, H., Peters, J., Neumann, G.: Learning interaction for collaborative tasks with probabilistic movement primitives. In: Proceedings of the International Conference on Humanoid Robots (HUMANOIDS), pp. 527–534 (2014)
14. Mainprice, J., Berenson, D.: Human-robot collaborative manipulation planning using early prediction of human motion. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 299–306. IEEE (2013)
15. Paraschos, A., Daniel, C., Peters, J., Neumann, G.: Probabilistic movement primitives. In: Advances in Neural Information Processing Systems (NIPS), pp. 2616–2624 (2013)
16. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. **26**(1), 43–49 (1978)
17. Schaal, S.: Is imitation learning the route to humanoid robots? Trends Cogn. Sci. **3**(6), 233–242 (1999)
18. Tanaka, Y., Kinugawa, J., Sugahara, Y., Kosuge, K.: Motion planning with worker's trajectory prediction for assembly task partner robot. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1525–1532. IEEE (2012)
19. Van Den Berg, J., Miller, S., Duckworth, D., Hu, H., Wan, A., Fu, X., Goldberg, K., Abbeel, P.: Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2074–2081 (2010)

20. Vuga, R., Nemec, B., Ude, A.: Speed profile optimization through directed explorative learning. In: Proceedings of the IEEE/RAS International Conference on Humanoids Robots (HUMANOIDS), pp. 547–553. IEEE (2014)
21. Yamane, K., Revfi, M., Asfour, T.: Synthesizing object receiving motions of humanoid robots with human motion database. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 1629–1636. IEEE (2013)

Averaged Anchoring of Decoupled Templates in a Tail-Energized Monoped

Avik De and Daniel E. Koditschek

1 Introduction

Dimension reduction enjoys a long analytical tradition in dynamical systems theory generally [1] and locomotion particularly [2], but the synthesis of complex, high dimensional dynamical behavior from simple low dimensional components is far less considered. Notwithstanding some early examples of anchored templates [3] in physical [4, 5] and numerical [6, 7] studies, the only presently available systematic account of how to embed low dimensional target dynamics in higher dimensional mechanical bodies entails inverse dynamics along the quotient space down to the attracting submanifold [8, 9]. Yet well before any of this work, Raibert had already shown empirically how to synthesize stable one-, two-, and four-legged spatial running behavior by parallel composition of simple, decoupled, one degree of freedom vertical, horizontal, and rotational feedback laws [10].

Motivation and contributions This paper builds on and significantly advances the ideas of [11, 12] to achieve what we believe to be the first formal account of such anchor synthesis via template composition. We examine an approximate model of a tailed monoped that exhibits stable sagittal plane translational hopping motion energized by tail pumping during stance. The tail controller excites its actuator by applying a new purely vertical hopping regulator [13] to the the robot's shank oscillation phase. The stepping controller adjusts the next leg touchdown angle by applying a modified active rimless wheel speed regulator [14] to the angular momentum of the robot's mass center relative to the pinned toe. We show that both the vertical and fore-aft regulators, acting alone on their respective one degree of freedom template plants in isolation, succeed in stabilizing hopping height and horizontal speed.

A. De (✉) · D.E. Koditschek
University of Pennsylvania, Philadelphia, PA, USA
e-mail: avik@seas.upenn.edu

D.E. Koditschek
e-mail: kod@seas.upenn.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_16

We then show that this parallel composition of template controllers stabilizes the hopping translational gait of the highly coupled approximate model.[1]

This paper introduces as well (to the best of our knowledge for the first time) in the locomotion literature the dynamical systems methodology of averaging. Harnessing specific symmetries in motion and force played a major role in Raibert's compositional methods [10] and we express his intuition here by integrating them out (or in) along the stance phase. Such appeal to averaging symmetric dynamical influences gains its theoretical traction [1, 15] because our mechanical system executes periodic orbits in its locomotory steady state along which (loosely speaking) the energy levels of various compartments of the dynamics evolve at a much slower time-scale than does the phase. The right confluence of these circumstances has the fortunate effect of insuring not only that the averaged dynamics closely approximate the original but guaranteeing that their steady state (limit cycle stability) properties are identical. Recourse to averaging, in turn, motivates a new, relaxed version of the template-anchor relation [3]. We develop this new idea using our motivational example in Sect. 4, but we leave a formal definition to future work.

Organization and results We first develop some mathematical tools for averaging a particular class (Definition 1) of hybrid systems in Sect. 2. Proposition 1 extends the classical averaging result [1] to our application domain. An application-focused reader may skip this section initially and refer back at its usage in Sects. 3–4.

In Sect. 3, we present two classical 1DOF template systems as phase-averageable hybrid systems: the vertical hopper [13] and the rimless wheel [16] (modified to include some control affordance [14]). We present template controllers for these template systems (inspired by [10, 17]) that provably stabilize the 1DOF plants.

In Sect. 4, we show how to take the parallel composition of these two simple constituents and apply it to a 3DOF model of the tailed SLIP (an approximation to a physical system that we have built). We use averaging theory to demonstrate that the desired steady state hopping behavior is indeed rendered asymptotically stable in this highly coupled 3DOF model.

In Sect. 5 we show simulation evidence for our theoretical claims, and data from a physical hopping machine showing (in spite of our approximations in the model of Sect. 4) qualitative behavior quite similar to the simulation results.

The contributions of this paper can be summarized as (a) introduction of a new notion of "average anchoring," and an extension of classical averaging theory (Proposition 1) to a relevant class of hybrid systems, (b) "unpacking" the classic SLIP template [18] as a "cross-product" (in our relaxed sense of anchoring) of a vertical heartbeat [13] and a rimless wheel [16], and (c) a proof of stability of compositional control using decoupled controllers in a coupled 3DOF tailed SLIP model system.

A note on notation The symbol $a_\star$ refers to an "energy" coordinate for subsystem $\star$ (not necessarily mechanical energy), which changes slowly relative to the phase of

[1] The chief advances beyond the initial composition study of [11] are that: (i) we achieve a stability proof of the steady state gait limit cycle for the 4DOF closed loop tailed biped (Fig. 1); and (ii) we are able to do so for a far less restricted range of tail-energizing energies (17).

Table 1 References to decoupled controllers and important ODE's and vector fields

	Vertical hopper	Rimless wheel	Anchor
Controller	u_{v} (8)	$u_{\mathsf{h}}(a_{\mathsf{h}})$ (13)	$\approx u_{\mathsf{v}}(t) \times u_{\mathsf{h}}(a_{\mathsf{h}})$ (16), (23)
Dynamics	$\ddot{r}$ (5), f_{v} (9)	$\ddot{\theta}$, f_{h} (11)	$\ddot{\mathbf{q}}$ (15)
Averaged dynamics	$\overline{f}_{\mathsf{v}}$ (10)	$\overline{f}_{\mathsf{h}} = f_{\mathsf{h}}$ (11)	$\overline{f} = \overline{f}_{\mathsf{v}} \times \overline{f}_{\mathsf{h}}$ (19), (22)

the system (Definition 1). The bold subscript is also used for other symbols pertaining to a specific subsystem; e.g. k_{v} is the gain chosen for the **v**ertical subsystem. We have collected references to important equations and symbols in Table 1.

2 Hybrid Averaging

For our analysis in this paper will focus on hybrid systems with a single domain. We specialize the general definition in [19], of a hybrid system with 1 domain, to add properties conducive to averaging (Proposition 1).

Definition 1 Given $\varepsilon > 0$ (a "separation of time scales" parameter quantifying slow energy-phase dynamics), an *averageable hybrid system* is a tuple $(\mathcal{A}, f, \mathcal{R})$ with

(i) a topological disk $\mathcal{A} \approx \mathbb{R}^d$ as the "energy" domain,

(ii) a C^2 non-autonomous ("phase-varying") vector field $f : \mathcal{A} \times [0, T_s] \times \mathbb{R}_+ \to T\mathcal{A}$, where $T_s > 0$, such that the system dynamics are

$$\dot{x} = \varepsilon f(x, t, \varepsilon), \text{ and} \tag{1}$$

(iii) an energy reset map, $\mathcal{R} : \mathcal{A} \to \mathcal{A}$ that is ε-close to the identity in the sense that there is some array-valued smooth map, $S(x)$ such that the equilibrium of f is also a fixed point of R, and the Jacobian of $\mathcal{R}$ is $\mathrm{D}\mathcal{R} = I + \varepsilon S$, and $S(x) + T_s \mathrm{D} f(x)$ is full rank at each x.

Remark Though the definition above is sufficient for this paper, we observe that the triple $(\mathcal{A}, f, \mathcal{R})$ is a mere specialization of a hybrid system as defined in [19, Definition 1]. To see how, think of decoupled "phase" dynamics $\dot{\psi} = \omega$ being appended to the state, yielding a standard hybrid system $H = (D, F, G, R)$, where $D := \mathcal{A} \times [0, T_s]$, $F := \begin{bmatrix} \omega f \\ \omega \end{bmatrix}$, $G := \mathcal{A} \times \{T_s\}$, $R := \begin{bmatrix} \mathcal{R} \\ 0 \end{bmatrix}$ with a single hybrid mode. The last condition above is difficult to check, and we leave a more complete generalization of this result to future work.

The "averaging" technique [1] can be extended to a method of approximating (with error bounds) solutions of the T_s-periodic system (1) with an *averaged hybrid system* by replacing f in (1) with

$$\frac{dx}{dt} = \frac{\varepsilon}{T_s}\int_0^{T_s} f(x,t,0)\,dt =: \varepsilon\overline{f}(x), \tag{2}$$

with the same decoupled phase, guard set and reset map.

Additionally, the Poincaré return maps for (1) and (2) can both be defined as usual, using $\mathcal{A} \times \{T_s\}$ as the section.

Proposition 1 (Hybrid averaging) *Let* $(\mathcal{A}, f, \mathcal{R})$ *be a averageable hybrid system. If* p_0 *is a hyperbolic fixed point of an equilibrium of (2), then there exists* $\varepsilon_0 > 0$ *such that for all* $0 < \varepsilon \le \varepsilon_0$*, (1) possesses a unique hyperbolic periodic orbit* $r(p_0) + \mathcal{O}(\varepsilon)$*, of the same stability type as* p_0*.*

Proof (adapted from [1]) Our system satisfies all conditions required for the proof of Theorem 4.1.1(i) in [1] on the set $t \in (0, T_s]$. We conclude that if at $\psi = 0^+$ (just after the reset), $|x_0 - y_0| = \mathcal{O}(\varepsilon)$, then $|x(t) - y(t)| = \mathcal{O}(\varepsilon)$ on a time scale $t \sim 1/\varepsilon$. Construct the change of coordinates $x = y + \varepsilon w(y, t, \varepsilon)$, as in [1], so that (1) becomes

$$\frac{dy}{dt} = \varepsilon\overline{f}(y) + \varepsilon^2 f_1(y, t, \varepsilon), \tag{3}$$

where f_1 is a lumped $\mathcal{O}(\varepsilon^2)$ "remainder term."

With the cross-section $\Sigma := \{(y,t) : t = T_s\}$, and $U \subset \Sigma$, an open set, the Poincaré maps $P_0 : U \to \Sigma$ and $P_\varepsilon : U \to \Sigma$ can be defined as usual (reset composed by a flow for time Ts) for the systems (2) and (3), respectively. P_ε is ε^2-close to P_0, since (a) r is the same for both, and (b) the "time to impact" is fixed and independent of ε (decoupled phase dynamics).

Since p_0 is a hyperbolic fixed point of P_0, it is also a hyperbolic fixed point of $\mathrm{D}P_0(p_0) = e^{\varepsilon T_s \mathrm{D} f(\mathcal{R}(p_0))}\mathrm{D}\mathcal{R}(p_0)$ ($\mathrm{D}\mathcal{R}$ is full rank for diffeomorphism $\mathcal{R}$). Therefore, substituting $\mathrm{D}r = I + \varepsilon S$, the matrix

$$\begin{aligned}\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}(\mathrm{D}P_0(p_0) - I) &= \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}(I + \varepsilon T_s \mathrm{D} f|_{\mathcal{R}(p_0)})\mathrm{D}\mathcal{R}(p_0) - I\\ &= S + T_s \mathrm{D} f|_{\mathcal{R}(p_0)}\end{aligned} \tag{4}$$

is invertible (from the resonance-free condition of Proposition 1). Since P_ε is ε-close to P_0, $\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}(\mathrm{D}P_\varepsilon(p_0) - I)$ is the same as the RHS of (4). Hence, using the implicit function theorem we see that the rank of $\frac{1}{\varepsilon}(\mathrm{D}P_\varepsilon(p_0) - I)$ form a smooth curve $(p_\varepsilon, \varepsilon)$. Now p_ε is a fixed point of P_ε, and the local Taylor expansion is $p_\varepsilon = p_0 + \mathcal{O}(\varepsilon)$. The eigenvalues of $\mathrm{D}P_\varepsilon(p_\varepsilon)$ are ε^2-close to those of $\mathrm{D}P_0(p_0)$, since $\mathcal{R}$ is a diffeomorphism, and

$$\mathrm{D}P_\varepsilon = [\exp(\varepsilon T_s \mathrm{D} f|_{\mathcal{R}(p_0)}) + \mathcal{O}(\varepsilon^2)] \cdot \mathrm{D}\mathcal{R}(p_0).$$

So, (3) has a periodic orbit ε-close to $\mathcal{R}(p_0)$, and by the coordinate change (3), (1) has a similar orbit. The stability properties of the periodic orbits of (1) and (2) are identical since $\mathrm{D}P_\varepsilon(p_\varepsilon)$ is ε^2-close to $\mathrm{D}P_0(p_0)$.

3 Templates as Constituents of Planar Hopping

Based on the motivation in Sect. 1, we attempt to develop a planar hopping behavior as a composition of (a) a vertical hopper, as empirically demonstrated by [10] and studied in [11, 13], and (b) a rimless wheel [16] with liftoff impulses [14]. As Fig. 1 notionally shows, these templates are "anchored" (Sect. 4) in the Tailed SLIP model our physical platform (Sect. 5) closely resembles.

3.1 Controlled 1DOF Vertical Hopper

Our template plant for vertical hopping is a point mass connected to a vertically-constrained series-elastic damped leg (Fig. 1, top left). As shown in [11], this 1DOF system can be stabilized to arbitrary hopping heights by recourse to an appropriately designed phase-locked, oscillatory pattern of energetic excitation entering through the shank (note the contrast to the hybrid energization strategy of [10, 13]). We impose the following additional assumption and follow [11]:

Assumption 1 (*Weakly nonlinear oscillator*) The vertical hopper is an LTI spring-mass-damper system actuated by a possibly nonlinear $\mathcal{O}(\varepsilon)$ force, where $\varepsilon \ll \omega$.

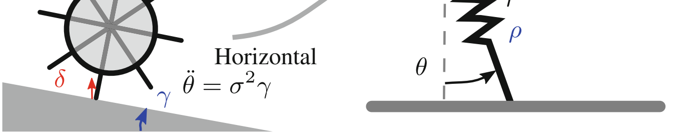


Fig. 1 The vertical hopper [13] and rimless wheel [16] template plants (*left*), and their corresponding equations of motion, and (*right*) the tailed SLIP model, labeled with configuration variables (*black*), actuators (*red*), and model parameters (*blue*)

Stance dynamics Define the normalized leg length $r := \tilde{r} - \rho + mg/\omega^2$, where $\tilde{r}$ is the physical length of the spring with rest length ρ. As in [11], the dynamics of a linear oscillator with small (Assumption 1) damping are given by

$$\ddot{r} + \omega^2 r = -u_\mathsf{v} - \varepsilon\beta\dot{r} \implies \dot{x} = -\omega \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ -u_\mathsf{v}/\omega - \varepsilon\beta x_2 \end{bmatrix}, \tag{5}$$

where $x := \begin{bmatrix} r \\ \dot{r}/\omega \end{bmatrix}$. Define the *vertical energy* $a_\mathsf{v} := \|x\|$, and note that by Assumption 1, for $u_\mathsf{v} = \mathcal{O}(\varepsilon)$, the corresponding phase coordinate is trivial; if $\tan \angle x := -x_1/x_2$ (such that "stance" is from $\angle x = 0$ to π), then

$$\frac{d}{dt}\angle x = -\omega - \frac{\varepsilon\beta x_1 x_2}{a^2{}_\mathsf{v}} - \frac{u_\mathsf{v} x_1}{a^2{}_\mathsf{v}\omega} = -\omega + \mathcal{O}(\varepsilon). \tag{6}$$

Thus the phase evolution is in the form of a perturbation problem [15, (10.1)]. The nominal system, $\frac{d}{dt}\angle x = -\omega$ has a trivial solution $(\angle x)_{\text{nom}} = -\omega t$. Using [15, Theorem 10.1], we conclude that for small ε, the perturbed system has a unique solution of the form $\angle x = -\omega t + \mathcal{O}(\varepsilon)$. Note that no "constant of integration" appears since at $t = 0$ (touchdown), $r(0) = 0$, $\dot{r}(0) < 0$, and so $\angle x(0) = 0$. Consequently,

$$\begin{aligned} x_2 &= a_\mathsf{v} \cos \angle x = a_\mathsf{v} \cos(-\omega t + \mathcal{O}(\varepsilon)) \\ &= a_\mathsf{v}\big(\cos \omega t \cos(\mathcal{O}(\varepsilon)) + \sin \omega t \sin(\mathcal{O}(\varepsilon))\big) = a_\mathsf{v}(\cos \omega t + \mathcal{O}(\varepsilon)), \end{aligned} \tag{7}$$

since $\cos(\mathcal{O}(\varepsilon)) = 1 - \mathcal{O}(\varepsilon^2)$ and $\sin(\mathcal{O}(\varepsilon)) = \mathcal{O}(\varepsilon)$. Following [11], set

$$u_\mathsf{v} := -\varepsilon k_\mathsf{v} \omega \cos \omega t, \tag{8}$$

and observe that

$$\dot{a}_\mathsf{v} = \varepsilon f_\mathsf{v}(a_\mathsf{v}, t, \varepsilon) := \frac{\varepsilon x_2}{a_\mathsf{v}} (k_\mathsf{v} \cos \omega t - \beta x_2) \tag{9}$$

$$\approx \varepsilon \overline{f}_\mathsf{v}(a_\mathsf{v}) := \frac{\varepsilon}{\pi/\omega} \int_0^{\pi/\omega} f_\mathsf{v}(a_\mathsf{v}, t, 0)\, dt = \frac{\varepsilon}{2}(k_\mathsf{v} - \beta a_\mathsf{v}), \tag{10}$$

where the bottom row corresponds to a usage of Proposition 1 on the limit cycle, and we used (7) to eliminate $\mathcal{O}(\varepsilon^2)$ terms.

Hybrid guard and reset For this system and our periodic forcing (8), we observe that the dynamics (9) are π/ω-periodic. Roughly following [13], we collapse the "flight map" to a reflection of the vertical velocity, or $\mathcal{R}_\mathsf{v}(a_\mathsf{v}) = a_\mathsf{v}$, and the guard set is $G_\mathsf{v} = \{x_1 = 0\}$ (lift-off occurs approximately at adjusted rest length).

Return map stability From the simple from (10), it is easy to see that the averaged hybrid system has a stable limit cycle at $a_\mathsf{v} = k_\mathsf{v}/\beta$, and hence by Proposition 1, we can conclude that the vertical hopper plant (5) has a stable limit cycle.

3.2 Controlled Forward Speed: 1DOF Active Rimless Wheel

Our conceptual physical template for an isolated fore-aft system stabilized by "stepping" is inspired by the rimless wheel due to [16] (parameters defined in Fig. 1), but now fitted out with a controllable liftoff impulse, δ (inspired by [14]).

Assumption 2 (*Gravity approximation*) As in [20], we assume that the effect of gravity is invariant to leg angle.

The conservative approximation is useful in our search for *sufficient* conditions for an analytical stability proof (Proposition 3), but we empirically observe stability with large leg angles (Sect. 5), indicating that it is not necessary.

Stance dynamics The implication of Assumption 2 on the rimless wheel (physically manifested when the interspoke angle is small compared to the slope angle) is that angular acceleration is roughly constant through stance. Using notation from [16], and angular momentum about the toe $a_{\mathsf{h}} := \ell^2\dot{\theta}$ as the *horizontal energy*,

$$\ddot{\theta} = \sigma^2\gamma \implies \dot{a}_{\mathsf{h}} = \varepsilon f_{\mathsf{h}} := \sigma^2\gamma/\ell^2, \tag{11}$$

where σ is a dimensionless frequency and ℓ is the spoke length (Fig. 1) [16]. We observe that the right hand side is a constant, hence bounded.

Reset From [16, (5)], the rimless wheel has a restitution coefficient $\eta := 1 - \sigma^2(1 - \cos 2\alpha_0)$. The controlled reset map (with our modification to [16, (3)]) is

$$a^{+}{}_{\mathsf{h}} = a_{\mathsf{h}}\eta + \delta = a_{\mathsf{h}} + u_{\mathsf{h}}, \tag{12}$$

where in the last equality, we parameterize the liftoff impulse as $\delta = (1-\eta)a_{\mathsf{h}} + u_{\mathsf{h}}$ for notational convenience. Using the discrete proportional controller (suggested by the "stepping" controller [10]) with $k_{\mathsf{h}} \ll 1$,

$$u_{\mathsf{h}}(a_{\mathsf{h}}) = k_{\mathsf{h}}(a^{*}{}_{\mathsf{h}} - a_{\mathsf{h}}), \tag{13}$$

the controlled reset map becomes $\mathcal{R}_{\mathsf{h}}(a_{\mathsf{h}}) = a_{\mathsf{h}} + k_{\mathsf{h}}(a^{*}{}_{\mathsf{h}} - a_{\mathsf{h}})$. The guard set is $\{\theta = -\alpha_0\}$.

Return map stability The integrable stance dynamics (11) together with the reset map above give us the closed-form return map

$$P_{\mathsf{h}}(a_{\mathsf{h}}) := a_{\mathsf{h}} + k_{\mathsf{h}}(a^{*}{}_{\mathsf{h}} - a_{\mathsf{h}}) + \nu_0, \tag{14}$$

where $\nu_0 := \int_0^{T_s} \sigma^2\gamma/\ell^2\, dt$ is the leg sweep over a fixed stance duration T_s. The Jacobian is $\mathrm{D}P_{\mathsf{h}}(a_{\mathsf{h}}) = 1 - k_{\mathsf{h}}$, and so the system will stabilize at the fixed point $a_{\mathsf{h}} = a^{*}{}_{\mathsf{h}} + \nu_0/k_{\mathsf{h}}$.

Remarks Both the continuous (11) and discrete (14) parts of the dynamics exhibit a θ-equivariance, and since an encoding of a planar hopping task [10, Chap. 2] is also

θ-invariant, we work in the quotient space with coordinates given by the simple projection $(\theta, \dot{\theta}) \mapsto a_{\mathsf{h}}$, mapping the second order system (11) to an energy coordinate without a corresponding phase.

4 Average Anchoring in 3DOF Tail-Energized SLIP

The classical notion [3] of anchoring calls for the template dynamics (blue arrows in the top picture of Fig. 2) to be embedded in that of the anchoring body as an attracting invariant submanifold (red arrows in that same picture) of the left hand with conjugate restriction dynamics (depicted by the red lining up with blue arrows on the embedded submanifold). Averaging theory guarantees that the anchor must have a stable cycle (red arrows in the bottom picture of Fig. 2) that is close to an embedding of the template's (blue arrows, again, in the bottom picture), but the actual unaveraged (time-varying) anchor will in general have no related invariant manifolds nor dynamical conjugacy.

Our candidate "anchoring" body is a planar pogo stick with a light tail (Fig. 1), where the actuator τ_t is connected between the leg and the tail. In stance phase, the toe of the springy leg is pinned to the ground, and as a result, the robot is dynamically constrained to 3-DOF. Even though the body has two masses, the dynamics are somewhat simplified by the following assumption:

Assumption 3 (*Light tail*) The tail mass is small, i.e. $m_t \ll m$ (such that the center of mass is configuration-independent), but the tail inertia, $i_t := m_t \rho_t^2$, is significant.

Looking ahead to (15), this assumption allows us to use the tail as an inertial actuator on the leg spring through τ_t even though $m_t \ll m$, while avoiding Coriolis forces due to motion of the center of mass.

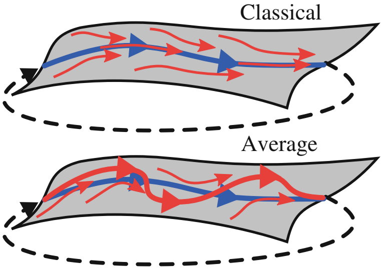


Fig. 2 A cartoon depicting the classical notion of anchoring [3] (*top*) and the new notion introduced in this paper based on averaging (*bottom*) as conceptual mechanisms of dimension reduction (Sect. 1), where the *blue line* represents a *template* flow, and the *red lines* depict flows on the anchoring body

4.1 Average Invariance of Template Flows in Stance Dynamics

In this subsection, we examine the stance behavior of the coupled tailed monoped in steady-state operation (we examine stability in the next subsection). The present method of proof requires (what simulation and empirical results suggest to be) a conservative assumption of operation in a slow fore-aft speed regime:

Assumption 4 (*Slow fore-aft speed*) The leg angular velocity $\dot{\theta} = \mathcal{O}(\varepsilon)$, or equivalently, the leg sweep is $\mathcal{O}(\varepsilon)$ of π/ω.

Additionally, we make a physically reasonable assumption about leg deflection:

Assumption 5 The leg deflection relative to its rest length is $a_{\mathrm{v}}/\rho = \mathcal{O}(\varepsilon)$.

Stance dynamics We use Assumption 3 (specifically, taking a limit $m_t/m \to 0$ with finite i_t) in a Lagrangian formulation with variables as in Fig. 1, to get the following stance dynamics (3 of the 4 DOF's in [12, (39)]):

$$\begin{aligned}
\ddot{r} &= -\omega^2 r - \varepsilon\beta\dot{r} + \widetilde{r}\dot{\theta}^2 - \frac{\tau_t \cos\xi}{\rho_t m} + g(1-\cos\theta),\\
\ddot{\theta} &= -\frac{2\dot{r}\dot{\theta}}{\widetilde{r}} - \frac{\tau_t \sin\xi}{\rho_t m \widetilde{r}} - \frac{g}{\widetilde{r}}\sin\theta,\\
\ddot{\xi} &= \tau_t/i_t, \qquad (15)
\end{aligned}$$

where ξ is the leg-tail angle (between which joints τ_t acts directly), r is the normalized leg extension as in Sect. 3.1, and β is now the mass-specific damping coefficient. Our specific usage of Assumption 3 in the equations of motion is that

- we assert that $m_b + m_t \approx m_b$, resulting in the dropping of Coriolis forces resulting from the configuration-dependent CoM,
- the tail can now be thought of as an inertial source of reaction forces on the $(\ddot{r}, \ddot{\theta})$ terms (which have familiar "SLIP-like" dynamics), as seen in the τ_t terms coupled through the physical tail angle, ξ).

We introduce as a stance controller for the coupled plant (15) a scaled version of the simple stance controller for the isolated vertical system (8),

$$\tau_t := m\rho_t u_{\mathrm{v}} = \varepsilon k_{\mathrm{v}}\omega m\rho_t \cos\omega t. \qquad (16)$$

Decoupled tail excitation The tail dynamics, $\ddot{\xi}$, take the form of a simple double integrator (15). When driven by the template controller (Table 1), and reset to $\xi(0) = -\frac{\varepsilon k_{\mathrm{v}} m\rho_t}{\omega i_t}$ during flight, the tail trajectory is

$$\xi(t) = -\frac{\varepsilon k_{\mathrm{v}} m\rho_t}{\omega i_t}\cos\omega t =: -a_{\mathrm{t}}\cos\omega t, \qquad (17)$$

where, for the purposes of design, we construe the amplitude of the tail oscillation as the energy contained in the tail compartment. Note that this represents a substantial generalization and thus a significant advance beyond the strict assumptions in [11] regarding the analysis of a similar planar hopping model: in this paper we allow for finite (and possibly large) tail motions, which couple into the other DOFs (15).

Proposition 2 *The averaged tailed SLIP stance vector field—but not the unapproximated one, obtained from (15)—is conjugate to a cross product of the averaged template vector fields (10), (11).*

Proof Because it is a decoupled double integrator, the feedforward influence of the tail subsystem (the output of the last row of (15), on the (r, θ) dynamics) can be represented as an independent time varying disturbance input, ξ, given in (17).

In the radial dynamics from (15), as in Sect. 3.1, let $x := \begin{bmatrix} r \\ \dot{r}/\omega \end{bmatrix}$. Reusing Assumption 1, the phase of the vertical hopper is interchangeable with time, $\angle x = -\omega t$. Similar to (9), setting $a_{\mathsf{v}} := \|x\|$, and using (17),

$$\dot{a}_{\mathsf{v}} = \varepsilon f_{\mathsf{v}}(a_{\mathsf{v}}, t, \varepsilon) = \varepsilon \cos^2 \omega t \, (k_{\mathsf{v}} \cos(-a_{\mathsf{t}} \cos \omega t) - \beta a_{\mathsf{v}}) + \mathcal{O}(\varepsilon^2) \tag{18}$$

$$\approx \varepsilon \overline{f}_{\mathsf{v}}(a_{\mathsf{v}}) = \frac{\varepsilon}{\pi/\omega} \int_{\mathcal{S}} f_{\mathsf{v}}(a_{\mathsf{v}}, t, 0)\, dt = \frac{\varepsilon}{2}(\widetilde{k}_{\mathsf{v}} - \beta a_{\mathsf{v}}), \tag{19}$$

where the $\mathcal{O}(\varepsilon^2)$ term in (18) consists of the following terms from the first row of (15), and the vertical "phase lock error" (7):

$$\widetilde{r}\dot{\theta}^2 + g(1 - \cos\theta) - \varepsilon\beta a_{\mathsf{v}} \mathcal{O}(\varepsilon), \tag{20}$$

and use Assumption 4 to get $\dot{\theta}^2 = \mathcal{O}(\varepsilon^2)$ as well as $1 - \cos\theta \approx \theta^2/2 = \mathcal{O}(\varepsilon^2)$.

In (19), $\widetilde{k}_{\mathsf{v}} := k_{\mathsf{v}}(2J_1(a_{\mathsf{t}})/a_{\mathsf{t}} - J_2(a_{\mathsf{t}}))$, and J_i are Bessel functions of the first kind [21] that act as an "attenuation" of our vertical gain k_{v} (bear in mind that a_{t}, introduced in (17) is a designed energizing magnitude term held constant for the duration of each stance), as the tail sweep gets larger (cf. Fig. 3).

In the leg-angle dynamics, define $a_{\mathsf{h}} := \widetilde{r}^2\dot{\theta}$ as in the template (Sect. 3.2), and use Assumptions 2, and 5 in the averaging step, to get:

$$\dot{a}_{\mathsf{h}} = \varepsilon f_{\mathsf{h}}(a_{\mathsf{h}}, t, \varepsilon) = -\varepsilon k_{\mathsf{v}} \omega \cdot m\rho_t \cos\omega t \left(\frac{\rho(1 - a_{\mathsf{v}}/\rho \sin\omega t)\sin\xi}{m\rho_t} \right) \tag{21}$$

$$\approx \varepsilon \overline{f}_{\mathsf{h}}(a_{\mathsf{h}}) = -\varepsilon k_{\mathsf{v}} \omega \cdot \rho J_1(a_{\mathsf{t}}), \tag{22}$$

where $J_1(a_{\mathsf{t}})$ amplifies (with the sign convention of Fig. 1, $a_{\mathsf{h}} < 0$ when the robot is making forward progress) the fore-aft acceleration as shown in Fig. 3.

Comparing (19) to (10), and (22) to (11), we see that the averaged body dynamics are conjugate to the template dynamics. Moreover, it is clear that (9) and (18) are not conjugate vector fields, and neither are (11) and (21).

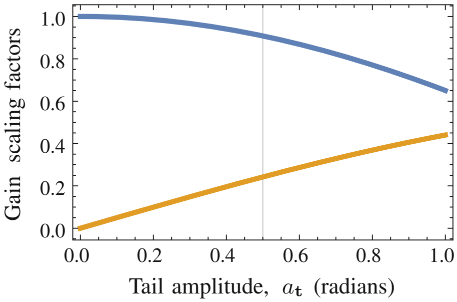


Fig. 3 Gain scaling terms $\widetilde{k}_{\mathsf{v}}/k_{\mathsf{v}}$ (*blue*, (19)) and $J_1(a_{\mathsf{t}})$ (*orange*, (22)), as a function of the tail sweep. As $a_{\mathsf{t}} \to 0$, we recover the unscaled vertical template dynamics, but with larger tail sweep (the thin vertical line corresponds to empirically observed tail sweeps from Sect. 5), the equilibrium hopping height decreases, and the forward speed increases

Remark Compared to classical anchoring [3], conjugacy of averaged fields is the analogue of the invariance of the embedded submanifold. However, note that there is no analogue of the "attraction" property yet (i.e. do asymmetric orbits attract to more symmetric ones?). We aim to study the "transient" behavior in future work.

4.2 Stability Derived from Templates

The strong relation between the averaged and unaveraged systems guaranteed by Proposition 1 allows us to conclude existence of a periodic orbit as well as its hyperbolic stability type from that of the averaged system, which we calculate below.

Reset map for Tailed SLIP "Stepping" (setting the leg touchdown angle $\theta_{\mathrm{td}+}$ as a function of the liftoff state, θ_{lo}, and a desired leg angular momentum $a^*{}_{\mathsf{h}}$) can be used to control forward speed for a pendular walking/running system [10]. In particular, a slight modification of the "scissor algorithm" [10, Chap. 5] yields

$$\theta_{\mathrm{td}} = -\theta_{\mathrm{lo}} + u_{\mathsf{h}}, \text{ where } u_{\mathsf{h}} := k_{\mathsf{h}}(a^*{}_{\mathsf{h}} - a_{\mathsf{h}}), \tag{23}$$

Assumption 3 also simplifies the flight map, since the CoM flight behavior is well represented by a ballistic motion. The guard set for the "flight" reset is identical to that for the embedded radial coordinate r (Sect. 3.1), $G(a_{\mathsf{v}}, a_{\mathsf{h}}, t) = G_{\mathsf{v}}(a_{\mathsf{v}}, t)$

We transform the liftoff velocity vector into Cartesian coordinates, reflect the vertical component (ballistic flight) and transform back to polar coordinates to get

$$\begin{bmatrix} \rho\dot{\theta} \\ \dot{r} \end{bmatrix}_{\mathrm{td}+} = \mathrm{Rot}(-\theta_{\mathrm{td}+}) \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} \mathrm{Rot}(\theta_{\mathrm{lo}}) \begin{bmatrix} \rho\dot{\theta} \\ \dot{r} \end{bmatrix}_{\mathrm{lo}},$$

and as a last step we can substitute in (23) with $a := \begin{bmatrix} a_{\mathsf{h}} \\ a_{\mathsf{v}} \end{bmatrix}$ (with the components as defined just before (19), (22)) to get

$$\begin{aligned} a_{\mathrm{td+}} &= \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} \begin{bmatrix} \rho & \\ & 1 \end{bmatrix} \mathrm{Rot}(-\theta_{\mathrm{td+}}) \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} \mathrm{Rot}(\theta_{\mathrm{lo}}) \begin{bmatrix} 1/\rho & \\ & 1 \end{bmatrix} a_{\mathrm{lo}} \\ \implies \mathcal{R}(a) &\overset{(23)}{=} \begin{bmatrix} \cos u_{\mathsf{h}} & -\rho \sin u_{\mathsf{h}} \\ \sin u_{\mathsf{h}}/\rho & \cos u_{\mathsf{h}} \end{bmatrix} a. \end{aligned} \tag{24}$$

As apparent from (24), this map does not depend upon the leg angle, θ (from (23), u_{h} only depends on $\dot{\theta}$). Together with the equivariance of the continous dynamics with θ (Assumption 2), this allows us to work in the a_{h}-quotient space (as in Sect. 3.2). A drawback of this reduction is that the hopping system exhibits pairs of antisymmetric steps [10, Chap. 5]: a period-2 evolution of non-neutral stances which are each mirror-symmetric to the subsequent step. In our empirical trials, with small leg sweep (Assumption 4) we did not observe any significantly skewed stances.

Existence of periodic orbits With fixed k_{v} (and by (17), a_{t}) the averaged fields (19), (21) yield simple flows of the form

$$\overline{f^{\pi}}(a) = \begin{bmatrix} a_{\mathsf{h}} - \nu_1, \\ \nu_2 a_{\mathsf{v}} + \nu_3 \end{bmatrix}, \tag{25}$$

where $a := (a_{\mathsf{h}}, a_{\mathsf{v}})$, ν_1, $0 < \nu_2 < 1$ and ν_3 are constants depending only on k_{v} and system parameters.

The zeros of $Q(a) = a - \mathcal{R} \circ \overline{f^{\pi}}(a)$ lie on periodic orbits of the averaged system. By the implicit function theorem, we only need to find an open neighborhood of a where $\mathrm{D}_a Q$ is full rank to check that periodic orbits exist. By design of u_{h}, $k_{\mathsf{h}} \ll 1$ (before (13)), and so from (24),

$$\mathrm{D}_a \mathcal{R} = \begin{bmatrix} \cos u_{\mathsf{h}} & -\rho \sin u_{\mathsf{h}} \\ \sin u_{\mathsf{h}}/\rho & \cos u_{\mathsf{h}} \end{bmatrix} + \mathcal{O}(k_{\mathsf{h}}) \approx \begin{bmatrix} \cos u_{\mathsf{h}} & -\rho \sin u_{\mathsf{h}} \\ \sin u_{\mathsf{h}}/\rho & \cos u_{\mathsf{h}} \end{bmatrix}, \tag{26}$$

and so, combining with $\mathrm{D}\overline{f}^{\pi}$ from (25), $\mathrm{D}_a Q \approx I - \begin{bmatrix} 1 & \\ & \nu_2 \end{bmatrix} \begin{bmatrix} \cos u_{\mathsf{h}} & -\rho \sin u_{\mathsf{h}} \\ \sin u_{\mathsf{h}}/\rho & \cos u_{\mathsf{h}} \end{bmatrix}$, which is full rank as long as the right summand has no unity eigenvalues (sufficient condition). Since $\nu_2 < 1$, $u_{\mathsf{h}} \neq 0$ is sufficient to ensure this.

Proposition 3 (Stability) *Tailed SLIP limit cycles have locally stable return maps.*

Proof We examine the stability properties of the discrete dynamical system $a \mapsto \overline{P}(a)$, the Jacobian of which factors into

$$\mathrm{D}_a \overline{P}|_a = \mathrm{D}_a(\overline{f^{\pi}} \circ \mathcal{R})|_a = \mathrm{D}_a \overline{f^{\pi}}|_{\mathcal{R}(a)} \cdot \mathrm{D}_a \mathcal{R}|_a. \tag{27}$$

Repeating the argument and calculation (26), for sufficiently small k_{h}, $\mathrm{D}_a\overline{P}(a) = \begin{bmatrix}1 & \\ & \nu_2\end{bmatrix}\begin{bmatrix}\cos u_{\mathsf{h}} & -\rho\sin u_{\mathsf{h}} \\ \sin u_{\mathsf{h}}/\rho & \cos u_{\mathsf{h}}\end{bmatrix}$. The eigenvalues of $\mathrm{D}_a\overline{P}$ are in the stable region iff its determinant and trace satisfy (i) det < 1, (ii) det $>$ tr $- 1$, and (iii) det $> -$tr $- 1$. A simple calculation yields

$$\det(\mathrm{D}_a\overline{P}) = \nu_2, \quad \mathrm{tr}\,(\mathrm{D}_a\overline{P}) = (1+\nu_2)\cos u_{\mathsf{h}}, \tag{28}$$

and since $0 < \nu_2 < 1$, $\mathrm{D}_a\overline{P}$ has stable eigenvalues. □

5 Numerical and Empirical Results

The theoretical development in this paper is motivated by the implementation of tail-energized hopping on the Penn Jerboa [12], a tailed monopedal robot (with coincident hip and tail axes) in the sagittal plane. To demonstrate different facets of the ideas presented here, we present both simulation results, as well as experiments on the physical hardware.

Relationship to model (Sect. 4) The model presented earlier in the paper is a reasonable representation of the physical platform, but with the distinctions

- the robot has an additional "body pitch" DOF, which we immobilize using a boom for our experiments, and
- the tail actuator on the robot is attached between the tail and the body, whereas in the model it is attached between the tail and leg. For small leg sweep (Assumption 4), the empirical behavior of either appears to be very similar.

Assumption 3 is reasonably satisfied by the physical platform: the tail mass is 0.15 kg, tail length is 0.3 m, while the body mass and inertia are (approx) 2.27 Kg and 0.025 Kg-m^2 respectively. That is, the tail mass is 6.6% of body mass, but tail inertia is 54% of body inertia. The overall control architecture of the robot is implemented as in Table 1.

Numerical simulation Figure 4 shows simulation traces at (or near) a limit cycle for different plant models. The various parameters used in the simulation (matched to the robot hardware as much as possible): $m = 2.27$ Kg, $\rho = 0.12$ m, $m_t = 0.15$ Kg, $\rho_t = 0.3$ m, $\omega = 36.35$ rad/s, and $\varepsilon k_{\mathsf{v}} = 0.118$. The template plants and the analytically intractable Jerboa simulations both simulations all attracted to a steady-state hopping behavior, and we picked the "limit cycle" stance trajectories for Fig. 4 by simply allowing the simulation to run for some time.

The effect of averaging is apparent in the traces: while the qualitative system behavior is identical for the various plant models, the red and blue curves suffer from periodic, within-stance, perturbations (mechanical coupling interactions that we reimagine as time-varying disturbances to be "integrated out.").

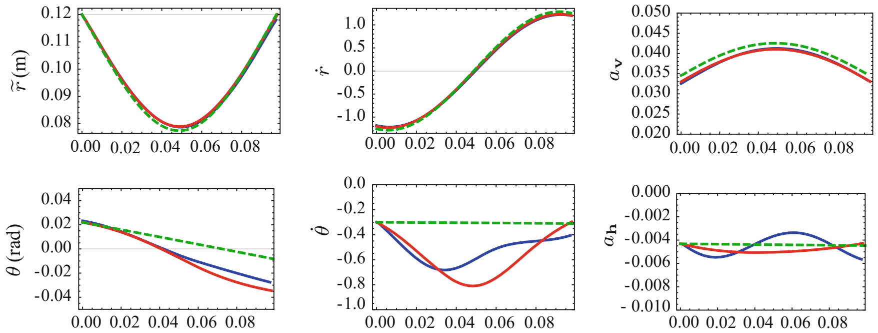


Fig. 4 Results from numerical simulation showing the evolution of various coordinates during a single stance phase: full Jerboa [12] simulation (no Assumption 3, analytically intractable) with pitch immobilized (*blue*), tailed SLIP as modeled in this paper (15) (*red*), and the isolated templates of Sect. 3 (*green*)

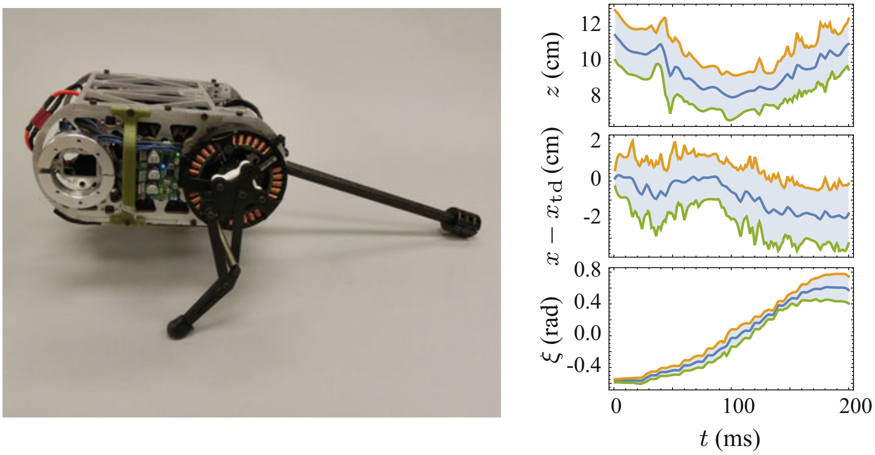


Fig. 5 *Left* The physical platform used in this paper, the Penn Jerboa [12]. *Right* stance data (mean and standard deviation) for a single trial on the robot, averaged over 36 steps. The z and x traces show rough profile comparable to the r, θ profiles in Fig. 4 modulo the polar transform (which leaves the profiles intact for small leg angles as we have assumed in Assumption 4), and the ξ traces shows the expected profile (17) with our sinusoidal driving input

Additionally, the gain scaling effect showed in Fig. 3 can be seen in the rightmost column of Fig. 4. For the same k_v and other parameters, the body (red, blue) has lower vertical energy and higher fore-aft energy than the template (green dashed) (Fig. 5).

Robot experiments For Jerboa [12] experiments, we constrain out-of-plane motion as well as body pitch using a boom. Qualitatively, the robot can hop stably (in multiple trials we recorded approximately 50 steps, and the experiment was terminated before failure), at limited speeds forward, backward, or in place. We have documented some

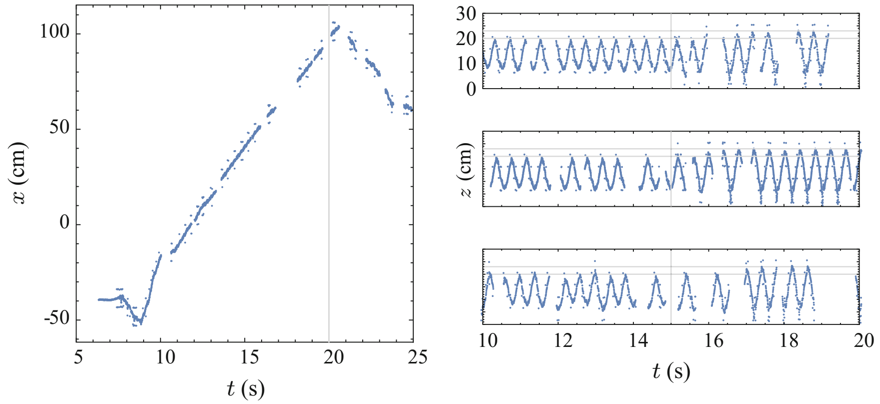


Fig. 6 We demonstrate stability and control authority over the hopping behavior by plotting (ground truth) fore-aft (*left*) and vertical (*right*) position for two trials where we imposed a step change in the desired energy levels, a_{h} (reversed at $t = 20$ s in the *left plot*) and a_{v} (increased by 33% at $t = 15$ s in the *right plots*). The "gaps" in the data are due to unfortunate communication dropouts with the instrumented boom

Table 2 Summary of 7 in-place hopping trials with height transition

Increase in k_{v}	Pre-transition		Post-transition	
	Mean ascent (cm)	Std. dev. (cm)	Mean ascent (cm)	Std. dev. (cm)
33%	8.87692	3.14861	10.2622	4.34376

trials in the supplementary video [22]. Figure 6 shows ground-truth time series data from the instrumented boom demonstrating that the tail-energized hopping strategy can attain stable hopping with tunable hopping height and fore-aft speed. Note that the high-frequency noise is caused by parasitic oscillations in the boom tube (which can be seen in the supplementary video [22]), and the blank portions are due to communication stalls in the data collection system.

In the left plot, the desired forward hopping speed is reversed at $t = 20$ s and it can be seen that the robot begins hopping backwards in a few strides.

In the in-place hopping trials (right), k_{v} is increased 33% at $t = 15$ s, and we can see that within around three strides, the hopping height increases to a new equilibrium value. Table 2 shows some statistics on these trials (the gathered data from 7 of the 12 trials was usable, with large communication drops in the remaining). Qualitatively, the robot stably switched between two hopping heights in each run.

6 Conclusion

To sum up the contributions of this paper, we (a) motivate, propose, and demonstrate a new, relaxed, notion of anchoring using our extension (Sect. 2) of classical averaging theory, (b) show that in a decoupled implementation (Table 1) of tail-energized hopping in an idealized (Assumptions 1–4) planar tailed monoped (15), the stance vector field decomposes in an averaged sense (but not in an exact sense) to a 1DOF vertical hopper and active rimless wheel (Proposition 2), and (c) give a proof of stability (Proposition 3) for the tailed monoped (d) test numerically and through experiments (Sect. 5) that the same feedback control rules stabilize planar hopping on the Jerboa [12] with controllable height and forward speed.

Future work We next plan to develop a formal definition of average anchoring (which we have skirted in this paper in favor of an illustration using the tailed monoped example), as well as an investigation into its relation to the classical anchoring notion [3].

The physical platform we have used, the Jerboa, has 12 unconstrained DOF's [12], but our tailed monoped model in this paper considers only 5 of them (2DOF body, 2DOF leg, 1DOF tail). In future work, we wish to explore the composition of more templates with the ones in Sect. 3 (as suggested in [11]), in order to be able to detach the robot from the boom.

Acknowledgements We would like to thank Sam Burden for helpful discussions on smoothing hybrid systems. This work was supported in part by the ARL/GDRS RCTA project, Coop. Agreement W911NF-1020016 and in part by NSF CABiR (CDI 1028237).

References

1. Guckenheimer, J., Holmes, P.: Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields. Applied Mathematical Sciences. Springer, New York (1990)
2. Holmes, P., Full, R.J., Koditschek, D.E., Guckenheimer, J.: The dynamics of legged locomotion: models, analyses, and challenges. SIAM Rev. **48**(2), 207–304 (2006)
3. Full, R.J., Koditschek, D.E.: Templates and anchors: neuromechanical hypotheses of legged locomotion on land. J. Exp. Biol. **202**, 3325–3332 (1999)
4. Buhler, M., Koditschek, D.E., Kindlmann, P.J.: A family of robot control strategies for intermittent dynamical environments. IEEE Control Syst. Mag. **10**(2), 16–22 (1990)
5. Nakanishi, J., Fukuda, T., Koditschek, D.E.: A brachiating robot controller. IEEE Trans. Robot. Autom. **16**(2), 109–123 (2000)
6. Saranli, U., Schwind, W.J., Koditschek, D.E.: Toward the control of a multi-jointed, monoped runner. In: 1998 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2676–2682 (1998)
7. Saranli, U., Koditschek, D.E.: Template based control of hexapedal running. In: Proceedings of 2003 IEEE International Conference on Robotics and Automation ICRA'03, vol. 1, pp. 1374–1379. IEEE (2003)
8. Grizzle, J.W., Chevallereau, C., Sinnet, R.W., Ames, A.D.: Models, feedback control, and open problems of 3d bipedal robotic walking. Automatica **50**, 1955–1988 (2014)
9. Westervelt, E., Grizzle, J.: Feedback Control of Dynamic Bipedal Robot Locomotion. Control and Automation Series. CRC PressINC, Boca Raton (2007)

10. Raibert, M.: Legged Robots that Balance. Artificial Intelligence. MIT Press, Cambridge (1986)
11. De, A., Koditschek, D.E.: Parallel composition of templates for tail-energized planar hopping. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 4562–4569 (2015)
12. De, A., Koditschek, D.E.: The Penn Jerboa: A platform for exploring parallel composition of templates. Technical report, University of Pennsylvania (2015). arXiv:1502.05347
13. Koditschek, D.E., Buehler, M.: Analysis of a simplified hopping robot. Int. J. Robot. Res. **10**, 587–605 (1991)
14. Bhounsule, J., Cortell, J., Ruina, A.: Design and control of ranger: an energy-efficient, dynamic walking robot. In: Proceedings of CLAWAR, pp. 441–448 (2012)
15. Khalil, H.: Nonlinear Systems, 3rd edn. Prentice Hall, Upper Saddle River (2002)
16. McGeer, T.: Passive dynamic walking. Int. J. Robot. Res. **9**(2), 62–82 (1990)
17. Secer, G., Saranli, U.: Control of monopedal running through tunable damping. In: 2013 21st Signal Processing and Communications Applications Conference (SIU), pp. 1–4 (2013)
18. Blickhan, R., Full, R.: Similarity in multilegged locomotion: bouncing like a monopode. J. Comp. Physiol. A **173**(5), 509–517 (1993)
19. Burden, S., Revzen, S., Sastry, S.S.: Dimension reduction near periodic orbits of hybrid systems. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pp. 6116–6121. IEEE (2011)
20. Geyer, H., Seyfarth, A., Blickhan, R.: Spring-mass running: simple approximate solution and application to gait stability. J. Theor. Biol. **232**(3), 315–328 (2005)
21. Weisstein, E.W.: Bessel Function of the First Kind. From MathWorld-A Wolfram Web Resource (2015). http://mathworld.wolfram.com/BesselFunctionoftheFirstKind.html
22. De, A., Supplementary material for this paper (2015). http://kodlab.seas.upenn.edu/Avik/AveragingTSLIP

A Reachability-Based Planner for Sequences of Acyclic Contacts in Cluttered Environments

S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon and J. Pettré

1 Introduction

We consider the problem of planning the acyclic sequence of contacts describing the motion of a multiped robot in a cluttered environment. Acyclic contact planning is a particular class of motion planning where every configuration of the resulting trajectory must be in contact with the environment in order to support the balance of the system. The difficulty of the problem comes both in practice from the proximity to the obstacles (that tends to make the sampling of valid configuration tedious) and in theory from the foliation of the configuration space, where zero-measure manifolds intersect in a combinatorial manner [20]. Acyclic motion planning is a problem of interest in robotics, neurosciences and biomechanics. It is also interesting for virtual character animation. Early contributions in this field rely on local adaptation of motion graphs [13] or ad-hoc construction of locomotion controllers [18]. These

S. Tonneau (✉) · N. Mansard
LAAS-CNRS, Toulouse, France
e-mail: Steve.Tonneau@laas.fr

N. Mansard
e-mail: Nicolas.Mansard@laas.fr

C. Park · D. Manocha
UNC, Chapel Hill, NC, USA
e-mail: chpark@cs.unc.edu

D. Manocha
e-mail: dm@cs.unc.edu

F. Multon
Unversité Rennes II, Rennes, France
e-mail: fmulton@irisa.fr

J. Pettré
Inria, Le Chesnay Cedex, France
e-mail: julien.pettre@inria.fr

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_17

approaches can intrinsically not adapt to new situations or discover complex behaviors in unforeseen contexts. In robotics, the attention of the community first focused on the generation of cyclic locomotion patterns, in particular for bipedal walking on flat terrains [12]. While planning cyclic bipedal contacts is now mature, with existing real-time solutions [3], the problem remains open for more generic acyclic contacts.

1.1 State of the Art

The issue of planning acyclic contacts was first completely described by Bretl et al. [6], where it is proven to require the handling of two simultaneous problems: $\mathscr{P}_1$: a relevant guide trajectory for the root of the robot in $SE(3)$; and $\mathscr{P}_2$: the planning of a discrete sequence of acyclic, balanced contact configurations along the trajectory.[1] A key issue is to avoid combinatorial explosion when considering at the same time the possible contact configurations and the potential trajectories. This seminal paper proposes a first effective algorithm, able to handle simple situations (such as climbing scenarios), but not scalable to arbitrary environments. Following it, several papers have applied this approach in particular situations, typically limiting the combinatorial by imposing a fixed set of possible contacts ([11, 21]).

Most of the following papers have explored alternative formulations to handle the combinatorial issue. Two main directions have been explored. **On one hand, local optimization of both the root trajectory $\mathscr{P}_1$ and the contact positions $\mathscr{P}_2$** has been used, to trade the combinatorial of the complete problem for a differential complexity, at the cost of local convergence [14, 15]. To keep reasonable computation times, the method uses a simplified dynamic model for the avatar. Still, the computation time is far from interactive (about 1 min of computation for a sequence of 20 contacts). Deits et al. [8] address the problem as a mixed integer one, but only cyclic, bipedal locomotion is considered. Aside from the computation cost, a major drawback of these optimization based approaches is thus that they only offer local convergence when applied to acyclic contact planning.

On the other hand, the two problems $\mathscr{P}_1$ and $\mathscr{P}_2$ might be decoupled to reduce the complexity. The interest of the decoupling has first been shown by manually setting up a rough guide trajectory (i.e. an ad-hoc solution to $\mathscr{P}_1$) [10]. $\mathscr{P}_2$ is then addressed as the combinatorial computation of a feasible contact sequence in the neighborhood of the guide. A solution can then be found, at the cost of prohibitive computation times (several hours). Furthermore, this approach is suboptimal because the quality of the motion is conditioned by the relevance of the guide trajectory, which is not evaluated a priori. Bouyarmane et al. [4] focus on automatically computing a guide trajectory with guarantees of contact feasibility, by extending key frames of the trajectory into whole-body, balanced contact configurations. Randomly sampled

[1] A third non trivial problem, $\mathscr{P}_3$, not adressed in this work, then consists in interpolating a complete motion between two postures of the contact sequence. We address $\mathscr{P}_3$ successfully on the HRP-2 robot in Carpentier et al. [7]. Another framework for 3D animation is proposed in Park et al. [17].

configurations are projected into the contact submanifold using a generalized inverse kinematics solver, a computationally expensive process (about 15 min are required to compute a guide trajectory in the examples presented). Moreover this explicit projection is yet an insufficient condition and does not provide strong guarantees on the feasibility of the path between two key positions in the trajectory.

1.2 Paper Contribution and Organization

We choose to focus on the sample-based methodology, more able to find complex trajectories in cluttered environments. While the theoretical structure of the problem is well understood, there is currently no scalable method to solve it. The combinatorial of the original problem is too high to have any hope of tackling it directly. Alternative formulations are necessary to obtain practical solutions. We believe that the separation between the guide trajectory and the contact sequence is the most promising direction [10]. However, this direction raises two theoretical questions that remain to be solved, or even to be properly formulated:

- The guide trajectory must satisfy a property guaranteeing the existence of a contact sequence to actuate it.[2] This property has not been studied yet for acyclic planning: the only way to validate a trajectory is to explicitly compute the contacts, which is computationally not reasonable [4].
- There is an infinite combination of possible contact sequences for a given root trajectory. The selection of one particular contact sequence with interesting properties (minimum number of contact change, robustness, efficiency or naturalness) has been studied for cyclic cases [11], but has not been efficiently applied to cluttered environments (Previous contributions mostly randomly pick one contact sequence, leading to possibly very tedious contact sequences [5]).

We claim that the desirable contact properties of a guide trajectory, proposed in [4], can be formulated in a space of lower dimension, which we call C_{reach}. This formulation can make the planning of a guide trajectory more efficient computationally, while providing equivalent guarantees to planning directly in the configuration space. Among the particular properties obtained when planning in C_{reach}, we would like to guarantee that any reduced trajectory can actually lead to a feasible sequence of contacts, in which case we say that the reduced trajectory is truly feasible. It is possible in theory to guarantee that any reduced trajectory is truly feasible, even if it is more efficient in practice to approximate this property. The true-feasibility of the guide trajectory then allows us to focus on the selection of one particular sequence of contacts, for example one that minimizes the number of contacts in the sequence or maximizes the robot efficiency or style.

[2] This property is related to the controllability of the root actuated by the contact forces, but for discrete bounded properties.

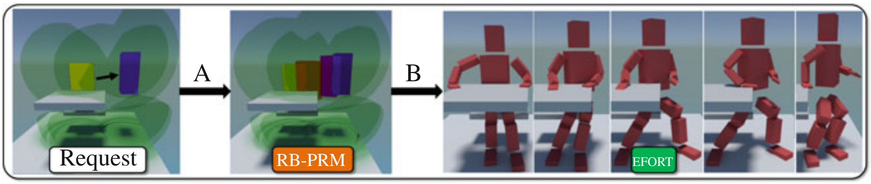


Fig. 1 Overview of our 2-stage framework. **a** Given a path request between the yellow and blue positions, a guide trajectory is computed in C_{reach} using RB-PRM. **b** The trajectory is extended into a discrete sequence of contact configurations using EFORT

Based on these fundamental observations, we implement a very efficient acyclic contact planner. Our method is based on a probabilistic roadmap (PRM), that computes offline guide trajectories that are approximately truly feasible. The planner then resolves online the contact sequence by refining a guide trajectory computed from the PRM. Our planner is able to compute physically-consistent contact sequences for very complex systems (a humanoid, 28 joints; and an insectoid, 48 joints) in a few seconds for classical scenarios like climbing, and less than a minute for very complex problems like egress from a damaged truck. The planner also generalizes to planning dexterous manipulation movements, as demonstrated by preliminary results.

The contributions of the paper are twofold. We propose the first theoretical characterization of today's most efficient practical approach to sampled-based planning of acyclic contacts. And based on this characterization, we propose a very efficient and general implementation of an acyclic contact planner, the first one compatible with interactive applications.

We propose a framework to address the motion planning problem for multiped robots in cluttered environments: given a start and a goal configuration, the objective is to compute a sequence of contact configurations allowing to achieve the motion. For instance, we can consider the task of standing up, illustrated in Fig. 1–right. The problem is decoupled into two sequential phases: (1) the computation of a guide trajectory for the root of the robot; (2) the computation of a discrete sequence of contact configurations allowing to achieve the motion along the trajectory. The remainder of this section presents the general organization of our method in Sect. 2. The two following Sects. 2 and 3 present respectively our answer to problems $\mathscr{P}_1$ and $\mathscr{P}_2$. Finally, we propose a complete experimental validation of the planner with three very different kinematic chains (humanoid, insectoid and three-finger manipulator) in various scenarios.

1.3 Computation of a Guide Trajectory

We first consider the problem of planning a relevant guide trajectory. The objective is to compute a trajectory of root placements which will allow contact creation. The objective of this first part is to preserve the completeness of the planner: it should

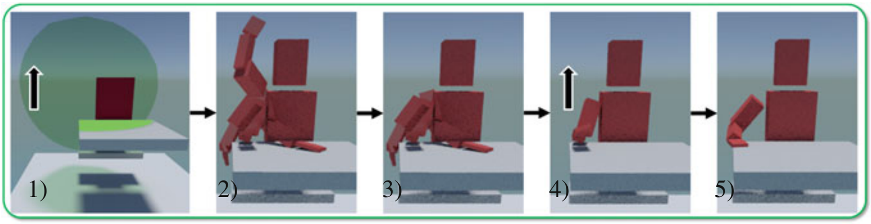


Fig. 2 Generation of a contact configuration for the right arm of a humanoid robot. *1* Selection of reachable obstacles. *2* A request is performed on a database of configurations. *3* Configurations too far from contact are eliminated. *4* The best candidate according to EFORT is chosen. *5* The final contact is achieved using inverse kinematics

be able to explore any possible guide trajectory, but at the same time, any computed guide trajectory must be truly feasible, i.e. must lead to a valid sequence of contacts.

An intuitive description of such placements is "close, but not too close": close, because a contact surface must be partially included in the range of motion of the robot (represented for the right arm in Fig. 2–1); not too close, because the robot must avoid collision (which is represented by the hull including the torso in Fig. 2–1). We define formally C_{reach}, the set of interesting root placements, in which we compute a guide trajectory with a sampling based planner, the reachability PRM–RB-PRM– (Fig. 1–a). Planning in C_{reach} boils down to planning in SE(3), which has an acceptable practical complexity. Details are presented in Sect. 2.

1.4 Generating a Discrete Sequence of Contact Configurations

The second stage is to extend the guide trajectory into a sequence of contact configurations (Fig. 1–b). Thanks to the nice property that we manage to obtain for the guide, obtaining a random sequence of contact is an easy problem. The goal of this part is to select an efficient sequence, in particular by reducing the number of contacts in the sequence. To create contacts in an efficient manner, we consider each limb as a manipulator attached to the root, and select the most relevant contact from a database of precomputed configurations (Fig. 2). The relevance is defined as the contribution to the quasi static balance of the robot, and as the contribution to the motion of the root. This task efficiency is measured based on the Extended Force Transmission ratio (EFORT) [22]. Details are presented in Sect. 3.

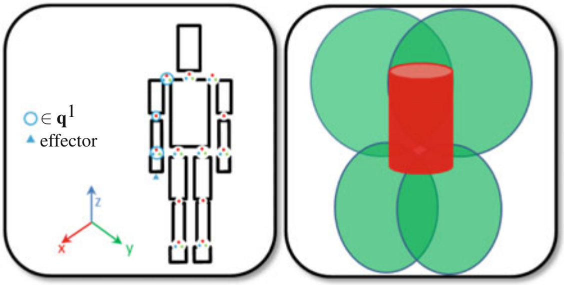


Fig. 3 *Left* Robot in a rest configuration. The right arm is denoted as the limb R^1. Each colored dot represents a degree of freedom around an axis. *Right* Volumes of the robot. The *red* geometry denotes W^0 and must remain collision-free. The *green spheres* are the W^k

1.5 Notation Conventions and Definitions

A vector $\mathbf{x}$ is denoted with a bold lower case letter. A matrix $\mathbf{A}$ is denoted with a bold upper case letter. A set C is denoted with a upper case italic letter. Scalar variables and functions are denoted with lower case italic letters, such as r or $f(\mathbf{x})$.

A robot is a kinematic chain R, comprising $n + 6$ degrees of freedom (DOFs). R is composed of l limbs R^k, $1 \leq k \leq l$, attached to a root. It is described by a configuration $\mathbf{q} \in SE(3) \times \mathbb{R}^n$. We define some relevant projections of $\mathbf{q}$:

- $\mathbf{q}^k$ denotes the configuration (a vector of joint values) of the limb R^k (Fig. 3);
- $\mathbf{q}^{\overline{k}}$ denotes the vector of joint values of R **not** related to R^k. We define for convenience $\mathbf{q} = \mathbf{q}^k \oplus \mathbf{q}^{\overline{k}}$;
- $\mathbf{q}^0 \in SE(3)$ denotes the position and orientation of the root of the robot R.

The environment O is defined as the union of the obstacles O_i it contains. The volume encompassing the trunk of the robot is denoted W^0 (Fig. 3-right: central cylinder). The range of motion of a limb R^k is denoted W^k (Fig. 3-right: the four ellipses).

$$W^k = \left\{\mathbf{x} \in \mathbb{R}^3 : \exists \mathbf{q}^k, \mathbf{p}^k(\mathbf{q}^k) = \mathbf{x}\right\} \tag{1}$$

where $\mathbf{p}^k$ denotes the end-effector position of R^k (translation only) for $\mathbf{q}^0$ the null displacement. We also define $W = \bigcup_{k=1}^{l} W^k$. Finally, we define $W^k(\mathbf{q}^0)$, $0 < k \leq l$ as the volume W^k translated and rotated by the rigid displacement $\mathbf{q}^0$.

2 Computation of a Guide Trajectory in C_{reach} (Stage 1)

We consider the problem of computing a guide trajectory $\mathbf{q}^0(t) : [0, 1] \longrightarrow SE(3)$ for the geometrical root of a multiped robot, connecting start and goal configurations. As said in previous section, the goal is to enforce that any configuration $\mathbf{q}^0$ of the guide is truly feasible, i.e. can be mapped to a balanced configuration in contact. We denote by $C_{contact} \subset SE(3) \times \mathbb{R}^n$ the contact submanifold of the robot.

We say that a root placement $\mathbf{q}^0$ is *truly feasible* if there exists a bijective mapping[3] π such that

$$\pi : \quad \mathbf{q}^0 \in SE(3) \longrightarrow \mathbf{q}^0 \oplus \mathbf{q}^{\bar{0}} \in C_{contact} \tag{2}$$

The set of all truly feasible root placements is denoted by C_{reach}. By extension, a trajectory $\mathbf{q}^0(t)$ is truly feasible if $\forall t \in [0, 1]$, $\mathbf{q}^0(t) \in C_{reach}$.

For a two-stage acyclic contact planner to be exact and complete, we need the combination of two conditions on a guide trajectory generator: all the generated trajectories must be *truly feasible* (sufficient condition); the guide planner must be complete, i.e. it must be able to discover any truly feasible trajectory (necessary condition).[4]

2.1 Conditions for True Feasibility

By default, the true feasibility implies a constructive demonstration by exhibiting a valid π. This is the approach chosen by [4]. However, computing a valid $\mathbf{q}^{\bar{0}}$ is costly in practice. In this section we rather define a necessary condition and a sufficient condition for true feasibility that do not require this explicit computation.

True Feasibility: Necessary Condition

For a contact to be possible, a volume $O_i \in O$ necessarily intersects with the range of motion $W(\mathbf{q}^0)$ (Fig. 2–1). Furthermore, if $\mathbf{q}^0$ is truly feasible, then the trunk of the robot $W^0(\mathbf{q}^0)$ is necessarily not colliding with the environment O.

Therefore we can approximate C_{reach} with a set $C_{reach} \subset C^1_{reach}$ with the reachability condition defined as:

$$C^1_{reach} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \wedge W^0(\mathbf{q}^0) \cap O = \emptyset\} \tag{3}$$

It is straightforward to prove that $C_{reach} \subset C^1_{reach}$ (by construction of the included set). This inclusion is very important: it directly implies that any motion-planning algorithm with a guaranty of completeness in $SE(3) \times \mathbb{R}^n$ is complete in $C_{reach} \times \mathbb{R}^n$. This is a strong reduction of the search space, which can be directly applied to any existing method.

The condition C^1_{reach} is only necessary which means that one such root placement might not be truly feasible: in practice it is not guaranteed to find a valid sequence of contacts for every guide trajectory in C^1_{reach}.

Sufficient Condition for True Feasibility

A trivial sufficient condition for true feasibility can be constructed as a variation of C^1_{reach}, by replacing W^0 with a bounding volume B^{max} encompassing the whole

[3] This mapping is not uniquely defined.

[4] The proof is immediate, using a contradiction approach.

robot in a given pose, except for the effector surfaces to be in contact.[5] We denote by $C^{\infty}_{reach} \subset C_{reach}$ the set of root placements corresponding to the sufficient condition.

In general, the inclusion is strict, which means that we lose the completeness of the two-stage contact planner (i.e. the planner is not able to discover a trajectory inside $C_{reach} \setminus C^{\infty}_{reach}$). However, the sufficient condition guarantees that any such trajectories leads to a valid sequence of contacts (i.e. π is defined).

2.2 Reachability: A Compromise Condition

The sufficient condition is not interesting in practice since it leads the solver to lose too many interesting trajectories. The necessary condition is not perfect either, since the first stage of the planner would stop on a guide that is not truly feasible in practice. It might be possible to find a shape B that is necessary and sufficient; however, it seems intuitively very unlikely in general. The construction of a shape $W^0 \subset B \subset B^{max}$ leading to a necessary and sufficient condition (or the proof of its inexistence) is out of the scope of this work.

However between W^0 and B^{max}, a trade-off can be found between a necessary and a sufficient condition. We define W^0_s as the volume W^0 subject to a scaling transformation by a factor $s \in \mathbb{R}^+$. We then consider the spaces C^s_{reach}

$$C^s_{reach} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \wedge W^0_s(\mathbf{q}^0) \cap O = \emptyset\} \tag{4}$$

The higher s is, the closer the reachability condition is to being sufficient, and if $s = 1$, the planner is complete. The parametrization of s allows to find a trade-off between these two desirable properties. Section 4 shows that in practice, it is easy to adjust s to keep most of the interesting guides without introducing incorrect guides.

2.3 Computing the Guide Trajectory in C^s_{reach} with RB-PRM.

Once a value of s has been fixed, any sampling-based motion planner can be used to plan a trajectory in C^s_{reach}. The only variation consists in replacing the classical collision checking method with a test of appartenance to C^s_{reach} when verifying that configurations and associated local paths are valid. For this reason, there is no need to provide a pseudo-code for RB-PRM, although we provide here relevant information about our own implementation.

We have chosen to implement RB-PRM as a variation of Visibility-PRM [16], which usually leads to a smaller set of nodes than classical PRM planners. The

[5] This condition is trivial in the sense that the resulting W has a zero measure. For the need of the proof, the trivial sufficient condition is enough. In practice, the construction of a non-trivial including shape W^0 was possible for all the robot structures we considered.

associated drawback is that the paths returned by the planner might not be the shortest ones, which is typically not an issue in highly cluttered environments.

To sample more efficiently configurations of C^{s}_{reach}, we bias the sampling process to generate near obstacles configurations, similarly to [1] to generate configurations in narrow passages. First, a configuration is set to a random point on the surface of one obstacle. The configurations are then translated and rotated randomly until the reachability condition is satisfied. Again, our implementation only differs in the fact that C^{s}_{reach} is sampled instead of C_{free}.

3 From a Guide Trajectory to a Discrete Sequence of Contact Configurations (Stage 2)

As an input of this stage, we consider a truly feasible guide trajectory $\mathbf{q}^0(t)$: $[0, 1] \longrightarrow SE(3)$ for the root of the robot R. We now consider the second problem of computing a trajectory $\overline{\mathbf{q}}^0(t)$ for the limbs of the robot. Since we assume true feasibility, we know that such a trajectory exists. Contrary to previous works [4, 10], the goal here is not to find any such trajectory but rather to select one with particular properties. Specifically, we show here how to build a contact sequence with a small number of contact variations and good-efficiency and naturalness of the postures.

More precisely, any mapping π introduced in (2) can be used to expand $\mathbf{q}^0(t)$ into a whole-body trajectory. We propose here a particular construction of π leading to interesting contact sequences.

3.1 Extension of the Guide Trajectory

The guide trajectory $\mathbf{q}^0(t)$ is first discretized into a sequence of j key placements:

$$\mathbf{Q}^0 = [\mathbf{q}^0_0; \mathbf{q}^0_i; \dots, \mathbf{q}^0_{j-1}]$$

where $\mathbf{q}^0_0$ and $\mathbf{q}^0_{j-1}$ respectively correspond to the start and goal configurations. To ensure continuity in the contact transition phases, we rewrite π under the following recursive form for any $0 < i < j$:

$$\pi : \begin{cases} \mathbf{Q}^0 \in C_{reach} \longrightarrow C_{contact} \\ \qquad\quad \mathbf{q}^0_i \longrightarrow \mathbf{q}^0_i \oplus g(\mathbf{q}_{i-1}, \mathbf{q}^0_i) \end{cases}$$

We initialize the recurrence with $\pi(\mathbf{q}^0_0) = \mathbf{q}_0$ the initial configuration of the robot. The function g is defined independently by g^k for each limb R^k. In defining g^k, two aspects must be considered. Is the limb R^k in contact? And which criteria is it optimizing?

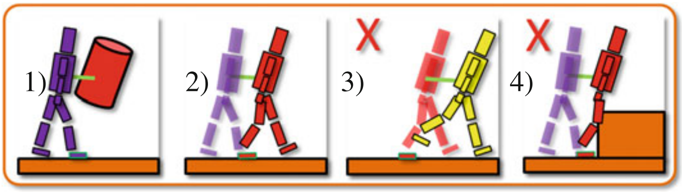


Fig. 4 Contacts are maintained unless their position is too far, or the environment prevents it

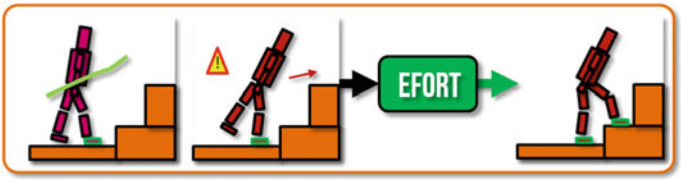


Fig. 5 Contacts are generated when the configuration is not balanced

Maintaining a contact: If possible, a limb in contact at time $i - 1$ remains in contact at i. The contact is broken if an inverse kinematics solver fails to find a collision free limb configuration which satisfies joint limits [2].

If the solver fails, the contact is broken and a collision free configuration is assigned to the limb.

Once a first candidate configuration is taken for all limbs, the quasi-static balance is tested by whether the weight wrench is in the gravito-inertial cone (i.e. there exists valid contact forces that compensate for the weight of the robot), using the geometric approach described in [19]. If the balance is not obtained, new contacts are randomly generated using the following procedure (Fig. 4).

Creating a contact: We consider a configuration where some limbs are in contact, some are free and quasi-static balance is not enforced. To enforce balance, we proceed in the following manner: we randomly select a contact free limb; if there is no contact free limb, we select the limb that made contact first. Using the contact generator introduced in [22], we project the configuration of this limb into a contact that enhances balance, if it exists (Fig. 5); If balance is not achievable and a contact is possible, it is generated anyway; If balance is not achieved, the next limb is selected and projected into a contact configuration, and so on. This approach can lead to the repositioning of existing contacts, in which case intermediate states are inserted to reposition the contacts. This current implementation does not guarantee that the planner will succeed in generating a balanced configuration, because true feasibility is not fully guaranteed. However in practice the planner is successful in the large majority of cases, as discussed in Sect. 4.2.

3.2 Contribution to the Global Movement: The EFORT Criteria

EFORT criterion: If only relying on the random sampling to select new contacts, the planner produces inefficient postures. The resulting contact sequence is then poorly efficient and unnatural. Moreover, the limbs are not well configured and are not able to efficiently follow the general movement: contacts break frequently.

When creating additional contacts, we therefore propose to select particular configurations that allow to exert a force compatible with the direction of motion. This task efficiency is measured with the Extended FORce Transmission ratio (EFORT) [22]. The measure of EFORT is given by

$$\alpha_{EFORT}(\mathbf{q}^k, \mathbf{m}) = [\mathbf{m}^T(\mathbf{J}\mathbf{J}^T)\mathbf{m}]^{-\frac{1}{2}}(\nu_0 \mathbf{n}^T \mathbf{m}) \tag{5}$$

where $\mathbf{J}$ is the Jacobian matrix of the limb R^k in configuration $\mathbf{q}^k$; ν_0 is the friction coefficient of the contact surface; $\mathbf{n}$ is the normal of the contact surface; and $\mathbf{m}$ is the direction opposite to the motion, given by the 3D vector connecting $\mathbf{q}_i^0$ and $\mathbf{q}_{i+1}^0$. The first part of the equation measures the ratio between the joint torques and the resulting force applied along $\mathbf{m}$. The second part quantifies the odds of slipping while applying a force along $\mathbf{m}$.

Optimization at creation: In practice, a database of configurations is stored for each limb, which can be considered as manipulator arms. The database is implemented as an octree data structure, indexed by the end-effector positions of the configurations (and additionally storing $\mathbf{J}$). Upon request, the octree returns a set of configurations close to contact (Fig. 2-3). These candidates are sorted based on their task efficiency, given by α_{EFORT}. The first candidate in this list satisfying the balance criterion and is collision free is selected and projected on the contact surface using our inverse kinematics solver.

4 Results

The main strength of our planner is that it efficiently works for arbitrary robot shapes. We first validate this aspect by producing a large variety of movements with three very different robots (humanoid, insectoid, dexterous hand) in five challenging scenarios. Two evaluations of the method are provided: qualitatively, by displaying the naturalness of the contact sequence in the companion video; and quantitatively by statistically measuring the validity of the compromise condition (Sect. 4.2) and the performances of the algorithm (Sect. 4.3).

4.1 Robot Models and Scenarios

Figure 6 describes the robot used in the experiments. The humanoid robot has four limbs, each with 7 DOFs. It has a total of 34 DOFs. The insectoid robot has six limbs, each with 8 DOFs, and a total of 54 DOFs. The hand has three fingers, each with 6 DOFs and a total of 24 DOFs.

In all the scenarios considered, the formulation of the problem is always the same: a start and goal configuration are provided as an input of the scenario, and the framework outputs a sequence of statically balanced contact configurations connecting the start and goal configurations. A companion video available at http://youtu.be/LmLAHgGQJGA displays the complete contact sequence obtained in all these scenarios. The video only renders the contact configurations (not the interpolation between contacts, which is out of the scope of the paper).

Truck egress (humanoid and insectoid): The robot must leave a truck the doors of which are blocked: it has to crawl through the front window. Figure 7 presents the sequence of contacts obtained for both robots: RB-PRM can find solutions in highly cluttered environments with narrow passages.

Climbing (humanoid and insectoid): The robot has to climb on a wall with several grasps disposed along it. In this scenario, we give stronger conditions for the sampled root placements: we require that more than one range of motion W^k collide with

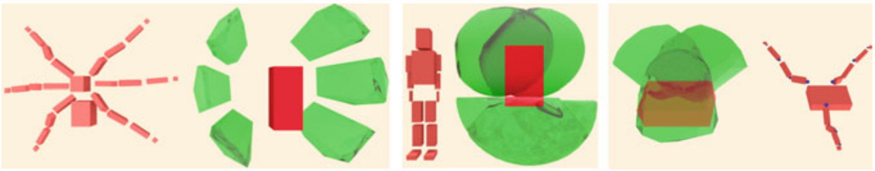


Fig. 6 Robots and associated volumes: in *red* W_s^0; in *green* the range of motion of each limb

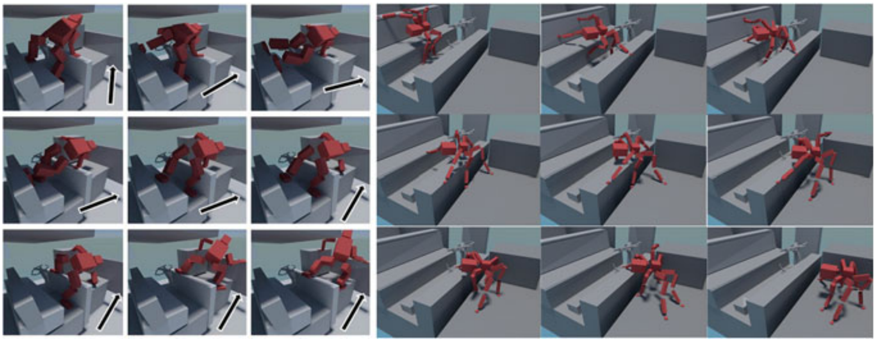


Fig. 7 The computed contact sequences for the truck egress scenario. Only selected postures are shown for the insect

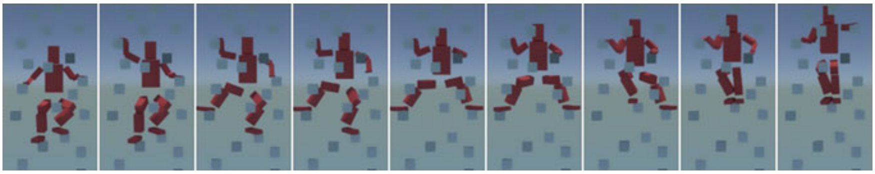


Fig. 8 The computed sequence for the climbing scenario

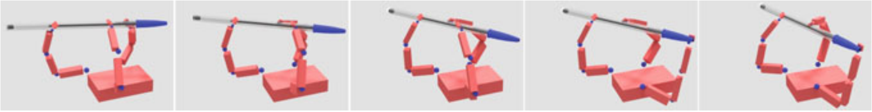


Fig. 9 Contact sequence found for a pen manipulation in a zero gravity environment

obstacles of the environment. Figure 8 presents the contact sequence obtained for the humanoid robot.

Manipulation of a pen (3-finger hand): This scenario is proposed to illustrate the genericity of our approach: we consider a manipulation task for a robotic hand and use our contact planner to compute a guide trajectory for the fingers, considered as effectors (Fig. 9). Although we do not address the hard issue of accounting for rolling motions, the planner is able to compute the shown sequences, this in less than 5 seconds.

Other scenarios (humanoids): The standing-up scenario (already presented in Fig. 1) is a setup taken from [10]: it corresponds to a long narrow passage in the configuration space. In the crouching scenario, demonstrated in the companion video, the character automatically goes from a standing to a crouching position to crawl under an obstacle.

4.2 Parametrization of the Reachability Condition

To find the appropriate C^s_{reach} in which to sample the guide trajectory, we computed the rejection rate for various values of s for each robot in the most cluttered truck scenario. For a given value of s, 10^6 root positions and orientations are computed in C^s_{reach}. In each case we try to generate a collision free contact configuration, with a database comprising $N = 10^5$ sample configurations for each limb. The rejection rate is the ratio between the number of failures and the number of trials. From Fig. 10 s is empirically chosen as the smallest value for which the rejection rate is minimal. For the humanoid, we thus chose $s = 2.2$, and for the insect, $s = 2.8$.

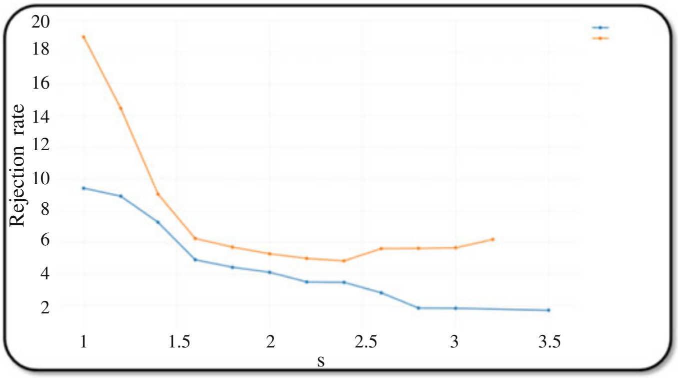


Fig. 10 Truck scenario rejection rates (%) for the humanoid (*orange*) and insectoid (*blue*), given *s*

4.3 Performance

The number of samples used for generating the contacts of each limb is 10000. Table 1 presents the average time (seconds) spent in the phases of the planner, for each phase and each scenario, and the number of contact phases of the sequence.

We observe that many contacts are required for the insect, which can be explained by its restricted range of motion. The time spent generating the navigation graph is about one minute. The time spent in generating the graph of the climbing scenario, despite the relatively open environment, is explained by the additional restrictions imposed on the reachability condition. The difficulty to find a balanced configuration essentially influences the time spent generating the contacts.

The number of contacts in the sequence gives a rough estimation of its duration in seconds. Except for the robots crawling out of the truck, all the contact generation are real-time. Additionally to the quality of the generated trajectories shown in the video, these computation times are a major practical achievement.

Table 1 **Average time** (in seconds) spent in RB-PRM generation, and the online generation of the contact sequence

	Generate RB-PRM (offline)	Generating the contact sequence	Number of contact states
Truck egress (humanoid)	73	15	10
Truck egress (insectoid)	70	23	48
Climbing (humanoid)	25	5	15
Climbing (insectoid)	21	27	51
Crouching (humanoid)	5	6	22

5 Discussion and Future Work

In this paper we consider acyclic contact planning in cluttered environments, formulated as two sub problems addressed sequentially: $\mathscr{P}_1$: the computation of a guide trajectory for the root of the robot that can be extended ; $\mathscr{P}_2$: the computation of a discrete sequence of contacts along this trajectory. Our contribution to $\mathscr{P}_1$ is a characterization of the properties that the guide trajectory must satisfy, in particular to enforce the completeness of the acyclic contact planner. We introduced a low dimensional space C_{reach} that can be mapped into the contact submanifold of the robot, approximated and efficiently sampled by our Reachability-Based planner. Our contribution to $\mathscr{P}_2$ is a pragmatic contact generation scheme that can take into account criteria to enforce interesting properties on the generated contacts. One such criterion, EFORT, is used to demonstrate the method and optimizes a force exertion compatible with the direction of motion.

Aside from the theoretical contributions, our results demonstrate that our method allows a compromise between three criteria that are hard to conciliate: generality, performance, and quality of the solution, making it the first acyclic contact planner compatible with interactive applications. **Regarding generality**, the reachability condition, coupled with an approach based on limb decomposition, allows the method to address arbitrary multiped robots. The only pre-requisite is the specification of the volumes W^0 which can is adjusted from a statistical analysis such as the one run in Sect. 4.2. **Regarding performance**, our framework outperforms existing methods in addressing either $\mathscr{P}_1$ or $\mathscr{P}_2$, leading to computation costs close to real-time in statically known environments. **Regarding the quality of the trajectories**, a parametrization of the reachability condition allows us to compute relevant trajectories in all the scenarios presented, with low rejection rates. As for [4], failures can still occur, due to the compromise criterion used in computing the guide trajectory.

Future work will focus on a more accurate formulation of C_{reach} to address this limitation. Despite these limitations, we have been able to tackle the generation of the complete motion, interpolated between the computed contact sequences, on the real HRP-2 robot with real time performances [7]. Our objective is now to formulate improved heuristics to guarantee the robustness of the planner regarding static equilibrium [9] and the transitions between postures.

Acknowledgements This research is supported by Euroc (project under FP7 Grant Agreement 608849), Entracte (ANR grant agreement 13-CORD-002-01), the ARO Contract W911NF-14-1-0437, the NSF award 1305286, and a grant from Boeing.

References

1. Amato, N.M., Burchan, O., Lucia, B., Dale, K., Jones, C., Vallejo, D.: Choosing good distance metrics and local planners for probabilistic roadmap methods. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (1998)

2. Baerlocher, P., Boulic, R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. Vis. Comput. **20**(6) (2004). doi:10.1007/s00371-004-0244-4
3. Baudouin, L., Perrin, N., Moulard, T., Lamiraux, F., Stasse, O., Yoshida, E.: Real-time replanning using 3D environment for humanoid robot. In: IEEE-RAS International Conference on Humanoid Robots (Humanoid'11). Bled, Slovenia (2011)
4. Bouyarmane, K., Escande, a., Lamiraux, F., Kheddar, a.: Potential field guide for humanoid multicontacts acyclic motion planning. In: 2009 IEEE International Conference on Robotics and Automation, pp. 1165–1170 (2009). doi:10.1109/ROBOT.2009.5152353
5. Bouyarmane, K., Kheddar, A.: Multi-contact stances planning for multiple agents. In: ICRA'11: International Conference on Robotics and Automation, Shanghai International Conference Center, Shanghai, Chine (2011)
6. Bretl, T., Rock, S., Latombe, J.C., Kennedy, B., Aghazarian, H.: Free-climbing with a multi-use robot. In: M.H.A. Jr., O. Khatib (eds.) ISER, Springer Tracts in Advanced Robotics, vol. 21, pp. 449–458. Springer, Heidelberg (2004)
7. Carpentier, J., Tonneau, S., Naveau, M., Stasse, O., Mansard, N.: A versatile and efficient pattern generator for generalized legged locomotion. In: Submitted to IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden (2016)
8. Deits, R., Tedrake, R.: Footstep planning on uneven terrain with mixed-integer convex optimization. In: 14th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2014, Madrid, Spain, November 18–20 (2014)
9. Del Prete, A., Tonneau, S., Mansard, N.: Fast algorithms to test robust static equilibrium for legged robots. In: Submitted to IEEE International Conference on Robotics and Automation (ICRA) (2016)
10. Escande, A., Kheddar, A., Miossec, S., Garsault, S.: Planning support contact-points for acyclic motions and experiments on HRP-2. In: Khatib, O., Kumar, V., Pappas, G.J. (eds.) ISER, Springer Tracts in Advanced Robotics, vol. 54, pp. 293–302. Springer, Heidelberg (2008)
11. Hauser, K., Bretl, T., Harada, K., Latombe, J.C.: Using motion primitives in probabilistic sample-based planning for humanoid robots. In: Akella, S., Amato, N.M., Huang, W.H., Mishra, B. (eds.) WAFR, Springer Tracts in Advanced Robot., vol. 47. Springer, Heidelberg (2006)
12. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: IEEE International Conference on Robotics and Automation (ICRA). Taipei, Taiwan (2003)
13. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. In: ACM Transactions on Graphics, vol. 21. ACM, New York, NY, USA (2002)
14. Mordatch, I., Lowrey, K., Todorov, E.: Ensemble-CIO: Full-Body Dynamic Motion Planning that Transfers to Physical Humanoids (2015)
15. Mordatch, I., Todorov, E., Popović, Z.: Discovery of complex behaviors through contact-invariant optimization. ACM Trans. Graph. **31**(4) (2012)
16. Nissoux, C., Siméon, T., Laumond, J.: Visibility based probabilistic roadmaps. In: Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems Hum. and Environment Friendly Robots with High Intelligent and Emoticon Quotes, October 17-21,1999, Hyundai Hotel, Kyongju, Korea (1999)
17. Park, C., Park, J.S., Tonneau, S., Mansard, N., Multon, F., Pettré, J., Manocha, D.: Dynamically balanced and plausible trajectory planning for human-like characters. In: To appear in Proceedings of I3D '16 Conference. Seatle, USA (2016)
18. Pettré, J., Laumond, J.P., Siméon, T.: A 2-stages locomotion planner for digital actors. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '03. Eurographics Association (2003)
19. Qiu, Z., Escande, A., Micaelli, A., Robert, T.: Human motions analysis and simulation based on a general criterion of stability. In: International Symposium on Digital Human Modeling (2011)
20. Siméon, T., Laumond, J., Cortes, J., Sahbani, A.: Manipulation planning with probabilistic roadmaps. Int. J. Robot. Res. **23**(7–8) (2004)

21. Stilman, M.: Global manipulation planning in robot joint space with task constraints. IEEE Trans. Robot. **26**(3) (2010)
22. Tonneau, S., Pettré, J., Multon, F.: Using task efficient contact configurations to animate creatures in arbitrary environments. Comput. Graph. **45**(0) (2014)

On the Dualities Between Grasping and Whole-Body Loco-Manipulation Tasks

Tamim Asfour, Júlia Borràs, Christian Mandery, Peter Kaiser, Eren Erdal Aksoy and Markus Grotz

1 Introduction

While efficient solutions have been found for walking in different scenarios [16, 25], including rough terrain and going up/down stairs, humanoid robots are still not able to robustly use their arms to gain stability, robustness and safety while executing locomotion tasks. The ability of reaching for supports can be crucial to increase robustness in tasks that require balance like walking or general locomotion, but also for increasing dexterity and maneuverability in complex manipulation tasks. Nevertheless, to execute such tasks in an autonomous way, we need to better understand the principles of whole-body coordination in humans, the variety of supporting whole-body postures available and how to transition between them.

Kinematically, a humanoid balancing with multi-contacts is equivalent to a closed kinematic chain mechanism, where the closed chains are formed through the contacts with the environment. Contacts can be modeled as joints that can range from 0 DoFs (a planar contact without friction) to 5 DoFs (point contact with friction) [34]. Closed kinematic chain mechanisms constitute a big family that includes parallel robots, cable driven robots, cooperative robotic arms, multi-legged robots and multi-fingered hands among others. Dynamically, when a humanoid uses its body to gain stability through contacts with its environment, the dynamic equations to achieve equilibrium are the same as those of closed kinematic chain mechanisms where the chains are closed through contacts with an object or with the environment. Although these parallelisms in kinematics and dynamics have been acknowledged by many authors [5, 27, 36, 37, 39], fewer works try to find connections and transfer of techniques between those fields of robotics [7, 8, 15, 42]. We are interested in using techniques developed in the context of grasping with multi-fingered hands to apply them to the study of whole-body postures with multi-contacts, where the body plays

T. Asfour (✉) · J. Borràs · C. Mandery · P. Kaiser · E.E. Aksoy · M. Grotz
Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology,
Karlsruhe, Germany
e-mail: asfour@kit.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_18

a double role: the role of the hand and the role of the manipulated object that can be moved through the contact reaction forces with the environment.

In this context, this paper presents our works exploring several different aspects of grasping that can be transferred to whole-body balance, such as grasping taxonomies [9], analysis of human motion data to classify and analyze grasps [31] and grasping affordances [23].

The main tools to understand how the hand can hold an object are the grasp taxonomies [4, 13, 19, 26]. Grasp taxonomies have been proven to be useful in many contexts: to provide a benchmark to test the abilities of a new robotic hand, to simplify grasp synthesis, to guide autonomous grasp choices, or to inspire hand design, among others. In our work in [9], we propose a classification of whole-body poses for balance exploring similar criteria as these used for taxonomies in grasping. Most grasping taxonomies define two main categories: precision and power grasping. In addition, Cutkosky classifies grasps according to object shape and tasks [13]. Kamakura according to task and the hand areas of contact [4, 26] and Feix et al. according to number/type of contacts and the configuration of the thumb [18]. We can directly use the idea of precision versus power grasping for whole-body poses: poses that use contact with the torso versus poses where contacts are realized only using the body end-effectors. But there is also an important difference since almost all whole-body poses are non-prehensile grasps, i.e., grasps that use the gravity to hold the object. In [9], we explore in detail criteria used in grasping to define a taxonomy of whole-body poses. A full taxonomy of whole-body poses can have many interesting applications and uses, such as a tool for autonomous decision making, a guide to design complex motions combining different whole-body poses, a way to simplify the control complexity, a benchmark to test abilities for humanoid robots, and a way to improve recognition of body poses and transitions between them.

Robotics in general, but particularly humanoid robotics, has always been inspired by biological human experience and the anatomy of the human body. In particular, grasping has collected human motion data to generate grasp synergies [11, 44] and classify and analyze most common grasps [17, 19], among other applications. However, human motions involving support contacts have almost not been studied [38] and even less how healthy subjects choose to make use of contacts with support surfaces. Our works like [31] explore the transitions between the whole-body poses of the proposed taxonomy by analyzing human motion data. While in classic locomotion actions such as walking and running the transitions between double and single support poses are very well-understood [29, 52], such transitions can become much more complex when e.g. the possibility of leaning against a surface with the hands is considered. In our work, we are interested in identifying balance poses during motions to be able to automatically perform a segmentation based on support poses. To study pose transition in [31], we analyze real human motion data captured with a marker-based motion capture system and post-processed using our unifying Master Motor Map (MMM) framework [2, 32, 33, 46], to gain information about the poses that are used while executing different locomotion and manipulation tasks like those shown in Fig. 1.

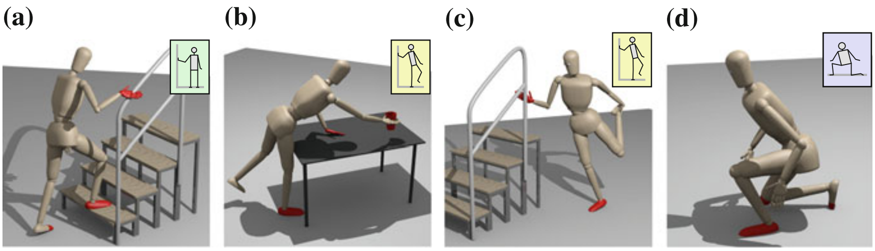


Fig. 1 When performing locomotion (**a**), manipulation (**b**), balancing (**c**) or kneeling (**d**) tasks, the human body can use a great variety of support poses to enhance its stability. Automatically detecting such support contacts allows for an automatic identification of the visited support poses and their transitions

Finally, we want to revisit the works where we explore the transfer of grasp affordances to the whole-body. The concept of affordances was originally introduced by Gibson [20] in the field of cognitive psychology for describing the perception of action possibilities. It states that objects suggest actions to the agent due to the object's shape and the agent's capabilities. A chair for example affords *sitting*, a cup *drinking* and a staircase *climbing*. Various works apply the concept of affordances to the field of grasping and manipulation, primarily for learning grasp affordances, e.g. by initial visual perception and subsequent interaction with an object [14, 41] or by focusing on either haptic [6] or visual [40] perception. In our previous work, we introduced the concept of Object-Action Complexes (OACs) to formalize affordances and link objects and actions into co-joint representations of sensorimotor behaviors in the robotic context [28]. In [23, 24], whole-body affordance hypothesis are defined as an association of a whole-body stable action to a perceived primitive of the environment. The concept of affordances is applied to actions related to the whole body of a humanoid agent, particularly actions for stabilization and combinations of whole-body locomotion and manipulation actions, i.e. loco-manipulation actions. Based on previous approaches like [6, 41], these works aim at deriving, refining and utilizing whole-body affordances like holding, leaning, stepping-on or supporting in unknown environments.

In this work, we present a summary review of these works to visualize successful transfer of knowledge from grasping to whole-body motion analysis with multi-contacts, and we point out directions of current and future work following the same idea of research. The paper is organized as follows. Section 2 summarizes the work where the taxonomy of whole-body poses is defined, summarizing the used criteria for classification. Section 3 revises the results of the analysis of motion data, and the automatic generation of a graph of pose transitions that is compared to the taxonomy of the previous section. Section 4 revisits the works where we extract whole-body affordances from unknown scenes and we validate the approach with an experiment executed with ARMAR-IIIa [1]. Finally, Sect. 5 gives a summary and provides ideas for current and future research.

2 Taxonomy of Whole-Body Support Poses

When considering the whole body interacting with the environment, there is a wide range of different postures that the robot can adopt. In [9], we were interested in those poses that use contacts for balancing. Then, the limb end-effectors that are not used for balancing can be used to perform other manipulation tasks. This way, we provide a framework for loco-manipulation poses. In other words, contacts with environmental elements that do not provide support are not considered for the taxonomy classification. For instance, in Fig. 2, green marked contacts define the support pose, while the rest are contacts intended to manipulate the environment that do not affect the support pose definition.

In Table 1, the taxonomy proposed in [9] is shown. It contains a total of 46 classes, divided into three main categories: standing, kneeling and resting. Each row corresponds to different number of supports, and in each row, different columns correspond to different contact types (see contact type legend at the bottom left corner of Table 1). In addition, colors differentiate type of leg supports and poses under the gray area use line contacts (with arms or legs). The lines between boxes indicate hypothesis of pose transitions between poses assuming only one change of support at a time.

The criteria considered for the definition of the taxonomy are

1. **Number of contacts**: One of the first relevant characteristics that greatly modifies the complexity of a motion is the number of contacts and supports with the environment. Kinematically, each support creates a new closed kinematic loop, and therefore, reduces by one the dimension of the feasible configuration space [3, 22]. Dynamically, planning of complex motions tested on humanoid robots report higher execution times per higher number of supports [43].
2. **Type of contacts**: From the control point of view, the nature of the contact used to provide the support [12, 35] and the part of the body that performs the support are relevant and important, because the resultant kinematics of the robot changes accordingly. A fingertip contact is usually modelled as a point contact with friction, the foot support as plane contacts and arm leaning can be modelled using line with friction model [34]. Figure 3 shows the types of support that we considered for the taxonomy with the legs and arms. To keep our taxonomy simple, we consider only 5 types: hold, palm, arm, feet, and knee support. These lead to the

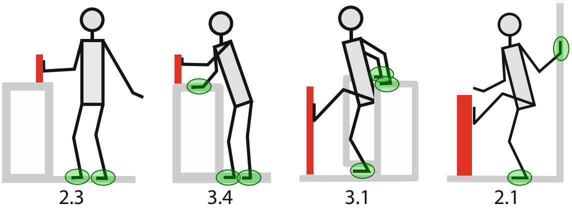
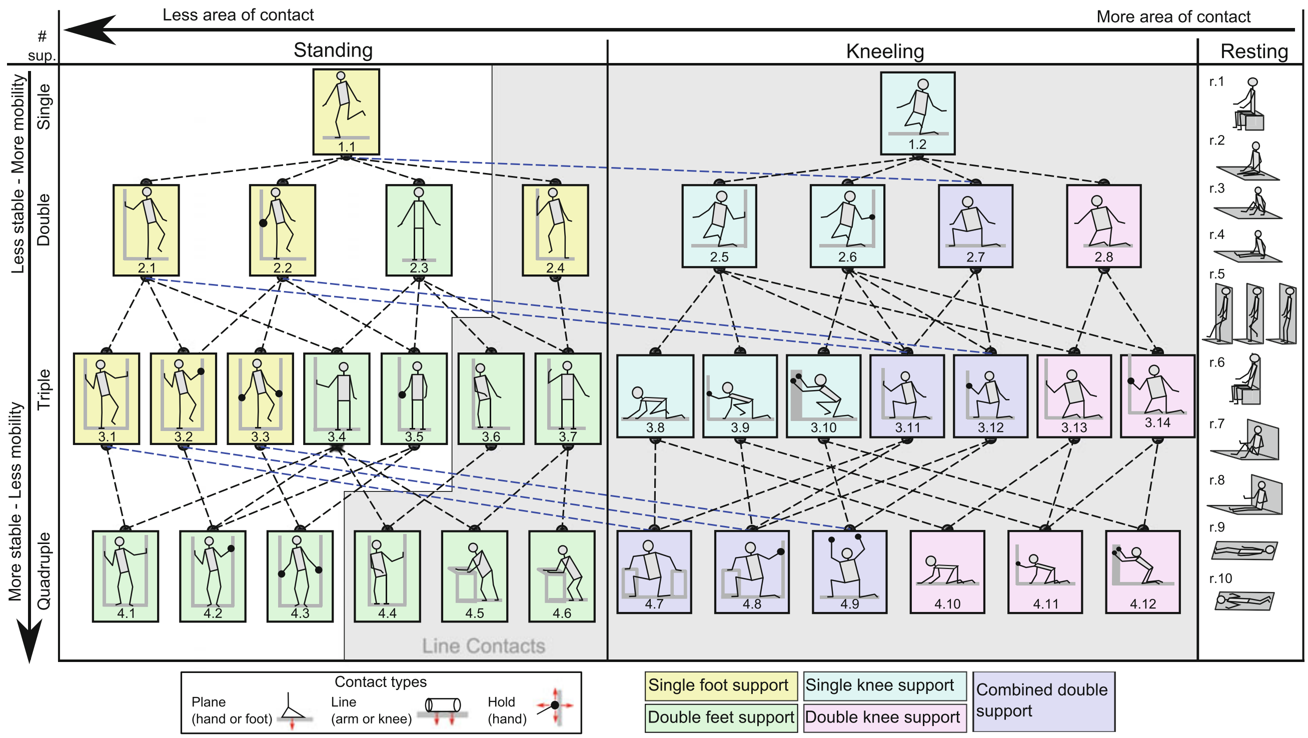


Fig. 2 The support poses to perform the task of hit an object are defined by the contacts highlighted in *green*. The numbers under the sketches refer to the id number of the support class in Table 1

Table 1 Taxonomy of whole-body support poses



The proposed taxonomy has 46 classes, including 18 standing poses, 18 kneeling poses and 10 resting configurations. Sketches represent all the ranges of poses with the same number of supports and type of contact. Each class includes the symmetric cases when applicable. The lines provide hypothetical road maps to transition from one pose to another, assuming one contact change at a time. Lines also provide a hierarchy among the poses. Blue lines represent transitions between different categories (from standing to kneeling).

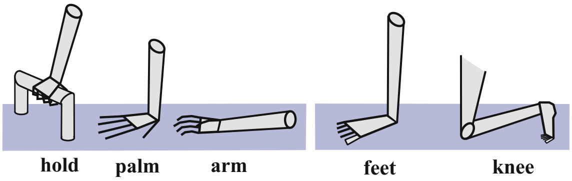


Fig. 3 Types of support contacts with arms and legs

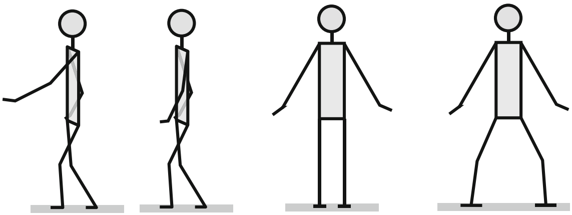


Fig. 4 Different body shapes for the pose 2.3

consideration of 51 combinations from which we have selected 36 (corresponding to the standing and kneeling poses). This choice has been done assuming that some combinations, while feasible, are not common.

3. **Shape of the environment**: Many grasping taxonomies include the shape of the object as a criteria for grasp selection. Indeed, object shape and size have a great influence on the ability for grasping and manipulation. However, there is a fundamental difference between hand grasping and whole-body poses: the need of gravity to reach force closure. A hand grasp will always start with no contacts at all, and after grasping, it may or may not start a manipulation motion that can be maintaining constant contacts (in-grasp manipulation) or performing re-grasps [30]. On the contrary, a whole-body grasp is always part of a motion sequence of re-grasps that will always start with at least one contact with the environment (even if one of the phases has no contact as in a running locomotion or jumping). For this reason, we believe that whole-body grasp choice will not depend as much on the shape of the environment, but on the task/motion the pose occurs. Therefore, our taxonomy does not include this criteria of classification.
4. **Shape of the body**: While we believe shape of the environment is not relevant in our case, the shape of the body performing the pose is relevant because it depends on the task and can influence the transitions between different poses. For instance, the shape of the body on the pose 3.4 when walking with a handrail will be different than going upstairs with a handrail. Also, if when performing a locomotion, the shape of the body in a double foot support pose (pose 2.3) contains a hand reaching further, like in the left figure of Fig. 4, it is very probable that the following pose will be a one with hand contact. However, the number of shapes that each pose can adopt is very large and the size of the taxonomy grows exponentially. Therefore, we will classify shape poses in a different hierarchy of classification, that is left for future work.

5. **Stability**: The taxonomy in Table 1 is organized so that the less stable poses, with less number of supports lie on the upper left side, while the most stable ones on the lower right side, assuming that the more number of contacts and the larger the surfaces of contact, the more stable the robot is. Works like [21] show that there is a trade-off between stability and maneuverability during a goal-directed whole-body movements. In the taxonomy, we observe a similar trade-off with mobility versus stability. However, it has to be noted that inside any class it is possible to obtain different levels of stability depending on the support region [10] (that greatly depends on the body shape) and the sum of the contact wrenches [50].
6. **Power grasps versus resting poses**: In addition to the standing and kneeling poses we have added 10 extra classes where there is contact with the torso. We call them resting poses and they are the equivalent to power grasps where there is contact of the object with the palm. Poses from r.1 to r.4 are poses where still balance needs to be achieved, but the inclination of the torso needs to be controlled. Poses from r.5 to r.6 are stable provided that the areas of contact are flat and with friction. Finally, using poses from r.7 to r.10 the robot is unlikely to lose balance and can be considered safe and completely in rest, but with very limited mobility.

   At this stage of work, no transitions are shown between resting poses and the rest of the table. Such transitions are more complex and require further analysis that will be left for future work.
7. **Pose transitions and motions**: In the next section, we will show how we have studied support pose transitions by analyzing human motion capture data. However, the taxonomy provides preliminary hypothesis of possible pose transitions using lines connecting poses in the taxonomy. Physically, a transition between two classes can happen by first imposing the constraints of the current and destination class, and then shifting to only the constraints of the destination class. This induces the definition of two types of motions:

   a. **Inside class motion**: A purely manipulation action will happen inside a single class. It includes other manipulation motions and therefore, extra contacts with objects, always with the objective of manipulation. As a manipulation motion, it can be semantically segmented and interpreted as done in [48].
   b. **Transition class motion**: motions that define a transition between poses. The motion still occurs inside a class, but the motion consists in the shifting towards a destination class, as part of a locomotion. For instance, a double feet support motion that shifts towards a right foot support (2.3 → 1.1). These are the kind of transitions that are studied in [31] and summarized in the next section.

   Note that both motions happen always inside the same support class, but in the second case, the destination class is relevant for the motion definition.

3 Detection of Whole-Body Poses and Segmentation

The framework presented in Sect. 2 introduces a set of segmentation criteria for a given motion that, provided that we can differentiate support contacts and manipulation contacts, subdivides a motion into pieces that can be related to types of actions. For actions identified as manipulation (inside pose motion), further segmentation based on the manipulation contacts can be performed [48], providing a hierarchy of segments distinguishing between the locomotion and the manipulation parts of an action. In the work [31], we proposed a method to detect support contacts that allows us to automatically segment motion data based on the support poses. This allows us to analyze support pose transitions during 121 loco-manipulation motions recorded using an optical marker-based Vicon MX motion capture system. This motion analysis can provide a better semantic understanding of complex locomotion and manipulation actions for imitation learning and autonomous decision making applications. Our motions recordings contain also information of the location and movement of the environmental elements, such as manipulated objects or objects to provide support. The KIT Whole-Body Human Motion Database [32] contains a large set of motions, providing raw motion capture data, corresponding time-synchronized video recordings and processed motions. The motions recorded for the work [31] can be found in the KIT Whole-Body Human Motion Database.[1]

Finally, the motions are post-processed using the The Master Motor Map (MMM) framework [33, 46]. This provides an open-source framework for capturing, representing and processing human motion. It includes a unifying reference model of the human body for the capturing and analysis of motion from different human subjects. The kinematic properties of this MMM reference model are based on existing biomechanical analysis by Winter [51] and allow the representation of whole-body motions using 58 degrees of freedom (DoF): 6 for the root pose and 52 for the body torso, extremities and head.

Support poses of the human subject are detected by analyzing the relation of the MMM reference model to the floor and environmental elements. For this purpose, we only consider objects which exhibit low movement during the recorded motion as suitable environmental elements to provide support. For every motion frame, we use the forward kinematics of the reference model to calculate the poses of the model segments that we consider for providing supports. These model segments represent the hands, feet, elbows and knees of the human body.

A segment s of the reference model is recognized as a support if two criteria are fulfilled. First, the distance of s to an environmental element must be lower than a threshold $\delta_{dist}(s)$. Distances to environmental elements are computed as the distances between pairs of closest points from the respective models with triangle-level accuracy using Simox [47]. Additionally, the speed of segment s, computed from smoothed velocity vectors, has to remain below a threshold $\delta_{vel}(s)$ for a certain

[1] See https://motion-database.humanoids.kit.edu/details/motions/<ID>/ with ID $\in$ {383, 385, 410, 412, 415, 456, 460, 463, 515, 516, 517, 520, 521, 523, 527, 529, 530, 531, 597, 598, 599, 600, 601, 604, 606, 607}.

number of frames, starting with the frame where the support is first recognized. The thresholds are chosen empirically: $\delta_{vel} = 200\,\frac{\text{mm}}{\text{s}}$, $\delta_{dist}(Feet) = \delta_{dist}(Hands) = 15\,\text{mm}$, $\delta_{dist}(Knees) = 35\,\text{mm}$ and $\delta_{dist}(Elbows) = 30\,\text{mm}$. The support pose is defined by the contacts that are providing support to the subject. We ignore parts of the motion where the human body is not supported at all as an empty support pose, e.g. during running. Also, some practical assumptions are used, such as that a knee support also implies a foot support. We have manually validated the segmentation method error by exploring frame by frame the detected support segments. Results can be seen in [31]. They show that about 4.5% of the poses are missed, but the missed poses are always double foot supports (with or without hand). Only 2.1% of the poses are incorrectly detected.

3.1 Data Driven Analysis of Transitions Between Whole-Body Support Poses

Without taking into account kneeling motions, we have recorded and analyzed 110 motions including locomotion, loco-manipulation and balancing tasks. In the following, we present some analysis on the most common pose transitions. We ignore kneeling motions because we do not have enough data yet to get significant results. In every motion, both the initial and the final pose are double foot supports and the time spent on these poses is arbitrary. Therefore, they have been ignored for the statistical analysis. Without counting them, we have automatically identified a total of 1323 pose transitions lasting a total time of 541.48 s (9.02 min). In Table 2, each cell represents the transition going from the pose indicated by the row name to the pose indicted by the column name. In each cell, we show first the percentage of

Table 2 Percentages of appearances and time spent for each transition (%appearance, %time)

	1Foot	1Foot-1Hand	2Feet	2Feet-1Hand	2Feet-2Hands	1Foot-2Hands	Totals × pose
1Foot	4.38, 5.69%	9.30, 7.90%	22.90, 25.56%	0.15, 0.26%	–	0.08, 0.04%	36.81, 39.44%
1Foot-1Hand	9.15, 3.64%	1.81, 2.26%	0.08, 0.03%	12.24, 16.59%	0.08, 0.02%	0.15, 0.02%	23.51, 32.57%
2Feet	16.02, 10.05%	0.15, 0.04%	×	3.48, 2.23%	0.08, 0.06%	–	19.73, 12.38%
2Feet-1Hand	0.23, 0.07%	11.72, 4.38%	4.61, 5.31%	×	0.98, 0.15%	–	17.54, 9.92%
2Feet-2Hands	–	–	–	0.83, 1.22%	×	0.68, 0.75%	1.51, 1.97%
1Foot-2Hands	–	0.53, 1.27%	–	–	0.38, 2.45%	×	0.91, 3.72%

occurrence of the transition with respect to the total number of transitions detected, and secondly the percentage of time spent on the origin pose before reaching the destination pose, with respect to the total time of all motions. The last column is the accumulation of percentages per each pose, and the rows are sorted from the most to the least common pose. According to Table 2, the most common transitions are 1Foot→2Feet (22.90% of appearance) and 2Feet→1Foot (16.02% of appearance). These are the same transitions of walking that have been widely studied. Although all motions contain some steps of normal walking, they also involve hand supports, and therefore, these transitions show different behaviours because they are part of a more complex set of transitions. It must be noted that the loop transitions 1Foot→1Foot, and 1Foot-1Hand→1Foot-1Hand are mostly missed double foot supports and we will not include them in the analysis.

Figure 5 shows the automatically generated transition graph, considering also the start and end poses of each motion and all the kneeling motions. Each edge

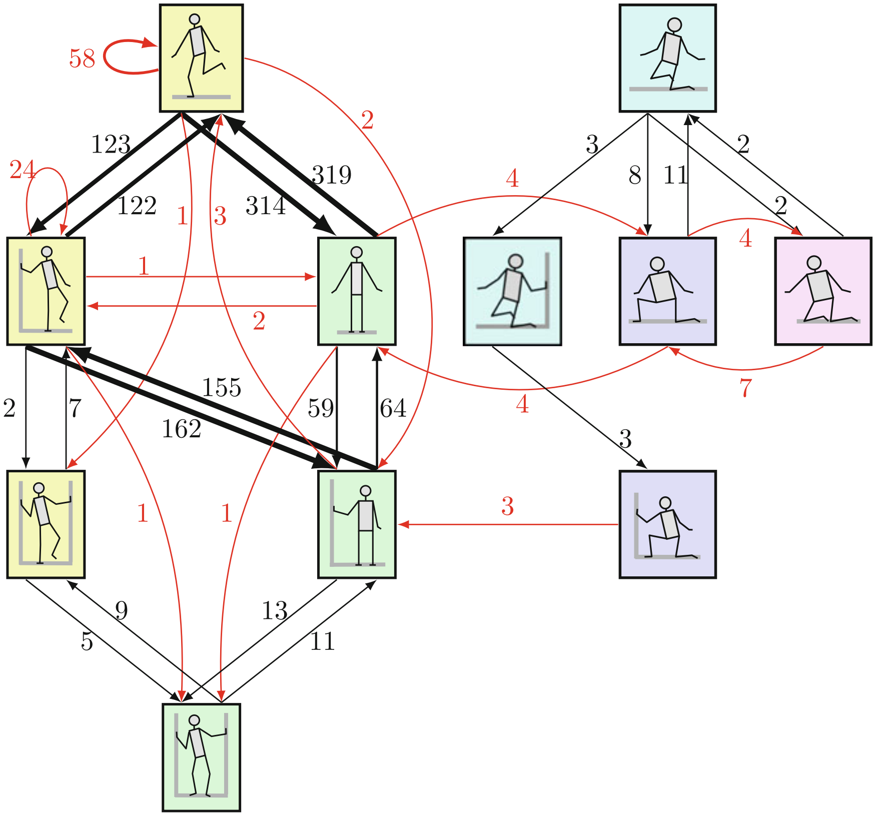


Fig. 5 Transition graph of whole-body pose transitions automatically generated from the analyzed motions. Labels on edges indicate the number of transitions found of each type

corresponds to a transition, and their labels to the number of times we have found it. Edges plotted in red correspond to transitions where two simultaneous changes of contacts occur. In the taxonomy of Table 1, we assumed that only one change of support should be allowed per transition. While this is still desirable for robotics, it is also obvious that some human transitions involve two contact changes. For instance, in push recovery motions, humans usually lean on the wall using both arms at the same time to increase stability. Many of the red edge transitions in Fig. 5 occur in balancing tasks. In the transition graph shown in Fig. 5, we can quickly see that red edges are of significantly lower frequency than the black ones, except the loop edges in the 1Foot and 1Hand-1Foot poses, that are caused by either jumps or missed double foot supports. They correspond to the 4.5% transitions missed by our segmentation method reported before. This data-driven transition graph is influenced by the type of motions we have analyzed, using only one handle or one hand support. Only balancing poses reach the four support poses. In future work, we will analyze walking motions with handles on both sides.

Most of the hypothetical transitions in Table 1 are correct, except 1Foot → 1Foot1Knee that does not appear in real data. This is because the subject uses support with the tip of the foot until contact is reached with the knee, and this is detected as a foot support that corresponds to a tip-toe support. Therefore, all the edges in red between double foot support (with and without hand) and kneeling are correctly detected and should be corrected in our taxonomy in Table 1. In the future, we will study if we should distinguish between tip-toe and sole support. Figure 6 shows an example of segmentation result, corresponding to the time line of a motion where the subject goes upstairs using a handle on his right side. In blue, we show the long locomotion transitions. The supporting pose for these transitions alternates between 1Foot-1Hand, used to swing forward the foot not in contact, and 1Foot, used to swing forward both the handle hand and the foot not in contact. This is because we only provide one handle. Another interesting thing to notice is that the short locomotion transitions appear in clusters, composed by a sequence of two transitions. We have observed this in many of the motions and we have observed that the order of the transitions inside these clusters does not matter, just the start and end poses. We believe that each cluster could be considered as a composite transition where several contact changes occur. As future work, we want to detect and model these clusters to

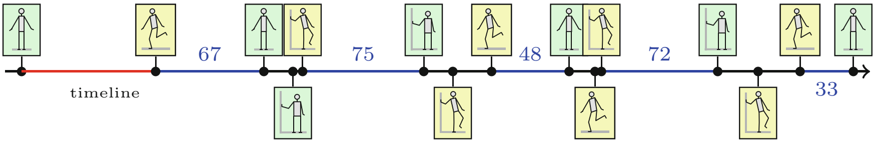


Fig. 6 Result of the segmentation for one of the motions upstairs with handle. The segment shown in *red* represents the initial pose transition, that has an arbitrary length. *Blue* segments represent transitions where the foot swings. *Blue* labels/numbers indicate transition durations. We can see that the human alternates between single foot support swing and 1Foot-1Hand support swing using the handle

identify rules that allow us to automatically generate sequences of feasible transitions according to extremities available for contacts.

4 Whole-Body Affordances

Grasp affordances rely on perception methods, either visual or haptic, to perceive the geometry of the object, and then associate grasp strategies according to the recognized geometric shapes [6]. Similarly, in [23, 24] we relay on a visual perception system with active cameras that can collect point clouds, register them and then extract geometric primitives from an unknown scene. In [23] we proposed to assign hypotheses for whole-body affordances like *support*, *lean*, *grasp* or *hold* to environmental primitives based on shape, size and orientation. Large vertical planes for instance are assumed to indicate *lean*-affordances. These kind of affordances are of basic importance for whole-body stabilization. However, further possible whole-body affordances exist and are of special interest when manipulating large, and possibly heavy, objects, for instance for removing debris from a blocked pathway. Examples for whole-body affordances indicating manipulability of objects are *pushability* and *liftability*, which are experimentally evaluated in [24]. The association of affordances is based on a set of rules shown in Table 3. While an exhaustive evaluation of the available types of whole-body affordances still remains to be done, *pushability* and *liftability* are certainly essential. The work show that we can integrate and evaluate the processes of affordance perception, validation and utilization on a real-world robotic system considering all the affordance types.

The constants λ_i from Table 3 are currently application specific. However, we think that there is a fixed set of affordance extraction parameters that will work reasonably well for our scenarios due to the following reasons:

- Research shows that agents infer affordances based on a body-scaled metric, i.e. with respect to the proportions of their bodies [49].
- We primarily focus our studies to disaster scenarios that contain at least partly intact man-made elements like doors, handrails or staircases. These elements usually have standardized dimensions known beforehand.

Figure 7 visualize the environmental primitives and their associated affordances from point cloud example corresponding to a staircase scenario. The primitives are assigned meaningful whole-body affordances based on the rules from Table 3. The proposed framework successfully identifies the existing cylindrical and planar primitives. More examples of different scenes can be found in [24]. The strategies for affordance extraction are purely based on visual perception and are therefore only *affordance hypotheses* subject to further investigation and validation by the robot. In [23], precomputed reachability maps help to discard non utilizable affordances. In [24] there is no reliable mechanism for verifying the existence of affordanceswithout

Table 3 Example of a set of rules for affordance derivation and possible validation strategies. The operator $\uparrow$ tells if two vectors point into the same direction. The values λ_i are implementation-specific constants

Affordance	Shape	Parameters	Conditions	Valid.
Support (S)	Planar	Normal $\boldsymbol{n}$	$\boldsymbol{n} \uparrow z_{world}$	(1a)
		Area a	$a \geq \lambda_1$	
Lean (Ln)	Planar	Normal $\boldsymbol{n}$	$\boldsymbol{n} \perp z_{world}$	(1a)
		Area a	$a \geq \lambda_2$	
Grasp (G)	Planar	Normal $\boldsymbol{n}$	$a \in [\lambda_3, \lambda_4]$	(3)
		Area a		
	Cylindrical	Radius r	$r \in [\lambda_5, \lambda_6]$	
	Spherical	Direction $\boldsymbol{d}$	$\Vert\boldsymbol{d}\Vert \leq \lambda_7$	
		Radius r	$r \in [\lambda_8, \lambda_9]$	
Hold (H)	Cylindrical	Radius r	$r \in [\lambda_{10}, \lambda_{11}]$	(2a)
		Direction $\boldsymbol{d}$	$\Vert\boldsymbol{d}\Vert \geq \lambda_{12}$	
Push (P)	Planar	Normal $\boldsymbol{n}$	$\boldsymbol{n} \perp z_{world}$	(1b)
		Area a	$a \leq \lambda_{13}$	
Lift (Lf)	Planar	Normal $\boldsymbol{n}$	$a \leq \lambda_{15}$	(2b)
		Area a		
	Cylindrical	Radius r	$r \leq \lambda_{15}$	
		Direction $\boldsymbol{d}$	$\Vert\boldsymbol{d}\Vert \leq \lambda_{16}$	
	Spherical	Radius r	$r \leq \lambda_{17}$	

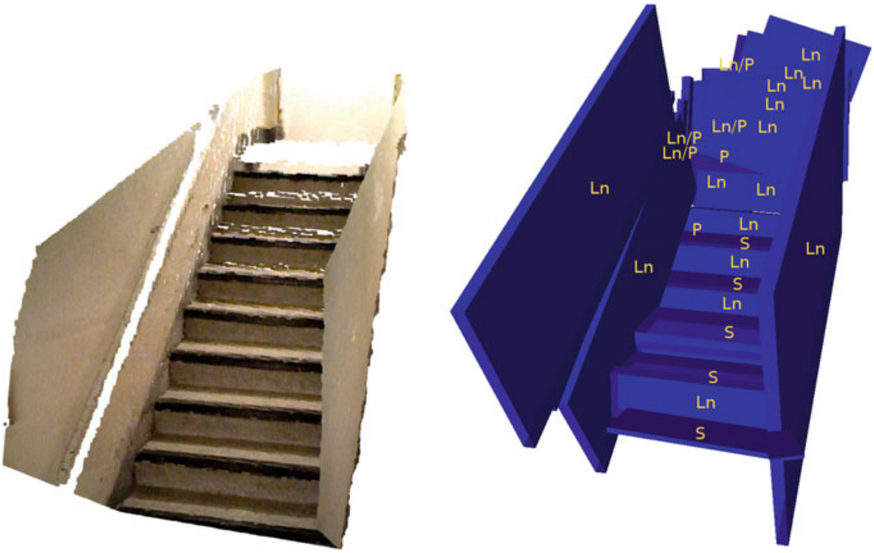


Fig. 7 Example of the results of the affordance extraction process (*right*) from a segmented point cloud (*left*). The example scenario is staircase. The affordance tags **S**, **Ln**, **G**, **P** and **Lf** refer to Table 3. For more examples see [24]

establishing contact to the underlying primitives. Referring to Table 3, different force-based validation strategies exist based on the affordance hypothesis to investigate:

1. Exert a force along the primitive's normal **n** and compare the resistance force against a minimum ϑ_1 (1a) or a maximum ϑ_2 (1b).
2. Grasp the primitive and exert forces perpendicular to the primitive's direction **d**. Compare the resistance force against a minimum ϑ_3 (2a) or a maximum ϑ_4 (2b).
3. Push the primitive and perceive the caused effect.

Considering further sensor modalities apart from contact forces is of interest and can lead to more sophisticated and accurate validation strategies. Validating the *pushability* of a very light object for instance might not result in a reliable resistance force feedback. Possible solutions for cases like this include tactile feedback or the comparison of RGB-D images before and after the push, similar to [45]. These strategies were validated with an experiment carried out on the humanoid robot

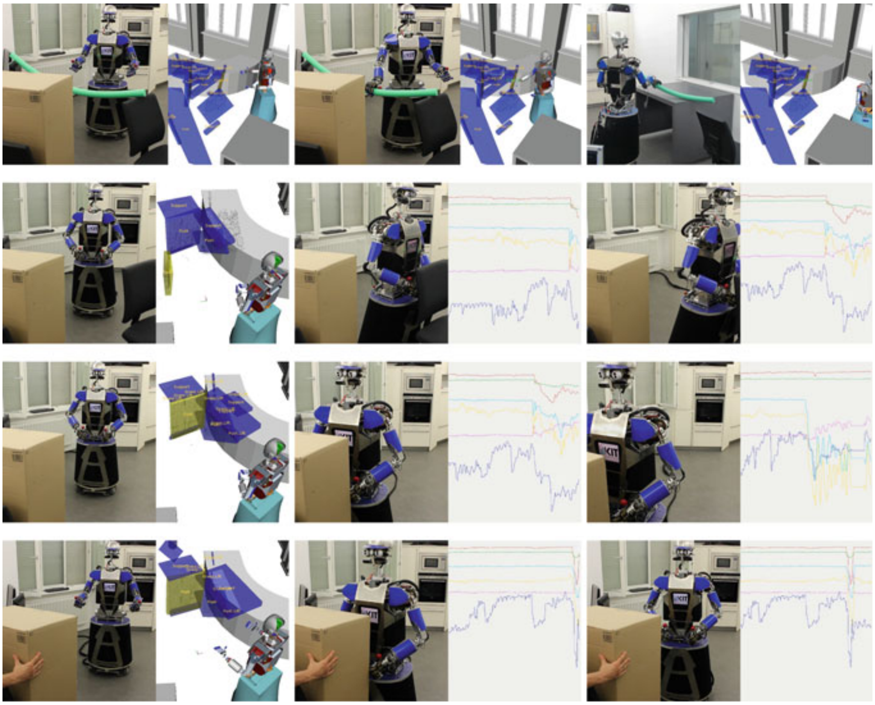

(a) Perception (b) Validation (c) Execution

Fig. 8 The three stages of perception, validation and execution of whole-body affordances in four different scenarios: A pipe that can be grasped and lifted (*first row*), a chair that can be pushed (*second row*), a box that can be pushed (*third row*) and a box that is fixed and cannot be pushed (*fourth row*). The plots visualize the force amplitudes (y-axis) measured in the robot's left wrist over time (x-axis), while the *blue* curve represents the force in pushing direction

ARMAR-III, demonstrating the perception and validation of affordance hypotheses for *pushability* and *liftability*. In the experiment ARMAR-III is facing a cluttered arrangement of different obstacles, i.e. debris, that block its way: A chair, a box and a pipe (see Fig. 8, top left corner). The robot has no prior knowledge on the types or locations of the employed obstacles, the only information it gets results from the perceptual pipeline. Figure 8 displays snapshots of different stages of the experiment: perception (first column), validation (second column) and execution (third column). The perception stage displays the initial obstacle arrangement and its representation after the perceptual pipeline in terms of primitives and affordance hypotheses. The validation stage includes the establishment of contact with the selected primitive and the affordance validation based on the obstacle's resistant force. In the execution phase, the robot has validated the affordance in question and starts pushing or lifting the obstacle, respectively. The robot successfully identifies all three obstacles and starts by validating the liftability of the pipe (Fig. 8, first row). The validated liftability is then exploited for moving the obstacle away. In the next steps the robot identifies the chair and the box as pushable obstacles and validates these affordances accordingly (Fig. 8, second row, third row). The last row of Fig. 8 displays a repetition of the previous scene with a fixed box. The robot again assigns a pushability hypothesis to the box, but fails to validate this hypothesis. Hence, the corresponding push cannot be executed. A detailed description of the whole process is given in [24].

5 Conclusions

This work revisits several works from our previous work that explore transfer of techniques from grasping to whole-body loco-manipulation tasks. In this context, we have proposed a taxonomy of whole-body balancing poses containing 46 classes, divided into three main categories, considering number and type of support and possible transitions between support poses. We have analyzed known grasping criteria used to classify robot grasps, but focusing on the demands of whole-body poses. As opposed to grasping, we have given less relevance to environment shape and more to the type of contact the body uses to provide a support pose. We have also presented an analysis of support poses of more than 100 motion recordings showing different locomotion and manipulation tasks. Our method allowed us to retrieve the sequence of used support poses and the time spent in each of them, providing segmented representations of multi-contact whole-body motions. Although the most common pose transitions are the ones involved in walking, we have shown that the 1Foot-1Hand and the 2Foot-1Hand poses also play a crucial role in multi-contacts motions. The data-driven generated graph of transitions validates the transitions proposed in our taxonomy. We believe that our motion segmentation by support poses and time spent per transition provides a meaningful semantic representation of a motion. Finally, we have shown how the concept of grasp affordances can also be applied to whole-body affordances. Using common sense knowledge of a perceived unknown scene, whole-body affordances are assigned to geometries and are then validated through physical interaction with the scene.

This work opens the door to many exciting future directions. First, each class of poses in the taxonomy corresponds to an infinite number of possible body configurations depending on location and orientation of contacts and the body shape. Future work directions include finding the most relevant *whole-body eigen-grasps* based on the collected human motion data, that is, we will apply dimensionality reduction to deal with the large space of whole-body configuration and to determine whole-body eigen-grasps associated with support poses. Secondly, we are interested in analyzing our motion representations to find semantic rules that can help define new motions for different situations, with the objective of building a grammar of motion poses based on the introduced taxonomy. Storing each transition as a motion primitive, we are also interested in performing path planning at a semantic level based on support poses. Finally, we plan to use the extracted and validated affordances to generate sequences of whole-body poses that generate locomotions with multi-contacts, utilizing perceived location of the possible contacts and the learned motion primitives for each pose transition. In conclusion, our works present a step further in the comprehension of how humans can utilize their bodies to enhance stability for locomotion and manipulation tasks. We believe the proposed ideas have a lot of potential to be used in many areas of humanoid robotics.

Acknowledgements The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no 611832 (WALK-MAN) and grant agreement no 611909 (KoroiBot).

References

1. Asfour, T., Regenstein, K., Azad, P., Schröder, J., Vahrenkamp, N., Dillmann, R.: ARMAR-III: an integrated humanoid platform for sensory-motor control. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids)
2. Azad, P., Asfour, T., Dillmann, R.: Toward an unified representation for imitation of human motion on humanoids. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2558–2563 (2007)
3. Berenson, D., Srinivasa, S.S., Kuffner, J.: Task space regions: a framework for pose-constrained manipulation planning. Int. J. Robot. Res. **30**(12), 1435–1460 (2011)
4. Bernardin, K., Ogawara, K., Ikeuchi, K., Dillmann, R.: A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models. IEEE Trans. Robot. **21**(1), 47–57 (2005). doi:10.1109/TRO.2004.833816
5. Bicchi, A., Melchiorri, C., Balluchi, D.: On the mobility and manipulability of general multiple limb robots. IEEE Trans. Robot. Autom. **11**(2), 215–228 (1995)
6. Bierbaum, A., Rambow, M., Asfour, T., Dillmann, R.: Grasp affordances from multi-fingered tactile exploration using dynamic potential fields. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 168–174 (2009)
7. Borràs, J., Dollar, A.M.: Analyzing dexterous hands using a parallel robots framework. Auton. Robots **36**(1–2), 169–180 (2014)
8. Borràs, J., Dollar, A.M.: Dimensional synthesis of three-fingered robot hands for maximal precision manipulation workspace. Int. J. Robot. Res. **34**(14), 1731–1746 (2015)

9. Borràs, J., Asfour, T.: A whole-body pose taxonomy for loco-manipulation tasks. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1578–1585 (2015)
10. Bretl, T., Lall, S.: Testing static equilibrium for legged robots. IEEE Trans. Robot. **24**(4), 794–807 (2008)
11. Ciocarlie, M., Goldfeder, C., Allen, P.: Dimensionality reduction for hand-independent dexterous robotic grasping. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3270–3275 (2007)
12. Collette, C., Micaelli, A., Andriot, C., Lemerle, P.: Robust balance optimization control of humanoid robots with multiple non coplanar grasps and frictional contacts. In: ICRA 2008. IEEE International Conference on Robotics and Automation, pp. 3187–3193. IEEE (2008)
13. Cutkosky, M.R.: On grasp choice, grasp models, and the design of hands for manufacturing tasks. IEEE Trans. Robot. Autom. **5**(3), 269279 (1989)
14. Detry, R., Kraft, D., Kroemer, O., Bodenhagen, L., Peters, J., Krüger, N., Piater, J.: Learning grasp affordance densities. Paladyn J. Behav. Robot. **2**(1), 1–17 (2011)
15. Ebert-Uphoff, I., Voglewede, P., et al.: On the connections between cable-driven robots, parallel manipulators and grasping. IEEE Int. Conf. Robot. Autom. **5**, 4521–4526 (2004)
16. Englsberger, J., Ott, C., Albu-Schaffer, A.: Three-dimensional bipedal walking control using divergent component of motion. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2600–2607 (2013)
17. Feix, T., Bullock, I.M., Dollar, A.M.: Analysis of human grasping behavior: object characteristics and grasp type. IEEE Trans. Haptics **7**(3), 311–323 (2014)
18. Feix, T., Pawlik, R., Schmiedmayer, H.B., Romero, J., Kragic, D.: A comprehensive grasp taxonomy. In: Robotics, Science and Systems: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation (2009)
19. Feix, T., Romero, J., Schmiedmayer, H.B., Dollar, A.M., Kragic, D.: The grasp taxonomy of human grasp types. IEEE Trans. Hum.-Mach. Syst. (2015). In press
20. Gibson, J.J.: The Ecological Approach to Visual Perception. Psychology Press, Hove (1979)
21. Huang, H.J., Ahmed, A.A.: Tradeoff between stability and maneuverability during whole-body movements. PLoS One **6**(7), e21,815 (2011)
22. Jaillet, L., Porta, J.M.: Path planning under kinematic constraints by rapidly exploring manifolds. IEEE Trans. Robot. **29**(1), 105–117 (2013)
23. Kaiser, P., Gonzalez-Aguirre, D., Schültje, F., Sol, J.B., Vahrenkamp, N., Asfour, T.: Extracting whole-body affordances from multimodal exploration. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robots (2014)
24. Kaiser, P., Grotz, M., Aksoy, E.E., Do, M., Vahrenkamp, N., Asfour, T.: Validation of whole-body loco-manipulation affordances for pushability and liftability. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids) (2015)
25. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 2, p. 16201626 (2003)
26. Kamakura, N.: Te no katachi Te no ugoki. Ishiyaku, Tokyo (1989)
27. Kerr, J., Roth, B.: Analysis of multifingered hands. Int. J. Robot. Res. **4**(4), 3–17 (1986)
28. Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., Ude, A., Asfour, T., Kraft, D., Omrčen, D., Agostini, A., Dillmann, R.: Object-action complexes: grounded abstractions of sensorimotor processes. Robot. Auton. Syst. **59**, 740–757 (2011)
29. Kwon, T., Shin, S.Y.: Motion modeling for on-line locomotion synthesis. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 29–38. ACM (2005)
30. Ma, R.R., Dollar, A.M.: On dexterity and dexterous manipulation. In: 2011 15th International Conference on Advanced Robotics (ICAR), pp. 1–7. IEEE (2011)
31. Mandery, C., Borràs, J., Jochner, M., Asfour, T.: Analyzing whole-body pose transitions in multi-contact motions. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 5411–5418 (2015)

32. Mandery, C., Terlemez, O., Do, M., Vahrenkamp, N., Asfour, T.: The KIT whole-body human motion database. In: International Conference on Advanced Robotics (ICAR), pp. 329–336 (2015)
33. Mandery, C., Terlemez, O., Do, M., Vahrenkamp, N., Asfour, T.: Unifying representations and large-scale whole-body motion databases for studying human motion. IEEE Trans. Robot. **32**(4), 796–809 (2016)
34. Mason, M.T.: Chapter 4.3 Kinematic models of contact. Mechanics of Robotic Manipulation, pp. 86–88. The MIT Press, Cambridge (2001)
35. Mason, M.T.: Mechanics of Robotic Manipulation. MIT Press, Cambridge (2001)
36. Nakamura, Y.: Grasp and manipulation. J. Meas. Control **29**(3), 206–212 (1990) (Japanese)
37. Nakamura, Y., Nagai, K., Yoshikawa, T.: Dynamics and stability in coordination of multiple robotic mechanisms. Int. J. Robot. Res. **8**(2), 44–61 (1989)
38. Nori, F., Peters, J., Padois, V., Babic, J., Mistry, M., Ivaldi, S.: Whole-body motion in humans and humanoids. In: Workshop on New Research Frontiers for Intelligent Autonomous Systems (2014)
39. Orin, D., Oh, S.: Control of force distribution in robotic mechanisms containing closed kinematic chains. J. Dyn. Syst. Meas. Control **103**(2), 134–141 (1981)
40. Pas, A., Platt, R.: Localizing grasp affordances in 3-D points clouds using taubin quadric fitting. In: International Symposium on Experimental Robotics (ISER) (2014)
41. Popovi, M., Kraft, D., Bodenhagen, L., Baeski, E., Pugeault, N., Kragic, D., Asfour, T., Krüger, N.: A strategy for grasping unknown objects based on co-planarity and colour information. Robot. Auton. Syst. **58**(5), 551–565 (2010)
42. Porta, J.M., Ros, L., Bohigas, O., Manubens, M., Rosales, C., Jaillet, L.: The cuik suite: analyzing the motion closed-chain multibody systems. IEEE Robot. Autom. Mag. **21**(3), 105–114 (2014)
43. Saab, L., Ramos, O.E., Keith, F., Mansard, N., Soueres, P., Fourquet, J.Y.: Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. IEEE Trans. Robot. **29**(2), 346–362 (2013)
44. Santello, M., Flanders, M., Soechting, J.F.: Postural hand synergies for tool use. J. Neurosci. **18**(23), 10105–10115 (1998)
45. Schiebener, D., Ude, A., Asfour, T.: Physical interaction for segmentation of unknown textured and non-textured rigid objects. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4959–4966 (2014)
46. Terlemez, O., Ulbrich, S., Mandery, C., Do, M., Vahrenkamp, N., Asfour, T.: Master Motor Map (MMM) - framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 894–901 (2014)
47. Vahrenkamp, N., Kröhnert, M., Ulbrich, S., Asfour, T., Metta, G., Dillmann, R., Sandini, G.: Simox: A robotics toolbox for simulation, motion and grasp planning. In: International Conference on Intelligent Autonomous Systems (IAS), pp. 585–594 (2012)
48. Wächter, M., Schulz, S., Asfour, T., Aksoy, E., Wörgötter, F., Dillmann, R.: Action sequence reproduction based on automatic segmentation and object-action complexes. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp. 189–195 (2013)
49. Warren, W.H.: Perceiving affordances: visual guidance of stair climbing. J. Expe. Psychol. **10**(5), 683–703 (1984)
50. Wieber, P.B.: On the stability of walking systems. In: Proceedings of the International Workshop on Humanoid and Human Friendly Robotics (2002)
51. Winter, D.A.: Biomechanics and Motor Control of Human Movement, 4th edn. Wiley, Hoboken (2009)
52. Yin, K., Loken, K., van de Panne, M.: Simbicon: simple biped locomotion control. ACM Trans. Gr. **26**(3), 105–1–105–10 (2007)

Part III
Robot Planning and Navigation

Session Summary

We believe that motion planning is already well-established as a research field, but it is still evolving while research focuses are changing. Navigation is currently very actively studied along with recent rapid progress in drones and field robotics. Perception planning and localization is intensively studied for smooth and efficient navigation, and we have three papers on these topics in this session: relative topometric localization in globally inconsistent maps (M. Mazuran et al.), active multi-view object recognition and online feature selection (C. Potthast et al.), an iterative Kalman smoother for robust 3D localization and mapping (D. Kottas and S. Roumeliotis), and incremental sparse GP regression for continuous-time trajectory estimation & mapping (X. Yan et al.).

In the first keynote talk, Nancy Amato introduced challenges in this field and emphasized that fundamental theory and algorithms are still actively investigated. The graph-based (e.g., PRMs) and tree-based (e.g., RRTs) methods have proven to be very powerful, enabling us to solve many previously unsolvable problems. Algorithmic challenges include how to deal with high-DOF systems, narrow passages and constrained systems. Two papers were presented on this topic, fast sample-based planning of dynamic systems by zero-control linearization-based steering (D.G. Caldwell and N. Correll) and a little paradoxical topic, deterministic sampling-based motion planning (L. Janson et al.).

Other challenges are uncertainty and unknown environments that require replanning. Recently, FIRM (Feedback-based Information RoadMap) extended the roadmap to belief space eliminating the need for replanning. Learning is now being integrated into planning. Three papers were presented about navigation under uncertainty: Bayesian learning for safe high-speed navigation in unknown environments (C. Richter et al.), Monte Carlo motion planning for robot trajectory optimization under uncertainty (L. Janson et al.) and inference-enabled information-theoretic exploration of continuous action spaces (S. Bai et al.). Environment-aware planning is also a related issue and two papers tackle this topic: a computational framework for environment-aware robotic manipulation planning (M. Gabiccini) and small and adrift with self-control: using the environment to improve autonomy (M.A. Hsieh et al.).

Finally, Amato addressed a very exciting challenge: human-in-the-loop planning. Why should we plan from scratch every time? Why don't we rely on human's guidance/input that is difficult to quantify automatically and required for co-operation? However, the information required for this planning may be different from existing formalism such as cost or complexity. Her latest research shows that humans can provide guidance and reduce search space leaving the planner to do fine-grained planning.

Eiichi Yoshida's next keynote talk was about "human(oid) motion planning" that is a more application-oriented topic. In the earlier stage of his research, he regarded a humanoid robot as just a robot with many DOFs that sampling-based approaches have advantages for planning. A variety of whole-body humanoid motion can be generated by combining generalized inverse kinematics as a local method of a sampling-based planner. In the Humanoids session, S. Tonneau et al. presented a related topic on a reachability-based planner for sequences of acyclic contacts in cluttered environments. The title of talk implies "rom humanoid planning to human planning". Humanoid motion planning recently makes Yoshida more and more interested in human motions: why humans do this motion rather than other motions, do they optimize something (inverse optimal control), and if yes, is it possible to apply the optimization to humanoid? Actually, optimization is another challenge actively studied in the domain of planning: a couple of papers report this topic: unifying classical and optimization-based methods for robot tracking control with control contract (I.R. Manchester et al.), path following for mobile manipulators (R. Gill et al.) and role of vision in locomotor trajectories formation (J.-P. Laumond) in the Humanoids session. Yoshida then presents some work of human motion analysis and its reproduction by a humanoid: not only for entertainment use, but also toward a "physical mannequin" that replaces human subjects for wearable power suits or assistive devices. Amato's human-in-the-loop planning is indeed one of the promising solutions. By immersive tele-existence interface that makes the human embody" in the robot, a humanoid can be planned in a more intuitive way. Yoshida concluded his talk by noting that implementing a "human" motion planner will allow a humanoid robot to collaborate with humans in a more natural and transparent way.

Bayesian Learning for Safe High-Speed Navigation in Unknown Environments

Charles Richter, William Vega-Brown and Nicholas Roy

1 Introduction

A common planning strategy in unknown environments is the receding-horizon approach: plan a partial trajectory given the current (partial) map knowledge, and begin to execute it while re-planning. By repeatedly re-planning, the robot can react to new map information as it is perceived. However, avoiding collision in a receding-horizon setting can be difficult, since the planner must not only ensure that its chosen actions are collision-free within the planning horizon, but it must also consider what actions will be feasible for the robot after the planned trajectory has been completed. To guarantee safety, a receding-horizon planner must plan trajectories that are known to be collision-free for all time [10]. This constraint is often satisfied through hand-coded rules or contingency plans, such as ensuring the existence of a collision-free emergency-stop maneuver or cyclic holding pattern [2, 23]. If the map is partially known, a safe planner must conservatively assume that every unknown cell may contain an obstacle, and therefore must confine its trajectories to lie within the observed free space. Figure 1a, b illustrate this scenario for a robot approaching a blind corner while guaranteeing safety. As the robot approaches the corner, it slows dramatically to preserve its ability to complete a feasible emergency-stop maneuver before entering unknown space. We use this "baseline" safe planner for comparison in this paper.

Safety constraints imply an assumption that driving into unknown regions of the map is always dangerous. However, our central claim in this line of research is that

C. Richter (✉) · W. Vega-Brown · N. Roy
Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
e-mail: car@mit.edu

W. Vega-Brown
e-mail: wrvb@mit.edu

N. Roy
e-mail: nickroy@mit.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_19

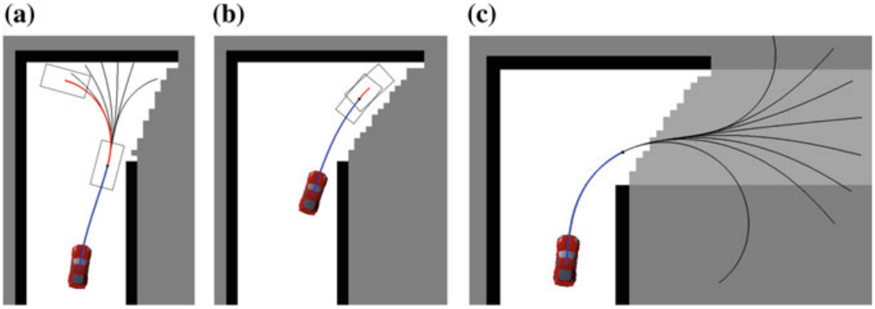


Fig. 1 Actions chosen for a robot approaching a blind corner while guaranteeing safety (**a**)–(**b**). Chosen trajectories are drawn in *blue*, and feasible emergency-stop maneuvers are drawn in *red*. Emergency-stop actions that would cause the robot to collide with obstacles or unknown cells are drawn in *black*. The safety constraint requires the robot to slow from 4 m/s in **a** to 1 m/s in **b** to preserve the ability to stop within known free space. In **c**, our solution plans a 6 m/s turn, which commits the robot to entering the unknown, but the planner's experience predicts a low risk of collision. The *light gray region* shows the hallway of the true hidden map

this assumption may be overly conservative. In many cases, the agent can safely drive at high speeds into the unknown if it is equipped with an accurate model of collision probability based on previous experience in similar situations [20]. For instance, many buildings have straight hallways with 90° turns and intersections. If a robot observes a hallway with a turn, and has trained in similar environments, it should be able to reason with high certainty that the hallway will continue, and that there will be free space around the turn. Figure 1c shows our proposed solution planning a high-speed turn around a blind corner, violating the safety constraint. Emergency-stop trajectories from the chosen action (illustrated in black) all enter unknown space. Yet, this type of maneuver is often empirically safe for hallway environments and our solution correctly infers that it is not excessively risky.

One way to estimate the likelihood of collision in unknown space might be to infer the probability that certain unknown map regions are occupied, using sensor measurements and an accurate prior distribution over maps. Unfortunately, modeling an accurate distribution over real-world environments would be extremely difficult due to the very high dimensionality of building-sized occupancy grid maps, the strong assumption of independence between map cells, and the richness of man-made and natural spaces which resist compact parameterization. Without significant modeling effort, or restriction of the problem domain to specific environments, the best prior knowledge we can reasonably provide is to say that every map is equally likely. Of course, this prior is very inaccurate and completely unhelpful for planning, and prevents the agent from exploiting any intuitive knowledge of "typical" environments. To compensate for this missing knowledge, we adopt a machine learning approach to predict collision probability from training data, which implicitly captures the relevant traits of our training environments rather than modeling the map distribution explicitly. This approach leads to a much lower-dimensional model that is significantly easier to learn and efficient enough to run online.

In the next section, we develop the POMDP formulation of our problem, which lays the foundation for decision theoretic planning. We then derive our approximations to the POMDP to make the problem tractable, which give rise to our learned model of collision probability. Then, we discuss how we learn that model and how we encode safety constraints as a prior over collision probability to help our planner remain safe in novel environments for which it has no relevant training data. Finally, we will present simulation and experimental results demonstrating 100% *empirical* safety while navigating significantly faster than the baseline planner that relies on formal constraints to remain safe.

2 POMDP Planning

We wish to control a dynamic vehicle through an unknown environment to a goal position in minimum expected time, where the expectation is taken with respect to the (unknown) distribution of maps. While solving this problem exactly is completely intractable, the POMDP formalism is useful for defining the planning task and motivating our approximations. The following POMDP tuple, (S, A, T, R, O, Ω), applies to an RC car equipped with a planar LIDAR being used for this research.

States S: $\{Q \times M\}$. Q is the set of vehicle configurations: $q = [x, y, \psi, k, v]$, representing position, heading, curvature (steering angle) and forward speed, respectively. M is the set of n-cell occupancy maps: $m = [m_0, m_1, \ldots, m_n] \in \{0, 1\}^n$, where m_i represents the i^{th} cell in the map. For a given problem instance, the true underlying map, m, is fixed, while the configuration, q, changes at each time step as the robot moves. We assume that q is fully observable, while m is partially observable since only a subset of the map can be observed from a given location. We also assume that n is fixed and known.

Actions A: A is a pre-computed discrete action library spanning the vehicle's maneuvering capabilities. Several examples are illustrated in Fig. 2. All actions reach a planning horizon of 2 m, but differ in their time duration as a function of their speeds.

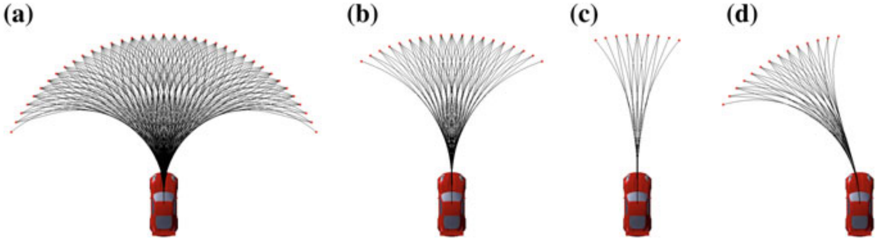


Fig. 2 Examples of pre-computed action sets from 1 m/s (**a**), 4 m/s (**b**), and 8 m/s (**c**) with zero initial curvature, and from 4 m/s with non-zero initial curvature (**d**)

Due to space constraints, we refer the reader to Howard et al. for a discussion of discrete action sets for dynamic vehicles [12].

We define several deterministic functions related to transition dynamics. The collision function $C(s, a) : S \times A \mapsto \{0, 1\}$ indicates whether taking action a from state s would result in a collision. The state-transition function $F(s, a) : S \times A \mapsto S$ returns the state reached by taking action a from state s according to the dynamics. In F, the map portion of the state remains fixed at its true value. If a collision would occur along trajectory a (i.e., if $C(s, a) = 1$), then $F(s, a)$ clamps the configuration to its last feasible value along a and sets the velocity to zero. The function $ICS(s) : S \mapsto \{0, 1\}$ indicates whether state s is an inevitable collision state [3], i.e., if there exists no infinite-horizon sequence of actions $\langle a_0, a_1, \ldots \rangle$ from s that will avoid collision with the environment.

Conditional Transition Probabilities T: We assume deterministic vehicle dynamics and a fixed map, $p(s_{t+1}|s_t, a_t) = 1$ for $s_{t+1} = F(s_t, a_t)$ and 0 otherwise. While actions have different time durations, we use the subscript $t + 1$ to indicate the discrete "time" after completing action a_t.

Observations Ω: Each observation consists of a perfect measurement of q_t and the partial map of cells visible to the robot from s_t. This partial map consists of occupied cells at locations $r_{i,xy}$ corresponding to each LIDAR range measurement r_i, and unoccupied cells along the ray from q_t to each $r_{i,xy}$.

Conditional Observation Probabilities O: We assume a noiseless sensor, so $p(o_{t+1}|s_{t+1}, a_t) = 1$ for the partial map corresponding to the map geometry visible from state s_{t+1}, along with a perfect measurement of q_{t+1}, and 0 otherwise.

Cost Function R: We use a minimum-time cost function and denote the time duration of action a as $J_a(a)$. Let S_G denote the set of goal states. $R(s, a) = 0$ for $s \in S_G$. For $s \notin S_G$, $R(s, a) = J_a(a)$ if $C(s, a) = 0$ and adds an additional collision penalty, J_c, if the action results in a collision: $R(s, a) = J_a(a) + J_c$ if $C(s, a) = 1$.

2.1 Missing Prior Distribution Over Environments

A POMDP agent maintains a belief over its state $b(s_t) = P(q_t, m)$, and has a state estimator, which computes a posterior belief that results from taking an action and then receiving an observation, given a current belief: $b(s_{t+1}) = P(s_{t+1}|b_t, a_t, o_{t+1})$. If the agent's current belief, b_t, assigns uniform probability to all possible maps with independence between map cells (a very unrealistic distribution), then an observation and subsequent inference over the map distribution simply eliminates those maps that are not consistent with the observation and raises the uniform probability of the remaining possible maps. If, on the other hand, the prior belief correctly assigns high probability to a small set of realistic maps with common structures such as hallways, rooms, doors, etc., and low probability to the unrealistic maps, then this belief update may help to infer useful information about unobserved regions of the map. For instance, it might infer that the straight parallel walls of a hallway are likely to continue out into the unobserved regions of the map. All of this prior knowledge about the distribution of environments enters the problem through the agent's initial belief b_0, which we must somehow provide.

Unfortunately, as we noted in Sect. 1, modeling the distribution over real-world environments would be extremely difficult. We have little choice but to initialize the robot believing that all maps are equally likely and that map cells are independent from each other. To compensate for this missing environment distribution, our approach is to learn a much more specific distribution representing the probability of collision associated with a given planning scenario. This learned function enables the agent to drive at high speed *as if* it had an accurate prior over environments enabling reasonable inferences about the unknown. In the following sections, we will derive our learned model from the POMDP and describe how we can efficiently train and use this model in place of an accurate prior over environments.

2.2 Approximations to the POMDP

Let $V_\pi(s) = \sum_{t=0}^{\infty} R(s_t, \pi(s_t))$ be the infinite-horizon cost associated with some policy π mapping states s_t to actions a_t. The optimal cost-to-go, $V^*(s)$ could then, in principle, be computed recursively using the Bellman equation. Having computed $V^*(s)$ for all states, we could then recover the optimal action for a given belief[1]:

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \Bigg\{ \sum_{s_t} b(s_t) R(s_t, a_t) + \sum_{s_t} b(s_t) \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \sum_{o_{t+1}} P(o_{t+1}|s_{t+1}, a_t) V^*(s_{t+1}) \Bigg\} \tag{1}$$

The summations over s_t and o_{t+1} perform a state-estimation update, resulting in a posterior distribution over future states s_{t+1} from all possible current states in our current belief b_t, given our choice of action a_t. We can rewrite (1) as:

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \Bigg\{ \sum_{s_t} b(s_t) R(s_t, a_t) + \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V^*(s_{t+1}) \Bigg\} \tag{2}$$

Since our cost function $R(s, a)$ applies separate penalties for time and collision, we can split $V^*(s)$ into two terms, $V^*(s) = V_T^*(s) + V_C^*(s)$, where T and C refer to "time" and "collision". $V_T^*(s)$, gives the optimal time-to-go from s, regardless of whether a collision occurs between s and the goal. We assume that if a collision occurs, the robot can recover and proceed to the goal. Furthermore, we assume that the optimal trajectory from s to the goal will avoid collision if possible. But there will be some states s for which collision is inevitable ($ICS(s) = 1$) since the robot's

[1] Since we use a discrete set of actions, there is a finite number of configurations that can be reached from an initial configuration. Therefore, we can sum over future states rather than integrating.

abilities to brake or turn are limited to finite, realistic values. Since we assume that collisions only occur from inevitable collision states, we can rewrite the total cost-to-go as: $V^*(s) = V_T^*(s) + J_c \cdot ICS(s)$. Substituting this expression, we can rewrite (2) as:

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \Bigg\{ \sum_{s_t} b(s_t) R(s_t, a_t) + \\ \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V_T^*(s_{t+1}) + J_c \cdot \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) ICS(s_{t+1}) \Bigg\} \tag{3}$$

The first term in Eq. (3) is the expected immediate cost of the current action, a_t. We assume no collisions occur along a_t since the robot performs collision checking with respect to observed obstacles, and it nearly always perceives obstacles in the immediate vicinity. This term simply reduces to $J_a(a_t)$, the time duration of a_t.

The second and third terms of (3) are expected values with respect to the possible future states, given a_t and b_t. However, as we have observed in Sect. 2.1, our initial uniform belief over maps means that b_t will be very unhelpful (and indeed misleading) if we use it to take an expectation over future states, since the true distribution over maps is surely far from uniform. The lack of meaningful prior knowledge over the distribution of environments, combined with the extremely large number of possible future states s_{t+1}, means that we must approximate these terms.

For the expected time-to-go, we use a simple heuristic function:

$$h(b_t, a_t) \approx \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V_T^*(s_{t+1}), \tag{4}$$

which performs a 2D grid search using Dijkstra's algorithm (ignoring dynamics) from the *end* of action a_t to the goal, respecting the observed obstacles in b_t, assuming unknown space is traversable and that the current speed is maintained. The use of a lower-dimensional search to provide global guidance to a local planner is closely related to graduated-density methods [13]. While other heuristics could be devised, we instead focus our efforts in this paper on modeling the collision probability.

The third term in Eq. (3), $J_c \cdot \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) ICS(s_{t+1})$, is the expected future collision penalty given our current belief b_t and action a_t. As we noted in Sect. 2.1, the fundamental problem is that our belief b_t does not accurately capture which map structures are likely to be encountered in the unknown portions of the environment. Correctly predicting whether a hallway will continue around a blind corner, for instance, is impossible based on the belief alone. We instead turn to machine learning to approximate this important quantity from training data:

$$f_c(\phi(b_t, a_t)) \approx \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) ICS(s_{t+1}) \tag{5}$$

The model, $f_c(\phi(b_t, a_t))$, approximates the probability that a collision will occur in the future if we execute action a_t given belief b_t. It is computed from some features $\phi(b_t, a_t)$ that summarize the relevant information contained in b_t and a_t, for example vehicle speed, distance to observed obstacles along a_t, etc. With the approximations of all three terms in Eq. (3), we arrive at our receding-horizon control law:

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}}\left\{J_a(a_t) + h(b_t, a_t) + J_c \cdot f_c(\phi(b_t, a_t))\right\} \tag{6}$$

At each time step, we select a_t^* minimizing (6) given the current belief b_t. We begin to execute a_t^* while incorporating new sensor data and re-planning. Next, we describe how we collect data and build a model to predict collision probabilities.

3 Predicting Future Collisions

In this section, we focus on learning to predict the probability of collision associated with a given planning scenario, represented as a point in feature space, Φ. We assume that for a given vehicle or dynamical system, there exists some true underlying probability of collision that is independent of the map and robot configuration, given features, ϕ. In other words, $P(\text{“collision”}|\phi, q, m) = P(\text{“collision”}|\phi)$. Under this assumption, we can build a data set by training in any environments we wish, and the data will be equally valid in other environments that populate the same regions of feature space. If two data sets gathered from two different environments do not share the same output distribution where they overlap in feature space, we assume that our features are simply not rich enough to capture the difference.

This assumption also implies that if an environment is fundamentally different from our training environments with respect to collision probabilities, it will populate a different region of feature space. If the robot encounters an environment for which it has no relevant training data nearby in feature space, it should infer that this environment is unfamiliar and react appropriately. In these cases, we require that our learning algorithm somehow provide a safe prediction of collision probability rather than naïvely extrapolating the data from other environments. Our features must therefore be sufficiently descriptive to predict collisions as well as differentiate between qualitatively different environment types.

Quantifying a planning scenario using a set of descriptive functions of belief-action pairs currently relies on the domain knowledge of the feature designer. For this paper, our features are four hand-coded functions: (a) minimum distance to the nearest known obstacle along the action; (b) mean range to obstacle or frontier in the 60° cone ahead of the robot, averaged over several points along the action; (c) length of the straight free path directly ahead of the robot, averaged over several points along the action; and (d) speed at the end of the action. While these features work adequately, our method is extensible to arbitrary numbers of continuous- and discrete-valued features from a variety of different sources, including features that operate on a history of past measurements. We expect that additional features will enable more intelligent and subtle navigation behaviors.

3.1 Training Procedure

To predict collision probabilities, we collect training data $D = \{(\phi_1, y_1), \ldots, (\phi_N, y_N)\}$. Labels y_i are binary indicators ("collision", "non-collision") associated with belief-action pairs, represented as points ϕ_i in feature space. To efficiently collect a large data set, we use a simulator capable of generating realistic LIDAR scans and vehicle motions within arbitrary maps. The training procedure aims to generate scenarios that accurately emulate beliefs the planner may have at runtime, and accurately represent the risk of actions given those beliefs. While at runtime, the planner will use a map built from a history of scans, we make the simplifying assumption that a single measurement taken from configuration q_t is enough to build a realistic map of the area around q_t, similar to one the planner might actually observe. With this assumption, we can generate training data based on individual sampled robot configurations, rather than sampling extended state-measurement histories, which not only results in more efficient training, but also eliminates the need for any sort of planner or random process to sample state histories.

We generate each data point by randomly sampling a feasible configuration, q_t, within a training map, and simulating the sensor from q_t to generate a map estimate, and hence a belief b_t. We then randomly select one of the actions, a_t, that is feasible given the known obstacles in b_t. Recall from Eq. (5) that our learned function models the probability that a collision will occur somewhere *after* completing the immediate chosen action a_t, i.e., the probability that state s_{t+1} (with configuration q_{t+1}) is an inevitable collision state. Therefore, to label this belief-action pair, we run a resolution-complete training planner from configuration q_{t+1} at the *end* of a_t. If there exists any feasible trajectory starting from q_{t+1} that avoids collision with the true hidden map (to some horizon), then $y_{\text{new}} = 0$, otherwise $y_{\text{new}} = 1$. Finally, we compute features $\phi_{\text{new}}(b_t, a_t)$ of this belief-action pair and insert $(\phi_{\text{new}}, y_{\text{new}})$ into D.

We use a horizon of three actions (6 m) for our training planner because if a collision is inevitable for our dynamics model, it will nearly always occur within this horizon. We have explored other settings for this horizon and found the results to be comparable, although if the horizon is too short, some inevitable collision states will be mis-labeled as "non-collision".

Figure 3 illustrates "collision" and "non-collision" training instances. In Fig. 3a, the hidden map includes a sharp hairpin turn, which is infeasible for our dynamics model, given that the car begins toward the inside of the turn at a relatively high speed (≈8 m/s). Therefore, the training planner fails to find a trajectory to the desired horizon and each partial path results in a dead-end (red dot) because the robot is moving too fast to complete the hairpin turn. On the other hand, the hidden map in Fig. 3b has a straight hallway rather than a sharp turn, and the training planner succeeds in finding a feasible trajectory to the three-action horizon, aided by the car's initial con-

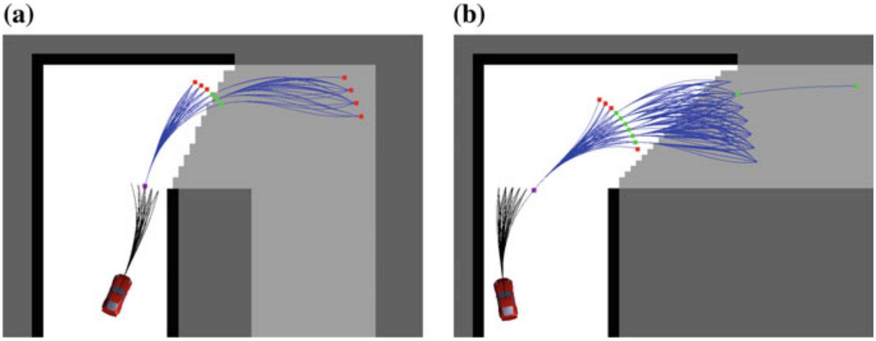


Fig. 3 Examples of "collision" (**a**) and "non-collision" (**b**) training events. One of the immediate actions (*black*) is chosen for labeling. The training planner determines whether the end of this action (*purple dot*) is an inevitable collision state with respect to the hidden map (shown in *light gray*). Feasible partial paths are shown in *blue*. Nodes successfully expanded by the training planner are *green*, and nodes for which no collision-free outgoing action exists are *red*. In **a**, all partial paths dead-end (*red nodes*) before reaching the desired three-action horizon because the vehicle speed is too great to complete the turn given curvature and curvature rate limits. In **b**, the training planner successfully finds a sequence of three actions

figuration toward the outside of the turn. The empirical distribution of "collision" and "non-collision" labels collected from these sampled scenarios therefore implicitly captures the relevant environmental characteristics of the true hidden map distribution, and the way they interact with our dynamics model, in a much more efficient way than modeling the complex, high-dimensional map distribution explicitly. Our training procedure captures sensible patterns, for instance that it is safe to drive at high speeds in wide open areas and long straight hallways, but that slower speeds are safer when approaching a wall or navigating dense clutter.

3.2 Learning Algorithm

We use a non-parametric Bayesian inference model developed by Vega-Brown et al., which generalizes local kernel estimation to the context of Bayesian inference for exponential family distributions [29]. The Bayesian nature of this model will enable us to provide prior knowledge to make safe (though perhaps conservative) predictions in environments where we have no training data. We model collision as a Bernoulli-distributed random event with beta-distributed parameter $\theta \sim \text{Beta}(\alpha, \beta)$, where α and β are prior pseudo-counts of collision and non-collision events, respectively. Using the inference model from Vega-Brown et al., we can efficiently compute the posterior probability of collision given a query point ϕ and data D:

$$f_c(\phi) = P(y = \text{“collision”}|\phi, D) = \frac{\alpha(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i) y_i}{\alpha(\phi) + \beta(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i)}, \qquad (7)$$

where $k(\phi, \phi_i)$ is a kernel function measuring proximity in feature space between our query, ϕ, and each ϕ_i in D. We use a polynomial approximation to a Gaussian kernel with finite support. We write prior pseudo-counts as functions $\alpha(\phi)$ and $\beta(\phi)$, since they may vary across the feature space. We use the effective number of nearby training data points, $N_{\text{eff.}} = \sum_{i=1}^{N} k(\phi, \phi_i)$, as a measure of data density. The prior contributes $N_{\text{pr.}}$ pseudo data points to each prediction, where $N_{\text{pr.}} = \alpha(\phi) + \beta(\phi)$. The ratio $N_{\text{eff.}}/(N_{\text{eff.}} + N_{\text{pr.}})$ determines the relative influence of the training data and the prior in each prediction. For data sets with 5×10^4 points in total, $N_{\text{eff.}}$ tends to be 10^2 or greater when the testing environment is similar to the training map, and $N_{\text{eff.}} \ll 1$ when the testing environment is fundamentally different (i.e., office building vs. natural forest). For this paper, we used $N_{\text{pr.}} = 5$, and results are insensitive to the exact value of $N_{\text{pr.}}$ as long as $N_{\text{eff.}} \gg N_{\text{pr.}}$ or $N_{\text{eff.}} \ll N_{\text{pr.}}$.

Machine learning algorithms should make good predictions for query points near their training data in input space. However, predictions that extrapolate beyond the domain of the training data may be arbitrarily bad. For navigation tasks, we want our planner to recognize when it has no relevant training data, and automatically revert to safe behaviors rather than making reckless, uninformed predictions. For instance, if the agent moves from a well-known building into the outdoors, where it has not trained, we still want the learning algorithm to guide it away from danger.

Fortunately, the inference model of Eq. (7) enables this capability through the prior functions $\alpha(\phi)$ and $\beta(\phi)$. If we query a feature point in a region of high data density ($N_{\text{eff.}} \gg N_{\text{pr.}}$), $f_c(\phi)$ will tend to a local weighted average of neighbors and the prior will have little effect. However, if we query a point far from the training data ($N_{\text{eff.}} \ll N_{\text{pr.}}$), the prior will dominate. By specifying priors $\alpha(\phi)$ and $\beta(\phi)$ that are functions of our features, we can endow the planner with domain knowledge and formal rules about which regions of feature space are safe and dangerous. In this paper, we have designed our prior functions $\alpha(\phi)$ and $\beta(\phi)$ such that $P(\text{“collision”}) = \alpha(\phi)/(\alpha(\phi) + \beta(\phi)) = 0$ if there exists enough free space for the robot to come safely to a stop from its current velocity, and $P(\text{“collision”}) = \alpha(\phi)/(\alpha(\phi) + \beta(\phi)) > 0$ otherwise. Computing this prior uses the information in features (c) and (d) described in Sect. 3. Therefore, as $N_{\text{eff.}}$ drops to zero (and for sufficiently large values of J_c), the learning algorithm activates a conventional stopping distance constraint, seamlessly turning our planner into one with essentially the same characteristics as the baseline planner we compare against.

Another natural choice of learning algorithm in this domain would be a Gaussian process (GP). However, classification with a GP requires approximate inference techniques that would likely be too slow to run online, whereas the inference model in Eq. (7) is an approximate model that allows efficient, analytical inference.

4 Results

We have obtained simulation results from randomly generated maps as well as experimental results on a real RC car navigating as fast as 8 m/s in laboratory, office and large open atrium environments at MIT. Our simulation and experimental results both show that replacing safety constraints with a learned model of collision probability can result in faster navigation without sacrificing empirical safety. The results also show that when the planner departs a region of feature space for which it has training data, our prior keeps the robot safe. Finally, our experiments demonstrate that even within a single real-world environment, it is easy to encounter some regions of feature space for which we have training data and others for which we do not, thereby justifying our Bayesian approach and use of prior knowledge.

4.1 Simulation Results

In this section, we present simulation results from hallway and forest environments that were randomly sampled from environment-generating distributions [21]. We use a Markov chain to sample hallways with a width of 2.5 m and a turn frequency of 0.4, and a Poisson process to sample 2D forests with an average obstacle rate of 0.05 trees $\cdot$ m^{-2} and tree radius of 1 m. Figure 5 shows a randomly sampled hybrid environment composed of both hallway and forest segments. To measure the benefit of our learned model, we compare against the baseline planner, illustrated in Fig. 1a and b, that enforces a safety constraint. This planner navigates as fast as is possible given that constraint, and represents the planner we would use if we did not have a learned model of collision probability.

Figure 4 shows the performance of the planner using different data/prior configurations (data + prior, data only, and prior only), for different J_c values, in 25 randomly sampled hallway environments. The training data set used for these trials was col-

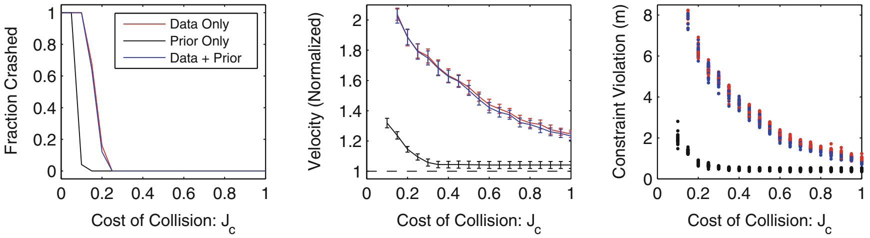


Fig. 4 Simulation results from 1600 simulations in 25 random hallway maps using our learned model with dense data + prior (*blue*), data alone without a prior (*red*), and prior alone without data (*black*). Velocities for each trial are normalized by the speed of the baseline planner (described in Sect. 1) in the same map. The *right plot* shows the average length by which a stopping-distance constraint was violated

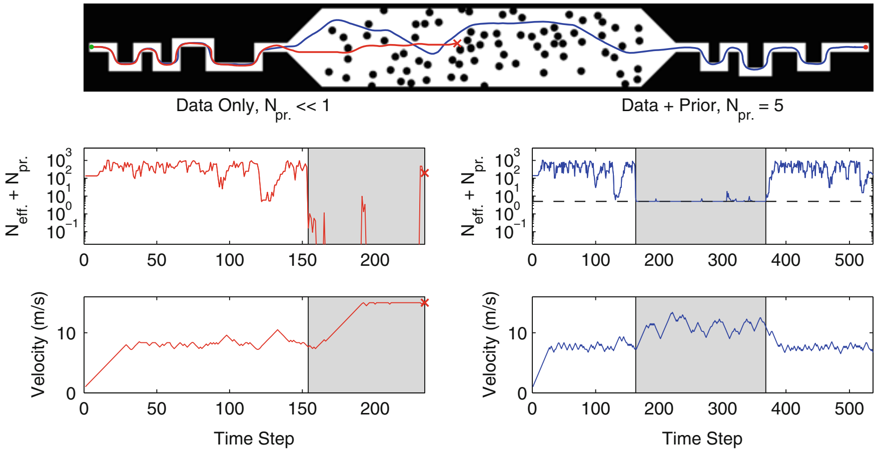


Fig. 5 Simulations from a hybrid hallway-forest map with $J_c = 0.5$. One planner uses hallway data alone with no prior (*red*), and another planner uses hallway data combined with the prior (*blue*). Without a safe prior, the planner's data density drops to zero and the robot recklessly accelerates to full speed, crashing in the forest (*red* '×'). However, when guided by the prior while in the forest, the planner safely reaches the goal. The forest regions on the four graphs are shaded in *gray*

lected in a separate hallway environment drawn from the same distribution. For low values of J_c all three planners crash in every trial, but become safer as J_c is increased. Above $J_c = 0.25$, all planners succeed in reaching the goal without collision 100% of the time. At $J_c = 0.25$, our solution navigates approximately 80% faster than baseline using data + prior, or data alone. In these trials, the prior contributes very little since the planner has dense training data from the same environment type. The right plot illustrates average violation of a stopping-distance constraint. For $J_c = 0.25$, the planners using training data violate the stopping-distance constraint by 5.75 m on average, indicating the degree to which this constraint is overly conservative, given our collision probability model. The planner using the prior alone chooses not to violate the stopping distance constraint by nearly as much, and essentially converges to the baseline performance as J_c is increased.

While results with dense training data make very little use of the prior, Fig. 5 illustrates the important role of a prior when transitioning from a familiar environment type (hallway) to an unfamiliar one (forest) where no data are available. For these simulation experiments, the planners had access to a training data set from a hallway environment, but not from a forest environment. If the planner has no data *and no prior*, the effective data density drops essentially to zero and it is unable to distinguish between safe and risky behaviors.[2] Therefore the planner accelerates to full speed resulting in a crash (red '×'). However, using our Bayesian approach, the effective

[2] We implement the no-prior case by setting prior values of α and β each to 0.0005 to ensure the solution is computable in regions of feature space with no data at all. The default prediction with no data is therefore $P(\text{“collision”}) = \alpha/(\alpha + \beta) = 0.5$, rather than undefined if $\alpha = \beta = 0$.

number of data points drops only as low as $N_{\text{pr.}} = 5$ when the agent enters the forest and the planner is safely guided by the information in the prior. In 25 trials of random hallway-forest maps, 100% succeeded using the prior, while only 12% succeeded without the prior.

4.2 Experimental Results

We conducted experiments on an RC car using a training dataset generated in simulation on a hallway-type environment. We performed state estimation by fusing a planar LIDAR and an IMU in an extended Kalman filter (EKF) [7]. We use the LIDAR to provide relative pose updates to the EKF using a scan-matching algorithm [4]. We use a Hokuyo UTM-30LX LIDAR, a Microstrain 3DM-GX3-25 IMU and an Intel dual-core i7 computer with 16GB RAM. Video of our experimental demonstrations, at speeds up to 8.2 m/s is available at: http://groups.csail.mit.edu/rrg/bayesian_learning_high_speed_nav.

We conducted tests to show that our planner can indeed navigate faster in certain real environments than the baseline planner. For the experiment shown here, we chose a narrow path within a lab space with several sharp turns and many obstacles. Figure 6 shows the trajectories and velocity profiles of both planners. The baseline planner has difficulty navigating quickly in this environment because free space is occluded from the sensor view by obstacles. Since the baseline planner must enforce the existence of emergency-stopping trajectories lying within the observed free space, it is forced to move very slowly. In the velocity profile, the baseline planner frequently applies the brakes to slow to about 1 m/s, whereas our planner uses its training data from the simulated hallway environment to predict that it is safe to travel up to about 3 m/s around most corners and therefore maintains a higher average speed. The baseline planner took 23.7 s to reach the goal in this case, whereas our planner took 11.5 s, representing a factor of two improvement.

We also conducted experimental trials to show that in real-world environments, the planner can safely navigate in unfamiliar regions where it has no relevant training data. Figure 7 shows an experimental trial in the Stata Center (MIT), which is

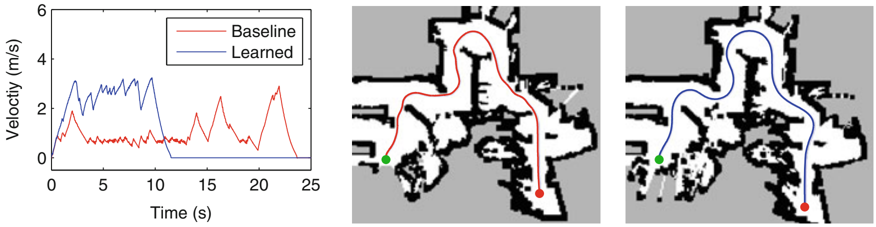


Fig. 6 Experiment in which our planner (*blue*) reached its goal over 2x faster than baseline (*red*). Velocity profiles and trajectories for each planner are shown

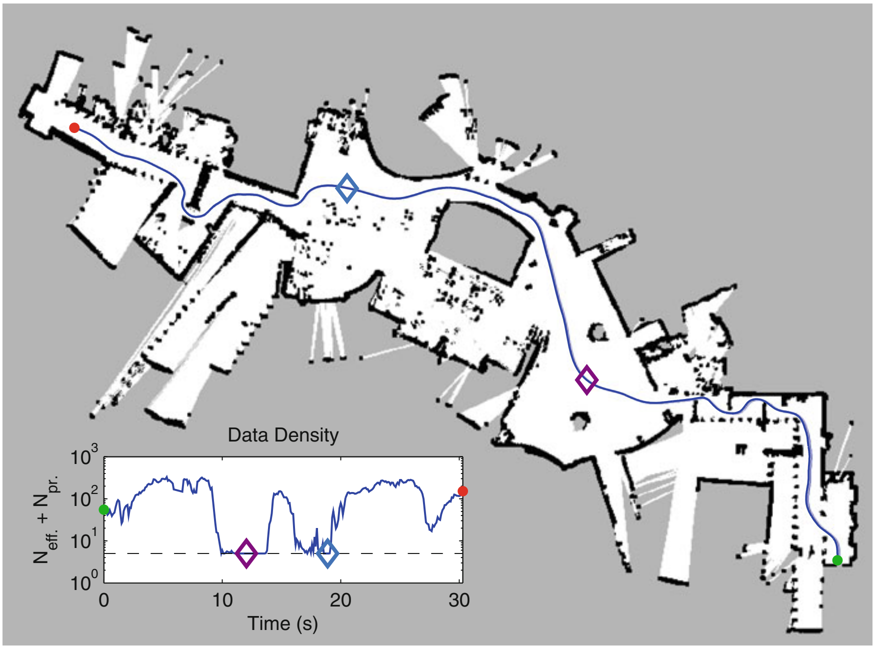


Fig. 7 Trajectory through the Stata Center (MIT), traversing hallway and large unstructured regions. Our planner was trained in a hallway environment, but not in a large unstructured environment. $N_{\text{eff.}}$ drops to zero in the open unstructured regions (*purple* and *blue diamonds*) and it must rely on the prior to navigate safely. The *top speed* in this environment was 8.2 m/s in a separate successful trial

composed of straight hallways and larger unstructured regions. For this experiment, training data were again provided from a simulated hallway environment, which resembles the straight hallway segments, but looks very different from the open, unstructured regions. The inset graph shows the effective data density. In the hallways, the planner uses $\approx 10^2$ effective data points per prediction. However, in the open unstructured regions, $N_{\text{eff.}}$ drops to zero, and the only information available to the planner is contributed by the prior. These open unstructured regions are marked with purple and blue diamonds. In this experiment, the weight of the prior was $N_{\text{pr.}} = 5$.

5 Related Work

A large body of work has established techniques for safe planning in static and dynamic environments [3, 8, 9, 25]. Bekris and Kavraki have shown kinodynamic navigation in unknown environments using safety constraints, without considering actions into the unknown [6]. Several examples use circling loiter maneuvers through observed free space to guarantee safety [2, 23]. Our method differs from this body of work by relaxing absolute safety constraints and replacing them with predictions of collision probability, which can be viewed as a form of data-driven constraints. Althoff et al. propose a collision probability concept similar to $f_c(\phi(b_t, a_t))$ for dynamic environments, but they assume a known distribution over the future behavior of each moving agent, which we do not have in the case of unknown maps [1].

In the exploration literature, the primary objective is to build a map, and actions may be taken to view unseen parts of the environment [28, 30]. Some exploration work has balanced the objectives of information gain, navigation cost and localization quality from a utility or decision-theory point of view [16, 26]. Unlike exploration, our objective is to reach a goal in minimum time, which places emphasis on collision probability and high-speed dynamics, rather than map information.

A natural way to reason about planning in an unknown map is through the POMDP formalism [14]. Despite notorious complexity, various strategies have grown the size of (approximately) solvable POMDPs [15, 19, 24]. However, we cannot simply apply POMDP techniques since we assume no explicit knowledge of the environment distribution, or black-box simulation capabilities to sample the world at planning time. Instead, we must learn a function of this missing distribution offline. POMDPs have been used for aircraft collision avoidance [5, 27], where a very large negative reward discourages collisions at nearly any cost. In contrast, we use small collision penalties to drive aggressively, trading off risk and reward.

Learning has been applied to autonomous vehicle navigation in several contexts. One example from the DARPA LAGR program used deep learning to classify traversable terrain up to 100 m away [11]. Neural networks and monocular depth estimation coupled with policy search [17, 18], as well as human-pilot demonstrations [22], have been used to learn high-speed reactive controllers to avoid obstacles, however these systems map sensory input directly to actions. They are not decision-theoretic planners, and do not trade off meaningfully between risk and reward.

6 Conclusion

We have shown that by using a Bayesian learning algorithm, with safety constraints encoded as a prior over collision probabilities, our planner can detect when it lacks the appropriate training data for its environment and seamlessly revert to safe behaviors. Our strategy offers a simple and elegant probabilistic method of merging the performance benefits of training experience with the the reassurance of safety constraints when faced with an unfamiliar environment. One of the main limitations of this work is the difficulty of hand-coding feature functions that describe complex planning scenarios. We plan to address this limitation by applying recent feature-learning techniques to automatically generate feature functions from data, extending our methods to handle different scenarios, map representations and sensor types.

References

1. Althoff, D., et al.: Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. Auton. Robot. **32**(3), 285–302 (2012)
2. Arora, S., et al.: A principled approach to enable safe and high performance maneuvers for autonomous rotorcraft. In: American Helicopter Society 70th Annual Forum (2014)
3. Asama, H., Fraichard, T.: Inevitable collision states - a step towards safer robots. Adv. Robot. **18**, 1001–1024 (2004)
4. Bachrach, A., et al.: RANGE - robust autonomous navigation in GPS-denied environments. J. Field Robot. **28**(5), 644–666 (2011)
5. Bai, H., et al.: Unmanned aircraft collision avoidance using continuous-state POMDPs. In: Proceedings of the Robotics: Science and Systems (2011)
6. Bekris, K.E., Kavraki, L.E.: Greedy but safe replanning under kinodynamic constraints. In: Proceedings of the ICRA (2007)
7. Bry, A., et al.: State estimation for aggressive flight in GPS-denied environments using onboard sensing. In: Proceedings of the ICRA (2012)
8. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. **17**(7), 760–772 (1998)
9. Fox, D., et al.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **4**(1), 23–33 (1997)
10. Fraichard, T.: A short paper about motion safety. In: Proceedings of the ICRA (2007)
11. Hadsell, R., et al.: Learning long-range vision for autonomous off-road driving. J. Field Robot. **26**(2), 120–144 (2009)
12. Howard, T.M., et al.: State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. J. Field Robot. **25**(6–7), 325–345 (2008)
13. Howard, T.M., et al.: Model-predictive motion planning: several key developments for autonomous mobile robots. IEEE Robot. Autom. Mag. **21**(1), 64–73 (2014)
14. Kaelbling, L.P., et al.: Planning and acting in partially observable stochastic domains. Artif. intell. **101**(1), 99–134 (1998)
15. Kurniawati, H., et al.: SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proceedings of the Robotics: Science and Systems (2008)
16. Makarenko, A.A., et al.: An experiment in integrated exploration. In: Proceedings of the IROS (2002)
17. Michels, J., et al.: High speed obstacle avoidance using monocular vision and reinforcement learning. In: Proceedings of the ICML (2005)
18. Muller, U., et al.: Off-road obstacle avoidance through end-to-end learning. In: Proceedings of the NIPS (2005)
19. Pineau, J., et al.: Point-based value iteration: an anytime algorithm for POMDPs. In: Proceedings of the IJCAI (2003)
20. Richter, C., et al.: High-speed autonomous navigation of unknown environments using learned probabilities of collision. In: Proceedings of the ICRA (2014)
21. Richter, C., et al.: Markov chain hallway and Poisson forest environment generating distributions. Technical report MIT-CSAIL-TR-2015-014 (2015)
22. Ross, S., et al.: Learning monocular reactive UAV control in cluttered natural environments. In: Proceedings of the ICRA (2013)
23. Schouwenaars, T., et al.: Receding horizon path planning with implicit safety guarantees. In: Proceedings of the ACC (2004)
24. Silver, D., Veness, J.: Monte-carlo planning in large POMDPs. In: Proceedings of the NIPS (2010)
25. Simmons, R.: The curvature-velocity method for local obstacle avoidance. In: Proceedings of the ICRA (1996)
26. Stachniss, C., et al.: Information gain-based exploration using Rao-Blackwellized particle filters. In: Proceedings of the Robotics: Science and Systems (2005)

27. Temizer, S., et al.: Collision avoidance for unmanned aircraft using Markov decision processes. In: AIAA Guidance, Navigation, and Control Conference (2010)
28. Thrun, S., et al.: Map learning and high-speed navigation in RHINO. AI-Based Mobile Robots: Case Studies of Successful Robot Systems. MIT Press, Cambridge (1998)
29. Vega-Brown, W., et al.: Nonparametric Bayesian inference on multivariate exponential families. In: Proceedings of the NIPS (2014)
30. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proceedings of the Computational Intelligence in Robotics and Automation (1997)

Monte Carlo Motion Planning for Robot Trajectory Optimization Under Uncertainty

Lucas Janson, Edward Schmerling and Marco Pavone

1 Introduction

Robotic motion planning is the problem of computing a path that connects an initial and a terminal robot state while avoiding collisions with obstacles and optimizing an objective function [11]. Despite the fact that finding a feasible, let alone optimal, solution to a motion planning problem is difficult (even the most basic versions of the problem are already PSPACE-hard [11, 23]), in the past three decades key breakthroughs have made the solution to this problem largely practical (see [11] and references therein for a comprehensive historical account). Most works in the literature, however, focus on a deterministic setup where the state of a robot is perfectly known and its actions lead to a deterministic, unique outcome. While this is usually an excellent approximation for robots operating in highly structured environments (e.g., manipulators in an assembly line), it falls short in unstructured settings, e.g., for ground or aerial field robots or surgical robotic systems [12]. In such cases, motion uncertainty, sensing uncertainty, and environment uncertainty may dramatically alter

Lucas Janson and Edward Schmerling contributed equally to this work. This work was supported by NASA under the Space Technology Research Grants Program, Grant NNX12AQ43G. Lucas Janson was partially supported by NIH training grant T32GM096982.

L. Janson (✉)
Department of Statistics, Stanford University, Stanford, CA 94305, USA
e-mail: ljanson@stanford.edu

E. Schmerling
Institute for Computational and Mathematical Engineering, Stanford University,
Stanford, CA 94305, USA
e-mail: schmrlng@stanford.edu

M. Pavone
Department of Aeronautics and Astronautics, Stanford University, Stanford,
CA 94305, USA
e-mail: pavone@stanford.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_20

the safety and quality of a path computed via deterministic techniques (i.e., neglecting uncertainty). Hence, accounting for uncertainty in the planning process is regarded as an essential step for the deployment of robotic systems “outside the factory floor” [12]. In this paper we introduce Monte Carlo Motion Planning, a novel approach to planning under uncertainty that is accurate, fast, and general.

Related work: Conceptually, to enable a robot to plan its motion under uncertainty, one needs to design a strategy for a decision maker. In this regard, robotic motion planning can be formalized as a partially observable Markov decision process (POMDP) [8], where the key idea is to assume that the state evolves according to a controlled Markov chain, the state is only partially observed, and one seeks to design a *control policy* that maps state probability distributions to actions. However, despite the theoretical [8] and practical [10] successes of the POMDP theory, the online computation of a control policy for robotic applications is extremely computationally intensive, and possibly even unnecessary as after a short time horizon the environment map may have changed [11]. The alternative and widely adopted approach is then to restrict the optimization process to *open-loop* trajectories, which involves the much simpler task of computing a *control sequence* (as opposed to a control policy), and recompute the reference trajectory in a receding horizon fashion (e.g., every few seconds). This is the approach we consider in this paper.

To select open-loop trajectories, a large number of works cast the problem into a chance-constrained optimization problem [5], where under the assumption of linear dynamics and convex obstacles, an open-loop control sequence is computed as the solution to a mixed-integer linear program. The works in [20, 28] extend this approach to an optimization over the larger class of affine output feedback controllers, comprising a nominal control input and an error feedback term. These works, however, require an explicit characterization of the obstacle space (in the configuration space), which is oftentimes unavailable [11, Chap. 5]. This has prompted a number of researchers to extend the sampling-based motion planning paradigm to the problem of planning under uncertainty (in the sampling-based paradigm, an *explicit* construction of the configuration space is avoided and the configuration space is probabilistically “probed” with a sampling scheme [11]). A common approach is to forgo path optimization and recast the problem as an *unconstrained* planning problem where the path collision probability (CP) is minimized. For example, the approach of LQG-MP [3] is to approximate a path CP by combining pointwise CPs as if they were independent, running the rapidly-exploring random trees (RRT) [13] algorithm multiple times, and then selecting the path with minimum (approximate) path CP. The pointwise CPs are computed within the model that a reference tracking controller is employed to track a nominal open-loop path. This is closely related to model predictive control (MPC) with closed-loop prediction [18] and leads to a less conservative collision probability estimate than if the nominal control was executed without feedback. A similar approach is used in [25], where the authors employ a truncation method [22] to improve the accuracy of path CP computation.

The interplay between minding collision probability while simultaneously optimizing a path planning cost objective function is considered in [26], although still with an approximation to the path CP. There, cost optimization is considered over the set of path plans satisfying a lower bound on success probability. The inclusion of path CP as a constraint is also considered in [15], where the authors propose CC-RRT, an RRT-based algorithm that approximates path CP via Boole's bound. CC-RRT has been extended to include dynamic obstacles via Bayesian nonparametric models [2], tailored to the control of unmanned aerial vehicles [9] and parafoils [17], and combined with RRT*, the asymptotically optimal version of RRT [16].

Contributions: In this paper we present an algorithm for robot planning under uncertainty that returns high quality solutions (in terms of a general planning objective) which satisfy a specified constraint on collision probability, or safety tolerance. The motivation of this work is that all of the aforementioned approaches approximate path CP in ways that can be quite inaccurate (see Fig. 2), thus potentially drastically mischaracterizing the feasible domain of path optimization. To address this problem, our first contribution is to design a variance-reduced (that is, quickly-converging) Monte Carlo (MC) probability estimation algorithm for CP computation. This algorithm estimates the collision probability of a given trajectory by sampling many realizations of a reference-tracking controller, modeling the effort of a robot to follow a reference path. In particular, in this paper, we assume a linear-quadratic-Gaussian (LQG) tracking controller, similar to LQG-MP [3] and MPC with closed-loop prediction [18]. Our algorithm does not suffer the inaccuracies of the approximations mentioned earlier, and indeed provides the exact path CP given enough time (in contrast to current approaches). Most importantly, our variance-reduction scheme, which combines and tailors control variate and importance sampling techniques in an original fashion to the problem at hand, enables the computation of very accurate estimates in a way compatible with real-time operations. This holds even when working with very small CPs, a regime in which a straightforward Monte Carlo method would require great computational expense to arrive at accurate estimates, to the point that MC has hitherto seen application only in offline contexts, e.g., [1]. Another key advantage of our algorithm is that it comes with an estimate of its variance, so that one has a measure of accuracy, unlike the aforementioned approximations. It is also trivially parallelizable and has the potential to be extended to very general controllers and uncertainty models.

Our estimation algorithm enables a novel approach to planning under uncertainty, which we call Monte Carlo Motion Planning (MCMP)—our second contribution. MCMP proceeds by performing bisection search over CP and obstacle inflation, at each step solving a deterministic version of the problem with inflated obstacles. To demonstrate the performance of MCMP, we present simulation results that illustrate the correctness (in terms of feasibility), efficiency (in terms of path cost), and computational speed of MCMP. From a conceptual standpoint, MCMP can be viewed as a planning analogue to MC approaches for robot localization [27].

Organization: This paper is structured as follows. Section 2 reviews some background on MC variance reduction. Section 3 formally defines the problem we consider in this paper. Section 4 elucidates the shortcomings of previous path CP approximation schemes. Section 5 presents variance-reduction techniques for fast MC computation of path CP. Section 6 presents the overall MCMP approach.

Section 7 presents results from numerical experiments supporting our statements. Finally, in Sect. 8, we draw some conclusions and discuss directions for future work.

2 Background on Monte Carlo Variance Reduction

The use of Monte Carlo (MC) to estimate the probability of complex events is well-studied. In this section we will briefly introduce MC and the two variance reduction techniques that provide the basis for our main result in Sect. 5. For more detail and other topics on Monte Carlo, the reader is referred to the excellent unpublished text [21], from which the material of this section is taken.

2.1 Simple Monte Carlo

In its most general form, MC is a way of estimating the expectation of a function of a random variable by drawing many independent and identically distributed (i.i.d.) samples of that random variable, and averaging their function values. Explicitly, consider a random variable $\boldsymbol{X} \in \mathbb{R}^n$ and a bounded function $f : \mathbb{R}^n \to \mathbb{R}$. For a sequence of m i.i.d. realizations of $\boldsymbol{X}$, $\{\boldsymbol{X}^{(i)}\}_{i=1}^m$, the central limit theorem gives

$$\sqrt{m}\left(\frac{1}{m}\sum_{i=1}^{m} f\left(\boldsymbol{X}^{(i)}\right) - \mathbb{E}\left[f\left(\boldsymbol{X}\right)\right]\right) \xrightarrow{\mathscr{D}} \mathscr{N}\left(0, \tau^2\right), \tag{1}$$

as $m \to \infty$, where $\xrightarrow{\mathscr{D}}$ denotes convergence in distribution, and $\mathscr{N}(0, \tau^2)$ refers to the Gaussian distribution with mean 0 and variance τ^2. This implies $\frac{1}{m}\sum_{i=1}^{m} f\left(\boldsymbol{X}^{(i)}\right) \xrightarrow{p} \mathbb{E}\left[f\left(\boldsymbol{X}\right)\right]$ as $m \to \infty$, where $\xrightarrow{p}$ denotes convergence in probability.

In this paper, $\boldsymbol{X}$ will be a random trajectory controlled to follow a nominal path, and f will be the indicator function that a trajectory collides with an obstacle; call this collision event A. Therefore, the expectation in Eq. (1) is just $\mathbb{E}[f(\boldsymbol{X})] = \mathbb{P}(A)$. Denote this collision probability by p, and define $\hat{p}_{\text{simple}} := \frac{1}{m}\sum_{i=1}^{m} f\left(\boldsymbol{X}^{(i)}\right)$. Then τ^2 can be consistently estimated by the sample variance of the $f(\boldsymbol{X}^{(i)})$, i.e.,

$$\hat{\tau}^2 := \frac{1}{m}\sum_{i=1}^{m}\left(f\left(\boldsymbol{X}^{(i)}\right) - \hat{p}_{\text{simple}}\right)^2 \xrightarrow{p} \tau^2, \tag{2}$$

as $m \to \infty$. $\hat{V}_{\text{simple}} := \hat{\tau}^2/m$ allows us to quantify the uncertainty in the CP estimator $\hat{p}_{\text{simple}}$ by approximating its variance. Note that the MC estimator is both unbiased and consistent for its target of $\mathbb{E}[f(\boldsymbol{X})]$. The material from this subsection can be found with more detail in [21, Chap. 2].

2.2 Control Variates

To reduce the variance of $\hat{p}_{\text{simple}}$, we can use the method of control variates (CV). CV requires a function $h : \mathbb{R}^n \to \mathbb{R}$ such that $\theta := \mathbb{E}[h(\boldsymbol{X})]$ is known. Then if $h(\boldsymbol{X}^{(i)})$ is correlated with $f(\boldsymbol{X}^{(i)})$, its variation around its (known) mean can be used to characterize the variation of $f(\boldsymbol{X}^{(i)})$ around its (unknown) mean, which can then be subtracted off from $\hat{p}_{\text{simple}}$. Explicitly, given a scaling parameter value β, we estimate $p = \mathbb{E}[f(\boldsymbol{X})]$ by

$$\hat{p}_\beta := \frac{1}{m}\sum_{i=1}^{m}(f(\boldsymbol{X}^{(i)}) - \beta h(\boldsymbol{X}^{(i)})) + \beta\theta = \hat{p}_{\text{simple}} - \beta(\hat{\theta} - \theta), \tag{3}$$

where $\hat{\theta}$ is the sample average of the $h(\boldsymbol{X}^{(i)})$. The optimal (variance-minimizing) choice of β can be estimated from the simulated data as

$$\hat{\beta} := \sum_{i=1}^{m}(f(\boldsymbol{X}^{(i)}) - \hat{p}_{\text{simple}})(h(\boldsymbol{X}^{(i)}) - \hat{\theta})\Big/\sum_{i=1}^{m}(h(\boldsymbol{X}^{(i)}) - \hat{\theta})^2. \tag{4}$$

We then use the CP estimator $\hat{p}_{\hat{\beta}}$, whose variance can be estimated by

$$\hat{V}_{\hat{\beta}} := \frac{1}{m^2}\sum_{i=1}^{m}(f(\boldsymbol{X}^{(i)}) - \hat{p}_{\hat{\beta}} - \hat{\beta}(h(\boldsymbol{X}^{(i)}) - \hat{\theta}))^2. \tag{5}$$

The data-dependent choice of β introduces a bias in $\hat{p}_{\hat{\beta}}$ that is asymptotically (in m) negligible compared to its variance, so we will ignore it here. As $m \to \infty$, the variance reduction due to CV can be characterized by $\text{Var}(\hat{p}_{\hat{\beta}})/\text{Var}(\hat{p}_{\text{simple}}) \to 1 - \rho^2$, where ρ is the correlation between $f(\boldsymbol{X})$ and $h(\boldsymbol{X})$. The material from this subsection can be found with more detail in [21, Sect. 8.9].

2.3 Importance Sampling

Another tool for MC variance reduction is importance sampling (IS). IS becomes of critical importance when $f(\boldsymbol{X})$ is the indicator function for a rare event, as it is in this paper (in most settings, path CP constraints are small to ensure a high likelihood of safety). Specifically, when $p \ll 1$, we can approximate the coefficient of variation (ratio of standard deviation to expected value) of the estimator $\hat{p}_{\text{simple}}$ as,

$$\frac{\sqrt{\text{Var}(\hat{p}_{\text{simple}})}}{\mathbb{E}[\hat{p}_{\text{simple}}]} = \frac{\sqrt{p(1-p)/m}}{p} \approx \frac{1}{\sqrt{m}}\frac{1}{\sqrt{p}}. \tag{6}$$

This means that in order to get the *relative* uncertainty in $\hat{p}_{\text{simple}}$ to be small, one needs $m \gg 1/p$ which can be very large, and this is simply due to the rarity of observing the event A. IS allows us to sample $\boldsymbol{X}$ from a distribution that makes the event A more common and still get an unbiased estimate of p.

Until now we have considered $\boldsymbol{X}$ to have some fixed probability density function (pdf) $P : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$. Denoting the expectation of $f(\boldsymbol{X})$ when $\boldsymbol{X}$ has pdf P by $\mathbb{E}_P[f(\boldsymbol{X})]$, then for any pdf Q whose support contains that of P (that is, $P(\boldsymbol{x}) > 0 \Rightarrow Q(\boldsymbol{x}) > 0$),

$$\mathbb{E}_P[f(\boldsymbol{X})] = \int_{\mathbb{R}^n} f(\boldsymbol{x})P(\boldsymbol{x})d\boldsymbol{x} = \int_{\mathbb{R}^n} f(\boldsymbol{x})\frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}Q(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_Q\left[f(\boldsymbol{X})\frac{P(\boldsymbol{X})}{Q(\boldsymbol{X})}\right].$$

Therefore, letting $\{\tilde{\boldsymbol{X}}^{(i)}\}_{i=1}^m$ be i.i.d. samples with pdf Q, the IS estimate and associated variance estimate are

$$\hat{p}_Q := \frac{1}{m}\sum_{i=1}^m \frac{f(\tilde{\boldsymbol{X}}^{(i)})P(\tilde{\boldsymbol{X}}^{(i)})}{Q(\tilde{\boldsymbol{X}}^{(i)})}, \qquad \hat{V}_Q := \frac{1}{m^2}\sum_{i=1}^m \left(\frac{f(\tilde{\boldsymbol{X}}^{(i)})P(\tilde{\boldsymbol{X}}^{(i)})}{Q(\tilde{\boldsymbol{X}}^{(i)})} - \hat{p}_Q\right)^2. \tag{7}$$

If Q can be chosen in such a way that A is common *and* the likelihood ratio $P(\boldsymbol{X})/Q(\boldsymbol{X})$ does not have high variance for $\boldsymbol{X} \in A$, then $\text{Var}(\hat{p}_Q)$ can be much smaller (orders of magnitude) than $\text{Var}(\hat{p}_{\text{simple}})$ for the same m. The material from this subsection can be found with more detail in [21, Chap. 9].

2.4 Comments

CV and IS may be combined into one estimator, summarized in Algorithm 1, the full mathematical details of which are contained in [21, Sect. 9.10]. Although CV and IS can both be excellent frameworks for variance reduction in MC, there is no general method for selecting h or Q, and good choices for either one are extremely problem-dependent. Indeed, the main contribution of this paper is to find, for the important case of linear dynamics and Gaussian noise, h and Q that make MC estimation of CPs converge fast enough for real-time planning.

3 Problem Statement

We pose the problem of motion planning under uncertainty with safety tolerance as a constraint separate from the path cost to be optimized. We consider robots described by linear dynamics with control policies derived as LQG controllers tracking nominal trajectories. These nominal trajectories are planned assuming continuous dynamics, but in order to make the computation of path CPs tractable, we assume discretized (zero-order hold) approximate dynamics for the tracking controllers. The full details of the continuous vs. discrete problem formulations are rather standard and due to space limitations are provided in Appendix A of the extended version of this paper [7]. Briefly here, with $\mathcal{N}(\mu, \Sigma)$ denoting a multivariate Gaussian with mean μ and covariance matrix Σ, the system dynamics are given by

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, V), \quad \mathbf{y}_t = C\mathbf{x}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, W), \tag{8}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ is the state, $\mathbf{u}_t \in \mathbb{R}^\ell$ is the control input, $\mathbf{y}_t \in \mathbb{R}^o$ is the workspace output, and $\mathbf{v}_t$ and $\mathbf{w}_t$ represent Gaussian process and measurement noise, respectively. With deviation variables from a nominal trajectory defined as $\delta\mathbf{x}_t := \mathbf{x}_t - \mathbf{x}_t^{\text{nom}}$, $\delta\mathbf{u}_t := \mathbf{u}_t - \mathbf{u}_t^{\text{nom}}$, and $\delta\mathbf{y}_t := \mathbf{y}_t - \mathbf{y}_t^{\text{nom}}$, for $t = 0, \dots, T$, the discrete LQG controller $\delta\mathbf{u}_t^{\text{LQG}} := L_t\, \widehat{\delta\mathbf{x}}_t$, with L_t and $\widehat{\delta\mathbf{x}}_t$ denoting the feedback gain matrix and Kalman state estimate respectively, minimizes the tracking cost function

$$J := \mathbb{E}\left[\delta\mathbf{x}_T^T F \delta\mathbf{x}_T + \sum_{t=0}^{T-1} \delta\mathbf{x}_t^T Q \delta\mathbf{x}_t + \delta\mathbf{u}_t^T R \delta\mathbf{u}_t\right].$$

The computation details of L_t and the dynamics of $\widehat{\delta\mathbf{x}}_t$ are standard and given in Appendix A [7]; in the remainder of this paper we use only the notation that the combined state/estimate deviations evolve as multivariate Gaussians $\left[\delta\mathbf{x}_t; \widehat{\delta\mathbf{x}}_t\right] \sim \mathcal{N}\left(\mu_t, \Sigma_t\right)$ and for suitable definitions of M_t and N_t we may write

$$\begin{bmatrix}\delta\mathbf{x}_{t+1}\\ \widehat{\delta\mathbf{x}}_{t+1}\end{bmatrix} = M_t \begin{bmatrix}\delta\mathbf{x}_t\\ \widehat{\delta\mathbf{x}}_t\end{bmatrix} + \mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}\left(\mathbf{0}, N_t\right). \tag{9}$$

Let $\mathcal{X}_{\text{obs}}$ be the obstacle space, so that $\mathcal{X}_{\text{free}} := \mathbb{R}^d \backslash \mathcal{X}_{\text{obs}}$ is the free space. Let $\mathcal{X}_{\text{goal}} \subset \mathcal{X}_{\text{free}}$ and $\boldsymbol{x}_0 \in \mathcal{X}_{\text{free}}$ be the goal region and initial state. Given a path cost measure c and letting $\overline{\mathbf{x}_0, \dots, \mathbf{x}_T}$ denote the continuous curve traced by the robot's random trajectory through the discretization points $\mathbf{x}_0, \dots, \mathbf{x}_T$, we wish to solve

Discretized stochastic motion planning (SMP):

$$\begin{aligned}
\min_{\mathbf{u}^{\text{nom}}(\cdot)} \quad & c(\mathbf{x}^{\text{nom}}(\cdot)) \\
\text{s.t.} \quad & \mathbb{P}\left(\overline{\mathbf{x}_0, \dots, \mathbf{x}_T} \cap \mathcal{X}_{\text{obs}} \neq \varnothing\right) \leq \alpha \\
& \mathbf{u}_t = \mathbf{u}_t^{\text{nom}} + \delta\mathbf{u}_t^{\text{LQG}} \\
& \mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}_0^{\text{nom}}, P_0), \quad \mathbf{x}_T^{\text{nom}} \in \mathcal{X}_{\text{goal}} \\
& \text{Equation}(8).
\end{aligned} \tag{10}$$

Note that the optimization is still over continuous-time nominal paths, which we discretize when computing the path collision probability $\mathbb{P}\left(\overline{\mathbf{x}_0, \dots, \mathbf{x}_T} \cap \mathcal{X}_{\text{obs}} \neq \varnothing\right)$. The termination constraint $\mathbf{x}_T^{\text{nom}} \in \mathcal{X}_{\text{goal}}$ in (10) is deterministic as written; it is also possible to define a goal chance constraint similar to the obstacle avoidance constraint, e.g., $\mathbb{P}(\mathbf{x}_T \in \mathscr{X}_{\text{obs}}) \leq \beta$. We do not consider this variant in this paper, but note that our Monte Carlo approach also applies to estimating $\mathbb{P}(\mathbf{x}_T \in \mathscr{X}_{\text{obs}})$. Similar to how we ensure satisfaction of the collision chance constraint by inflating the obstacle set (see Sect. 6), satisfaction of a goal chance constraint could be ensured by shrinking the goal region when planning a nominal trajectory.

This formulation is inspired by [3, 18] and represents a compromise between a POMDP formulation involving a minimization over the class of output-feedback control laws, and an open-loop formulation, in which the state is assumed to evolve in an open loop (i.e., no tracking). This can be justified in two ways. One is that the general constrained POMDP formulation is vastly more complex than ours and would require much more computation. The other is that, in practice, a motion plan is executed in a receding horizon fashion, so that computing output-feedback policies may not even be useful, since after a short time-horizon the environment map may have changed, requiring recomputation anyway. We note that the problem formulation could be readily generalized to a nonlinear setup and to any tracking controller (e.g., LQG with extended Kalman filter estimation is essentially already in the same form)—indeed, one of the key advantages of the MC approach is that it is able to handle (at least theoretically) such general versions of the problem. However, in the present paper, we limit our attention to the aforementioned LQG setup.

In the remainder of the paper, we discuss how to quickly and consistently (i.e., in a way that is asymptotically exact as the discretization step $\Delta t \to 0$) estimate the path CP appearing in Eq. (10), and then we will employ MCMP to generate approximate solutions to the discretized SMP problem.

4 The Problem of Computing Path CP

In general, the key difficulty for planning under uncertainty (provided a probabilistic uncertainty model is given) is to accurately compute path CP. All previous methods beyond simple Monte Carlo essentially rely on two approaches, namely:

- **Additive approach**, e.g., [15]: using Boole's inequality, i.e., $\mathbb{P}(\cup_i A_i) \leq \sum_i \mathbb{P}(A_i)$, by which a path CP is approximated by *summing* pointwise CP_i at a certain number of waypoints along the path, i.e., $\text{CP} \approx \sum_i \text{CP}_i$.
- **Multiplicative approach**, e.g., [3]: a path CP is approximated by *multiplying* the complement of point-wise CP_i, specifically $\text{CP} \approx 1 - \prod_i (1 - \text{CP}_i)$.

There are three approximations inherent in both approaches:

(A) The path CP is approximated by combining waypoint CP_i's. That is, no accounting is made for what happens in between waypoints.

(B) The waypoint CP_i's are combined in an approximate manner. That is, in general there is a complex high-dimensional dependence between collisions at different waypoints, and these are not accounted for in either approach. In particular, the additive approach treats waypoint collisions as mutually exclusive, while

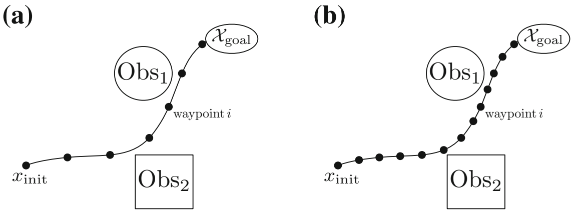


Fig. 1 Illustration of the interplay between approximations (A) and (B). In **a**, there are not enough waypoints to properly characterize the pathwise CP, while in **b**, the waypoints may be too close to not account for their dependence

the multiplicative approach treats them as independent. Since neither mutual exclusivity nor independence hold in general, this constitutes another approximation.

(C) Each waypoint CP$_i$ is approximated (e.g., by using a nearest obstacle). This is usually done because integrating a multivariate density over an intersection of half-planes (defining the obstacle set) can be quite computationally expensive.

A fundamental limitation in both approaches comes from the interplay between approximations (A) and (B). Specifically, while approximation (A) improves with higher-resolution waypoint placement along the path, approximation (B) actually gets worse, see Fig. 1. In Fig. 1a, although Obs$_2$ comes very close to the path, it does not come very close to any of the waypoints, and thus the pathwise CP will not be properly accounted for by just combining pointwise CPs. In Fig. 1b, the waypoints closest to Obs$_1$ will have highly-correlated CPs, which again is not accounted for in either the additive or multiplicative approaches. For the linear Gaussian setting considered here, as the number of waypoints along a fixed path goes to infinity, the path CP estimate from the additive approach actually tends to ∞, while that of the multiplicative approach tends to 1, *regardless* of the true path CP. To see this, note that for any fixed path, there exists a positive number $\varepsilon > 0$ such that CP$_i$ is larger than or equal to ε for *any* point on the path. Therefore, the additive and multiplicative approximations tend to, respectively,

$$\sum_{i=1}^{k} \mathrm{CP}_i \geq k\,\varepsilon \overset{k\to\infty}{\longrightarrow} \infty, \qquad 1 - \prod_{i=1}^{k} (1 - \mathrm{CP}_i) \geq 1 - (1-\varepsilon)^k \overset{k\to\infty}{\longrightarrow} 1, \tag{11}$$

where k is the number of waypoints. In other words, both approaches are asymptotically tautological, as they approximate the path CP with a number greater than or equal to one. An important consequence of this is that as the number of waypoints approaches infinity, either approach would deem *all* possible paths infeasible with respect to *any* fixed non-trivial path CP constraint. This point is emphasized in Fig. 2, which compares true path CP to approximations computed using the additive and multiplicative approaches for two different paths, as a function of the number of waypoints along the path. Off the plotted area, the additive approach passes through

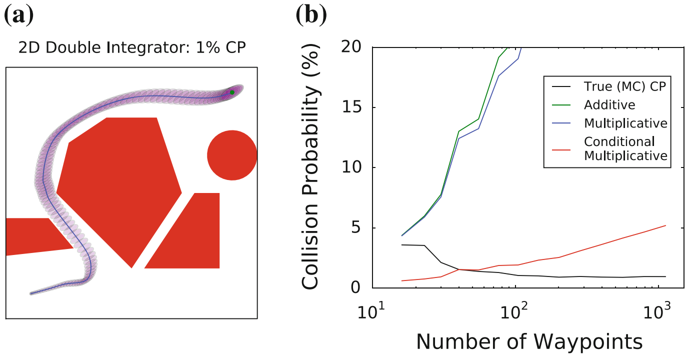


Fig. 2 Illustration of path CP approximation schemes for a robotic system where the true path CP is around 1% under the continuous controller. The *blue curve* in **a** represents the nominal path, the *red* polygons are the obstacles, and the *purple* ellipses represent 95% pointwise marginal confidence intervals at each of 104 discretization points. Panel **b** shows the collision probability estimated by each approximation scheme as a function of the number of discretization points. Approximation C in all approaches is matched to their respective papers (additive: [2, 9, 15, 16], multiplicative: [3], conditional multiplicative: [14, 22, 25])

an approximate probability of 1 and continues to infinity, while the multiplicative approach levels off at 1. Even with few waypoints, both approaches are off by hundreds of percent. The overly conservative nature of the multiplicative approach has been recognized in [22], where the authors replace approximate marginal pointwise CPs in the multiplicative approach with approximate pointwise CPs *conditional* on the previous waypoint being collision-free. While this is a first-order improvement on the standard approaches, the conditional pointwise probabilities are quite complex but are approximated by Gaussians for computational reasons, with the result that their approximate path CPs can still be off by many multiples of the true value, especially for small path CPs (which will usually be the relevant ones). The red curve in Fig. 2b shows that the approximation of [22], while a substantial improvement over the alternatives, can still be off by factors of 5 or more, and the discrepancy appears to be increasing steadily with the number of waypoints.

A few comments are in order. First, there is nothing pathological in the example in Fig. 2, as similar results are obtained with other obstacle configurations and in higher dimensions. Second, we note that approximations such as these may be very useful for *unconstrained* problems that penalize or minimize path CP, since they may measure the *relative* CP between paths well, even if they do not agree with the true path CP in absolute terms. However, to address the chance-constrained SMP, one needs an accurate (in absolute terms) and fast method to estimate path CP, which is one of the key contributions of this paper. Third, the additive approach is guaranteed to be conservative with respect to approximation (B). That is, ignoring (A) and (C) (the latter of which can also be made conservative), the additive approach will produce an overestimate of the path CP. Although this can result in high-cost paths or even problem infeasibility, it is at least on the safe side. This guarantee comes at a cost of extreme conservativeness, and the two less-conservative multiplicative approaches have no such guarantee for any finite number of waypoints. Fourth, the limits in Eq. (11) apply to any uncertainty model supported on the entire configuration space, and even to bounded uncertainty models so long as the path in question has a positive-length segment of positive pointwise CP.

In the next section, we will present a MC approach that addresses all three approximations (A)–(C) stated earlier. Specifically, for (A), although collisions can truly be checked along a continuous path only for special cases of obstacles, Monte Carlo simply checks for collisions along a sampled path and thus can do so at arbitrary resolution, regardless of the resolution of the actual waypoints, so approximation (A) for MC has no dependence on waypoint resolution. For (B), the high-dimensional joint distribution of collisions at waypoints along the path is automatically accounted for when sampling entire realizations of the tracking controller. And for (C), since MC only has to check for collisions at specific points in space, no multivariate density integration needs to be done.

5 Variance-Reduced Monte Carlo for Computing Pathwise CP

5.1 Control Variates

As discussed in Sect. 2.2, a good control variate h for f should have a known (or rapidly computable) expected value, and should be highly correlated with f. As mentioned in the previous section, existing probability-approximation methods, while not accurate in absolute terms, can act as very good proxies for CP in that they resemble a monotone function of the CP. Coupled with the fact that such approximations are extremely fast to compute, they make ideal candidates for h.

Since even individual waypoint CPs are expensive to compute exactly for all but the simplest obstacle sets, we approximate the obstacle set locally as a union of half-planes, similar to [22]. For each waypoint $\mathbf{x}_t^{\text{nom}}$ along the nominal path, we compute the closest obstacle points $\mathbf{z}_t^{(i)}$ and their corresponding obstacle half-planes such that none of these points are occluded by each others' half-planes (see Fig. 3a). "Close" is measured in terms of the Mahalanobis distance defined by the covariance matrix of the robot state at that waypoint, and the obstacle half-planes are defined as tangent to the multivariate Gaussian density contour at each close point. Mathematically this corresponds to at most one point per convex obstacle region $\mathscr{X}_{\text{obs}}^{(i)}$ (with $\mathscr{X}_{\text{obs}} = \bigcup_{i=1}^{M} \mathscr{X}_{\text{obs}}^{(i)}$), i.e.,

$$\mathbf{z}_t^{(i)} = \underset{\mathbf{a} \in \mathscr{X}_{\text{obs}}^{(i)}}{\arg\min} (\mathbf{a} - \mathbf{x}_t^{\text{nom}})^T \Sigma_t^{-1} (\mathbf{a} - \mathbf{x}_t^{\text{nom}}).$$

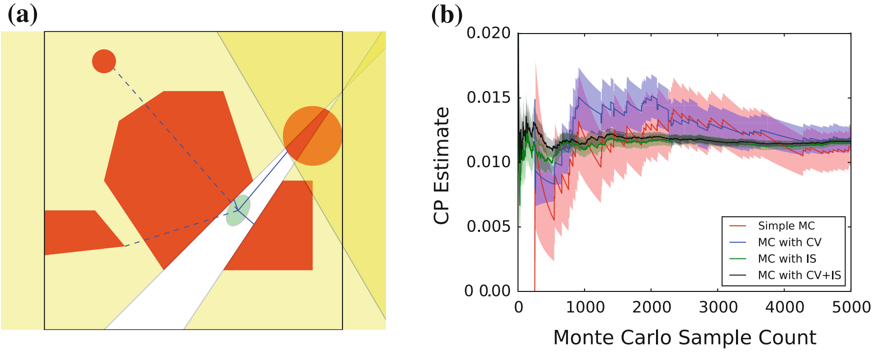


Fig. 3 **a** Approximation of an obstacle set as a union of half-planes. Half-planes for the *dashed-line* closest obstacle points are omitted, as their corresponding obstacle regions are occluded by a closer half-plane. The *green* ellipse represents the shape of the uncertainty covariance matrix which defines the "close" distance metric. **b** Comparison of MC variance-reduction methods applied to a path with true CP near 1%. Bands denote 95% confidence intervals for their respective estimates

We then approximate the pointwise probability of collision by the probability of crossing any one of these half-planes; this probability is approximated in turn by Boole's bound so that an expectation is simple to compute. That is, we define $h_{ti}(\boldsymbol{X})$ to be the indicator that $\mathbf{x}_t$ crosses the $\mathbf{z}_t^{(i)}$ obstacle half-plane, and define $h(\boldsymbol{X}) = \sum_{t,i} h_{ti}(\boldsymbol{X})$, the count of all half-planes crossed by the points of the MC trajectory $\boldsymbol{X}$. We note that considering multiple close obstacle points, as opposed to only the closest one, is important when planning in tight spaces with obstacles on all sides. Correlations between h and f in testing were regularly around 0.8.

5.2 *Importance Sampling*

From a statistical standpoint, the goal in selecting the importance distribution Q is to make the pathwise CP sampled under Q on the order of 1, while keeping the colliding paths sampled from Q as likely as possible under the nominal distribution P. In the language of Sect. 2.3, the former goal makes our collision event common under Q, and the latter goal ensures that the likelihood ratio $P(\boldsymbol{X})/Q(\boldsymbol{X})$ does not have high variance. From a computational standpoint, we want Q to be fast to sample from and for the likelihood ratio P/Q to be easy to compute. Our method for importance sampling constructs Q as a mixture of sampling distributions $Q_0, \ldots, Q_K$—one for each close obstacle point $\mathbf{z}_t^{(i)}$ along the nominal trajectory. The intent of distribution Q_{ti} is to sample a path that is likely to collide with the obstacle set at waypoint t. We accomplish this by shifting the means of the noise distributions $\mathbf{n}_s$, $0 \le s \le t$, leading up to time t so that $\mathbb{E}_{Q_{ti}}[\delta \mathbf{x}_t] = \mathbf{z}_t^{(i)} - \mathbf{x}_t^{\text{nom}}$ (see (12) constraint). To minimize likelihood ratio $P(\boldsymbol{X})/Q_{ti}(\boldsymbol{X})$ variance, we distribute the shift in the most likely manner according to Mahalanobis distance (see (12) objective). This amounts, through Eq. (9), to solving the least-squares problem

$$\begin{aligned} \min_{\Delta\mu_0,\ldots,\Delta\mu_t} \quad & \sum_{s=0}^{t} \Delta\mu_s^T N_s^{-1} \Delta\mu_s \\ s.t. \quad & \sum_{s=0}^{t} [I \; 0] \left(\prod_{r=0}^{t-s-1} M_{t-r} \right) \Delta\mu_s = \mathbf{z}_t^{(i)} - \mathbf{x}_t^{\text{nom}} \end{aligned} \tag{12}$$

and sampling the noise as $\tilde{\mathbf{n}}_s \sim \mathcal{N}\left(\Delta\mu_s, N_s\right)$ for $0 \leq s \leq t$.

We weight the full mixture IS distribution, with $\theta = \mathbb{E}[h(\boldsymbol{X})]$, as

$$Q = \sum_{t,i} \left(\frac{\mathbb{E}[h_{ti}(\boldsymbol{X})]}{\theta} \right) Q_{ti}.$$

That is, the more likely it is for the true path distribution to collide at t, the more likely we are to sample a path pushed toward collision at t.

5.3 Combining the Two Variance-Reduction Techniques

Due to space limitations, we do not discuss the full details of combining CV and IS here, but simply state the final combined procedure in Algorithm 1. We note here, however, a modification to the set of close obstacle points $\mathbf{z}_t^{(i)}$ defined above which slightly changes the control variate h and sampling distribution Q. In order to speed up the Monte Carlo CP estimation, we prune the close obstacle points so that the ones that remain are expected to have their term in the mixture distribution Q sampled at least once. This style of pruning does not bias the results; it only affects computation time (omitting many near-zero terms) and estimator variance.

6 MCMP Algorithm

We now incorporate this algorithm for path CP estimation into a simple scheme for generating high-quality paths subject to a path CP constraint. Algorithm 2 describes the Monte Carlo Motion Planning (MCMP) algorithm in pseudocode.

The idea of MCMP is simple: solve the deterministic motion planning problem with inflated obstacles to make the resulting path safer, and then adjust the inflation so that the path is exactly as safe as desired. A bisection search is employed to select this inflation. Note that in line 3, MC could be replaced by any of the approximations from Sect. 4, but the output would suffer in quality. In the case of the multiplicative approaches, the CP may be underestimated, in which case the safety constraint will be violated. More commonly, for most discretizations, the CP will be substantially overestimated for both additive and multiplicative approaches. Although the resulting path will not violate the safety constraint, it will be inefficient in that it will take a

Algorithm 1 Monte Carlo Path CP Estimation

```
Require: Nominal distribution $P$, control variate $h$ as in Sect. 5.1, $\theta := \mathbb{E}_P[h(\boldsymbol{X})]$, importance distribution $Q$ as in Sect. 5.2, number of samples $m$
Sample $\{\tilde{\boldsymbol{X}}^{(i)}\}_{i=1}^m$ i.i.d. from $Q$
Denoting the likelihood ratio $L(\tilde{\boldsymbol{X}}^{(i)}) := P(\tilde{\boldsymbol{X}}^{(i)})/Q(\tilde{\boldsymbol{X}}^{(i)})$, compute
    $$\hat{p}_Q = \frac{1}{m}\sum_{i=1}^m f(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}), \qquad \hat{\theta}_Q = \frac{1}{m}\sum_{i=1}^m h(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}),$$
    $$\hat{\beta}_Q = \frac{\sum_{i=1}^m \left(f(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) - \hat{p}_Q\right)\left(h(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) - \hat{\theta}_Q\right)}{\sum_{i=1}^m \left(h(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) - \hat{\theta}_Q\right)^2},$$
    $$\hat{p}_{Q,\hat{\beta}_Q} = \frac{1}{m}\sum_{i=1}^m f(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) - \hat{\beta}_Q h(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) + \hat{\beta}_q\theta$$
    $$\hat{V}_{Q,\hat{\beta}_Q} = \frac{1}{m^2}\sum_{i=1}^m \left(f(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) - \hat{p}_{Q,\hat{\beta}_Q} - \hat{\beta}\left(h(\tilde{\boldsymbol{X}}^{(i)})L(\tilde{\boldsymbol{X}}^{(i)}) - \theta\right)\right)^2$$
return $\hat{p}_{Q,\hat{\beta}_Q}$, $\hat{V}_{Q,\hat{\beta}_Q}$
```

Algorithm 2 Monte Carlo Motion Planning

```
Require: Maximum inflation $I_{\max}$ (e.g., configuration space diameter), minimum inflation $I_{\min}$ (e.g., 0), number of bisection steps $r$, path CP constraint $\alpha$
for $i = 1 : r$ do
    Compute an (approximately) optimal path $\hat{\sigma}$ using, e.g., an asymptotically optimal sampling-based motion planning (SBMP) algorithm, for the deterministic version of the problem with the obstacles inflated by $(I_{\min} + I_{\max})/2$
    Use Algorithm 1 to compute a MC estimate $\hat{p}$ of the CP of $\hat{\sigma}$ (set $\hat{p} = 0$ if the previous step fails to find a feasible solution)
    if $\hat{p} > \alpha$ then
        $I_{\min} = (I_{\min} + I_{\max})/2$
    else
        $I_{\max} = (I_{\min} + I_{\max})/2$
    end if
end for
return $\hat{\sigma}$
```

costlier path than needed (or than the one returned by using MC CP estimation) in order to give the obstacles a wider berth than necessary. Another possibility is that the obstacle inflation needed to satisfy the conservative safety constraint actually closes off all paths to the goal, rendering the problem infeasible, even if it may have been feasible using MC estimation.

It is worth pointing out that the tails, or probability of extreme values, of the Gaussian distribution fall off very rapidly, at a double-exponential rate. For instance, the 0.01^{th} percentile of a Gaussian distribution is only about 20% farther from the mean than the 0.1^{th} percentile. In the Gaussian framework of this paper, this means that a path that already has a small CP can make its CP much smaller by only shifting slightly farther from the obstacles. Thus although the additive or multiplicative approximations may overestimate the pathwise CP by hundreds of percent, the cost difference between using them in line 3 of Algorithm 2 and using MC in line 3 of Algorithm 2 may not be nearly so drastic. However, as noted above, the cost difference could be great if the increased obstacle inflation required closes off an entire homotopy class, or renders the problem infeasible altogether. Thus accurate CP computation is essential to the application of the MCMP bisection algorithm.

7 Numerical Experiments

We implemented variance-reduced path CP estimation and MCMP in Julia [4] for numerical experiments on a range of linear dynamical systems and obstacle sets, run using a Unix operating system with a 2.0 GHz processor and 8 GB of RAM. Many implementation details and tuning parameters have been omitted in the discussion below; the code for these results may be accessed at https://github.com/schmrlng/MCMP-ISRR15.

Figures 2 and 4 display some example results for single integrator ($\dot{x} = u$) and double integrator ($\ddot{x} = u$) systems in a two-dimensional workspace, and Table 1 summarizes a range of statistics on algorithm performance in three-dimensional workspaces as well. The deterministic planning step (Algorithm 2, line 2) was accomplished using the differential FMT* algorithm [24] on a fixed set of nodes. By caching nearest-neighbor and distance data for these nodes (Offline Planning), the total replanning time over all inflation factors (Online Planning, essentially consisting only of collision checking) was significantly reduced. For the single integrator systems 2D SI and 3D SI, the planning problem is equivalent to geometric planning, which allowed us to apply the ADAPTIVE-SHORTCUT rubber-band-style heuristic for smoothing

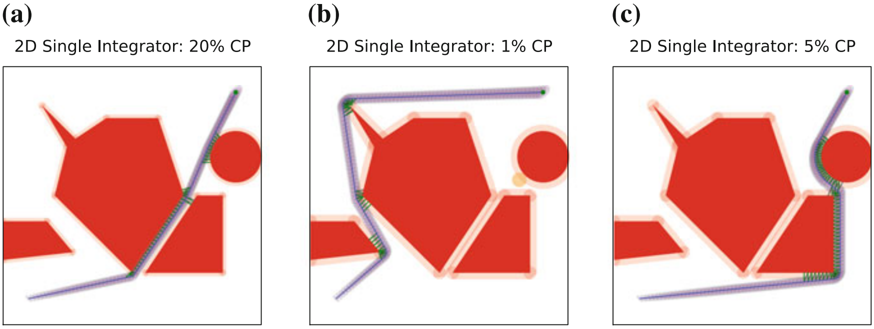

(a)
2D Single Integrator: 20% CP
(b)
2D Single Integrator: 1% CP
(c)
2D Single Integrator: 5% CP

Fig. 4 Illustration of the MCMP algorithm output given a range of target path CPs for a 2D single integrator system. For these uncertainty parameters, we see that the precise safety tolerance value (between 1–20%) will correspond to a nominal solution in one of three distinct homotopy classes. The *orange* obstacle in **b** is added by the "block and backtrack" modification, discussed in Sect. 7, to the basic MCMP bisection Algorithm 2. The *black* and *green lines* denote the close obstacle points and vectors defining their half planes respectively; only the pruned set is depicted

planned paths [6]. Applying this smoothing heuristic ensures that the path CP varies continuously with inflation factor within a path homotopy class. Between homotopy classes the CP may be discontinuous as a function of inflation factor. If increasing the inflation factor increases the CP discontinuously, the bisection process is not affected; otherwise if the CP decreases (e.g., Fig. 4b, c—the CPs are 0.3 and 1.5% respectively around the inflation factor which closes off the (c) route) the MCMP bisection algorithm may get stuck before reaching the target CP α (e.g. 1% in the case of Table 1 row 2). To remedy this issue, we implemented a "block and back-track" modification to Algorithm 2 which blocks off the riskier homotopy class with an obstacle placed at its waypoint most likely to be in collision, and then resets the bisection lower bound for the inflation factor. This results in increased computation time, but returns a path with the goal CP in the end.

We did not implement any smoothing procedure for the double integrator systems. Each nominal trajectory is selected as a concatenation of local steering connections, subject to variance in the placement of the finite set of planning nodes. In practice, this means that path CP is piecewise constant, with many small discontinuities, as a function of inflation factor. If the bisection procedure terminates at an interval around the desired CP, we choose the path satisfying the safety constraint: this explains the mean True CP below the goal value in Table 1 rows 4 and 5.

From Table 1 we see that MCMP run times approach real time in a range of state spaces from 2–6 dimensions, on the order of 5–10 s total, excluding planning computation that may be cached offline. This is accomplished even at a level of tracking discretization sufficient to approximate continuous LQG. Planning time and probability estimation time are similar in magnitude, indicating that the MC portion of MCMP is not significantly holding back algorithm run time compared to a faster approximation scheme, even in this single processor implementation. Computing the Monte Carlo path simulations (MC Particles) in parallel could greatly reduce that time. We note that the few thousand simulations required in total by MCMP would not be enough to certify, using simple Monte Carlo, that a path CP is within the interval (0.9, 1.1%) even once, which highlights the effectiveness of our proposed estimator variance reduction techniques.

As can be seen from the simulations, the accuracies of the additive, multiplicative, and conditional multiplicative approximations vary over problems and parameters, even occasionally being quite accurate. At this level of discretization, we see that the conditional multiplicative approximation scheme is within a factor of 2 of the true CP value, but may either underestimate or overestimate depending on which of approximation (A) or (B) from Sect. 4 has the stronger effect. This sheds light on a key difference between using MC to estimate path CP as opposed to its alternatives: MC not only gives accurate estimates, but also comes with a *standard error* of that estimate, effectively allowing the user to know whether or not the estimate is a good one. There is no such information for the various other approximations; they simply return point values with no bearing on their relation to the ground truth. The standard error estimate that comes from MC can be used as a kind of certificate of accuracy that gives the user confidence in a trajectory plan's estimated probability of success, as well as some assurance that it is not overly conservative.

Table 1 (MCMP in various state spaces). Results averaged over 400 MCMP runs: 20 runs each for 20 SBMP sample sets. SI and DI denote single and double integrator respectively. We aim to minimize arc-length for the SI systems, and a mixed time/control energy cost for the DI systems. 2D SI (A) refers to the obstacle set in Fig. 2, and 2D SI (B) refers to the obstacle set in Fig. 4. We note that for the DI systems, the mean True CP found through MCMP bisection lies below the goal value. This is because no smoothing heuristic was applied to the solutions output by the deterministic planning step. Thus replanning over the range of inflation factors is choosing from a discrete set of path options; in the event that none of these options corresponds exactly to the goal CP, we choose the side of the bisection interval that satisfies the safety constraint

	Goal CP (%)	Offline Planning (s)	Online Planning (s)	MC Time (s)	Discretization Points	Bisection Iterations	MC Particles
2D SI (A)	1	0.25 ± 0.03	1.25 ± 0.24	2.64 ± 0.83	102.5 ± 0.8	6.3 ± 1.5	2085 ± 686
2D SI (B)	1	0.27 ± 0.04	2.48 ± 0.77	4.65 ± 1.70	116.5 ± 0.7	13.3 ± 4.4	2955 ± 1052
3D SI	1	0.35 ± 0.03	1.95 ± 0.76	3.00 ± 0.89	83.6 ± 1.4	6.3 ± 3.1	1667 ± 764
2D DI	1	6.64 ± 0.14	2.86 ± 0.98	5.82 ± 2.33	107.7 ± 6.0	8.6 ± 2.9	2383 ± 952
3D DI	1	20.90 ± 1.11	6.27 ± 2.40	7.45 ± 3.75	71.7 ± 10.8	7.8 ± 3.3	2117 ± 938

	Goal CP (%)	Nominal Path Cost	True (MC) CP (%)	Additive Estimate (%)	Multiplicative Estimate (%)	Cond. Mult. Estimate (%)
2D SI (A)	1	1.47 ± 0.00	1.01 ± 0.06	22.67 ± 2.39	20.35 ± 1.92	2.04 ± 0.20
2D SI (B)	1	1.69 ± 0.01	1.00 ± 0.06	12.88 ± 3.72	12.10 ± 2.97	1.37 ± 0.26
3D SI	1	1.28 ± 0.03	1.00 ± 0.06	47.48 ± 7.98	38.84 ± 5.51	2.15 ± 0.23
2D DI	1	7.20 ± 0.43	0.67 ± 0.27	15.04 ± 8.97	13.78 ± 7.59	1.39 ± 0.68
3D DI	1	9.97 ± 1.61	0.66 ± 0.33	12.26 ± 5.88	11.68 ± 5.43	0.59 ± 0.32

8 Conclusion

We have presented a computationally fast method for provably-accurate pathwise collision probability estimation using variance-reduced Monte Carlo. The variance-reduction techniques employ a novel planning-specific control variate and importance distribution. This probability-estimation technique can be used as a component in a simple meta-algorithm for chance-constrained motion planning, generating low-cost paths that are not conservative with respect to a nominal path CP constraint.

Simulation results confirm our theory, and demonstrate that computation can be done at speeds amenable to real-time planning.

This works leaves many avenues for further investigation, the foremost of which is parallelization. As noted earlier, a key feature of MC is that it is trivially parallelizable (which is not changed by CV or IS). As most of the computation time is spent computing likelihood ratios, which is mostly linear algebra, our technique is ideally suited for implementation on a GPU. Another future research direction is to extend this work to more general controllers and uncertainty models. For instance, confirming compatibility of MCMP with LQG with an extended Kalman filter would allow it to be used for systems with nonlinear dynamics and Gaussian noise. Heavier-tailed distributions than Gaussian would require larger shifts in inflation factor to affect similar changes in path CP, making a non-conservative CP estimation procedure all the more important. Monte Carlo itself is extremely flexible to these parameters, but it remains to be seen if appropriate control variates or importance distributions can be developed to speed it up. We note that the meta-algorithm mentioned in this paper is extremely simple, and can surely be improved upon. One potential improvement is to incorporate domain knowledge to differentially inflate the constraints, or to do so in an iterative or adaptive way, similar in spirit to [19]. Another improvement could be to make bisection search adaptive and to incorporate the uncertainty in the probability estimates. Finally, although we use our MC method to solve the chance-constrained motion planning problem, it is in no way tied to that problem, and we plan to test our method on other problems, such as minimizing CP or optimizing an objective function that penalizes CP.

References

1. Agha-Mohammadi, A., Chakravorty, S., Amato, N.M.: FIRM: sampling-based feedback motion planning under motion uncertainty and imperfect measurements. Int. J. Robot. Res. **33**(2), 268–304 (2014)
2. Aoude, G.S., Luders, B.D., Joseph, J.M., Roy, N., How, J.P.: Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. Auton. Robots **35**(1), 51–76 (2013)
3. Berg, J.V.D., Abbeel, P., Goldberg, K.: LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. Int. J. Robot. Res. **30**(7), 895–913 (2011)
4. Bezanson, J., Karpinski, S., Shah, V.B., Edelman, A.: Julia: A Fast Dynamic Language for Technical Computing (2012). http://arxiv.org/abs/1209.5145
5. Blackmore, L., Ono, M., Bektassov, A., Williams, B.C.: A probabilistic particle-control approximation of chance-constrained stochastic predictive control. IEEE Trans. Robot. **26**(3), 502–517 (2010)
6. Hsu, D.: Randomized Single-Query Motion Planning in Expansive Spaces. Ph.D. thesis, Stanford University (2000)
7. Janson, L., Schmerling, E., Pavone, M.: Monte Carlo Motion Planning for Robot Trajectory Optimization Under Uncertainty (Extended Version) (2015). http://arxiv.org/abs/1504.08053
8. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artif. Intell. **101**(1–2), 99–134 (1998)
9. Kothari, M., Postlethwaite, I.: A probabilistically robust path planning algorithm for UAVs using rapidly-exploring random trees. J. Intell. Robot. Syst. **71**(2), 231–253 (2013)

10. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Robotics: Science and Systems, pp. 65–72 (2008)
11. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
12. LaValle, S.M.: Motion planning: wild frontiers. IEEE Robot. Autom. Mag. **18**(2), 108–118 (2011)
13. LaValle, S.M., Kuffner, J.J.: Randomized Kinodynamic Planning. Int. J. Robot. Res. **20**(5), 378–400 (2001). doi:10.1177/02783640122067453. http://ijr.sagepub.com/content/20/5/378.short
14. Liu, W., Ang, M.H.: Incremental Sampling-Based Algorithm for Risk-Aware Planning Under Motion Uncertainty. In: Proceedings of IEEE Conference on Robotics and Automation, pp. 2051–2058 (2014)
15. Luders, B., Kothari, M., How, J.P.: Chance constrained RRT for probabilistic robustness to environmental uncertainty. In: AIAA Conferenvce on Guidance, Navigation and Control (2010)
16. Luders, B.D., Karaman, S., How, J.P.: Robust sampling-based motion planning with asymptotic optimality guarantees. In: AIAA Conference on Guidance, Navigation and Control (2013)
17. Luders, B.D., Sugel, I., How, J.P.: Robust trajectory planning for autonomous parafoils under wind uncertainty. In: AIAA Conference on Guidance, Navigation and Control (2013)
18. Oldewurtel, F., Jones, C., Morari, M.: A tractable approximation of chance constrained stochastic MPC based on affine disturbance feedback. In: Proceedings of IEEE Conference on Decision and Control, pp. 4731–4736 (2008)
19. Ono, M., Williams, B.C.: Iterative risk allocation: a new approach to robust model predictive control with a joint chance constraint. In: Proceedings of IEEE Conference on Decision and Control, pp. 3427–3432 (2008)
20. Ono, M., Williams, B.C., Blackmore, L.: Probabilistic planning for continuous dynamic systems under bounded risk. J. Artif. Intell. Res. **46**, 511–577 (2013)
21. Owen, A.B.: Monte Carlo theory, methods and examples (2013). http://statweb.stanford.edu/owen/mc/
22. Patil, S., van den Berg, J., Alterovitz, R.: Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In: Proceedings of IEEE Conference on Robotics and Automation, pp. 3238–3244 (2012)
23. Reif, J.H.: Complexity of the Mover's Problem and Generalizations. In: 20th Annual IEEE Symposium on Foundations of Computer Science, pp. 421–427 (1979)
24. Schmerling, E., Janson, L., Pavone, M.: Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics. In: Proceedings IEEE Conference on Decision and Control (2015)
25. Sun, W., Torres, L.G., Berg, J.V.D., Alterovitz, R.: Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In: International Symposium on Robotics Research (2013)
26. Sun, W., Patil, S., Alterovitz, R.: High-frequency replanning under uncertainty using parallel sampling-based motion planning. IEEE Trans. Robot. **31**(1), 104–116 (2015)
27. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust Monte Carlo localization for mobile robots. Artif. Intell. **128**(1–2), 99–141 (2001)
28. Vitus, M.P., Tomlin, C.J.: On feedback design and risk allocation in chance constrained control. In: Proceedings of IEEE Conference on Decision and Control, pp. 734–739 (2011)

A Computational Framework for Environment-Aware Robotic Manipulation Planning

Marco Gabiccini, Alessio Artoni, Gabriele Pannocchia and Joris Gillis

1 Introduction

Careful observation of how humans use their hands in grasping and manipulation tasks clearly suggests that their limbs extensively engage functional interactions with parts of the environment. The physical constraints imposed by the manipulandum and the environment are not regarded as obstacles, but rather as opportunities to guide functional hand pre-shaping, adaptive grasping, and affordance-guided manipulation of objects. The exploitation of these opportunities, which can be referred to as *environmental constraints* (EC), enables robust grasping and manipulation in dynamic and highly variable environments. When one considers the exploitation of EC, i.e. when manipulation actions are performed *with the help* of the environment, the boundary between grasping and manipulation is blurred, and traditional categories such as grasp and manipulation analysis, trajectory planning and interaction control appear somewhat artificial, as the problem we aim to solve seems to inextricably entangle all of them.

In this paper, we set out to formulate environment-aware manipulation planning as a nonlinear optimal control problem and discretize it according to a *direct transcription* scheme [3]. In Sect. 3, two approaches to describe the dynamics of systems with contacts are proposed and evaluated: in the first one, continuous contact reaction

M. Gabiccini (✉) · A. Artoni · G. Pannocchia
Department of Civil and Industrial Engineering, University of Pisa, Pisa, Italy
e-mail: Marco.Gabiccini@unipi.it

A. Artoni
e-mail: Alessio.Artoni@unipi.it

G. Pannocchia
e-mail: Gabriele.Pannocchia@unipi.it

J. Gillis
Department of Electrical Engineering, KU Leuven, Leuven, Belgium
e-mail: joris.gillis@kuleuven.be

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_21

forces are generated by nonlinear virtual springs, and the requirement to avoid sliding contacts is handled in an apparently original way; in the second one, contact collisions are approximated as impulsive events causing discontinuous jumps in the velocities according to a modified version of the Stewart-Trinkle time-stepping scheme [47]. The introduction of two different models is motivated by the relative ease for the first (second) one to enforce EC exploitation primitives that avoid (profitably exploit) sliding motions during interaction.

Both formulations lead to a nonlinear programming (NLP) problem (Sect. 4) that we solve by using the Interior-Point (IP) method implemented in IPOPT [4] and discussed in Sect. 5. To improve computational efficiency, we also annotate sparsity in the linear algebra expressions and leverage algorithmic differentiation (AD) [23] to calculate derivatives both quickly and accurately: the adoption of the CasADi framework [2], described in Sect. 5, provides a profitable interface to all the above tools with a consistent API.

In Sect. 6, we evaluate our approaches in three simulated planar manipulation tasks: (i) moving a circular object in the environment with two independent fingers, (ii) rotating a capsule with an underactuated two-fingered gripper, and (iii) rotating a circular object in hand with three independent fingers. Tasks (i) and (ii) show that our algorithm quickly converges to locally optimal solutions that opportunistically exploit EC. Task (iii) demonstrates that even dexterous fingertip gaits can be obtained as a special solution in the very same framework. Conclusions are drawn in Sect. 7.

It is worth noting that with our method, approach planning, grasping, manipulation, and environment constraint exploitation phases occur automatically and opportunistically for a wide range of tasks and object geometries, with no a-priori specification of the ordering of the different stages required.

2 Related Work

2.1 Exploitation of Environmental Constraints

The concept of exploiting environmental constraints is well-rooted in robotics. Pioneering work was performed already in the eighties in the context of motion planning [34] and manipulation [35]. However, these concepts did not have the proper influence on many of the recent developments on either area, perhaps due to the inadequacy of the mechanical impedance properties of contemporary industrial manipulators to achieve sliding motion primitives stably, thus precluding the adoption of strategies that exploit environmental constraints, e.g. by sliding one object over another [10]. Also the idea of programming using environmental constraints is well entrenched in robotics literature, starting with the seminal work [1], proceeding with [45], and culminating in the *iTaSC* framework [11].

The exploitation of complex interactions with the environment in a manipulation task also plays a central role in automation and manufacturing to design fixtures [7] and part feeders [6]. However, these works are highly specialized and they are limited to the case of handling a single object geometry.

2.2 Traditional Grasp Planners

State-of-the-art general grasping algorithms and dexterous mechanical hands lie at the opposite end of the spectrum: they are designed to perform grasping and manipulation of a wide range of object geometries and for many different tasks. Traditional grasp planners (such as OpenRAVE [12] and GraspIt! [36]) rely on precise finger-to-object contact points while avoiding the surrounding environment. In real-world scenarios these models, as well as the motion of the hand, will be highly uncertain leading to poor grasping performance for grasps that were deemed highly robust based on theoretical considerations.

The recent paper [5] has proposed a pipeline for automated grasp synthesis of common objects with a compliant hand by developing a full-fledged multi-body simulation of the whole grasping process in the presence of EC: however, to date, the approach seems time consuming and the sequence of primitive actions needed to perform complex tasks must be scripted in advance. Also recently, both grasp planning algorithms and grasping mechanisms have begun to take advantage of EC [14], albeit not systematically. While sequences of EC exploitation primitives have been shown to be robust and capable [9, 28], there has been no comprehensive research on how to enumerate and describe these primitives or how to sequence them into task-directed manipulation plans. A noteworthy exception where a sequence of task-directed manipulation plans exploiting EC is created and performed on-line with minimalistic sensory information and limited previous assumptions about the scene was recently presented in [16].

2.3 General Purpose Planning Algorithms

State-of-the-art sampling-based planning algorithms like RRT* [27] seem not tailored for situations where a-priori unknown contact interactions may cause a combinatorial explosion of system configurations. Recent extensions, initially presented in [20] for RRT, and successively devised for RRT* in [42], were able to cope with systems described by complex and underactuated dynamics. In [30], the authors presented an approach to moving an object with several manipulators. This problem presents some similarities to ours, since a sequence of phases has to be both planned and solved. The strong assumption that the plan called by the high-level scheduler will succeed — which is not always possible — has been removed in the recent contribution [8].

However, situations where intermitted contact sequences are not easily enumerated from the outset, but are key for the success of environment-aware manipulation plans, still appear to be out of their reach.

2.4 Machine Learning Approaches

Although significant progresses have been made in this area in recent years [29], learning robot motion skills still remains a major challenge, especially for systems with multiple intermittent contacts. Policy search is often the preferred method as it scales gracefully with system dimensionality, even if its successful applications typically rely on a compact representation that reduces the numbers of parameters to learn [31, 43, 49]. Innovative policy classes [25] have led to substantial improvements on real-world systems. However, designing the right low-dimensional representation often poses significant challenges. Learning a state representation that is consistent with physics and embeds prior knowledge about interactions with the physical world has recently been proposed in [26] and seems a promising venue to find effective methods to help improve generalization in reinforcement learning: however, the simulated robotic tasks solved by this methods are still far in complexity from environment-aware manipulation scenarios.

The recent contribution [32] developed a policy search algorithm which combines a sample-efficient method for learning linear-Gaussian controllers with the framework of guided policy search, which allows the use of multiple linear-Gaussian controllers to train a single nonlinear policy with any parametrization, including complex and high-dimensional policies represented by large neural networks. In [33], this method has been recently applied, with some modifications that make it practical for deployment on a robotic platform, to solve contact-rich manipulation tasks with promising results.

2.5 Optimization-Based Trajectory Planning

Various research groups are currently pursuing direct trajectory optimization to synthesize complex dynamic behaviors for systems with intermittent contacts. Those working in locomotion [46] mainly adopt a multi-stage hybrid-mode approach (usually employing multiple-shooting), where the optimization is constrained to operate within an a priori specification of the mode ordering. Interestingly, a recent contribution [52] explored the synthesis of optimal gaits for legged robots without the need to specify contact sequences. Certainly, the adoption of such an approach seems the only viable solution for a multi-fingered hand manipulating an object also by exploiting EC. Along this line, it is worth mentioning the contact-invariant approach originally proposed in [38] to discover complex behaviors for humanoid figures and extended to the context of manipulation in [37]. The previously described trajectory optimization method has been recently employed to gradually train a neural network by following an Alternating Direction Method of Multipliers (ADMM) strategy with interesting results [39].

The approach presented in [44] inspired our work and is the one which is definitely closest. However, remarkable differences in the formulation of the dynamics for systems with contacts, in the choice of the solution algorithm, solver and framework, and in the focus of the paper — here, EC exploitation is sought as a key factor — render our work significantly different.

3 Dynamics of Systems with Contacts

3.1 Penalty-Based Contact Model

To our eyes, manipulation planning has to rely on a dynamic model of the system, namely of manipulandum, manipulator, and the environment, and of their mutual interactions through contact. We consider here deterministic systems with continuous state and control spaces, denoted by x and u respectively. The dynamic evolution of our controlled (non-autonomous) system can be described in continuous time t by a set of Ordinary Differential Equations (ODEs):

$$\dot{x}(t) = F(x(t), u(t)) \quad \text{or} \quad \tilde{F}(\dot{x}(t), x(t), u(t)) = 0 \tag{1}$$

(explicit dependence on t is omitted hereafter). Together with an initial value $x(0) = x_0$, Eq. (1) defines an initial value problem. In general, additional algebraic dependencies may exist among $\dot{x}$, x and u leading to a dynamic system governed by differential-algebraic equations (DAEs). In the presence of contact, for instance, and if contact forces are included among the controls u, contact interactions establish functional dependencies between u and x through a set of algebraic equations/inequalities. As an example, if f_N is the normal contact force and $g_N = g_N(x)$ the normal gap (shortest distance) between a finger and the object being manipulated, the complementarity and non-negativity conditions $0 \leq f_N \perp g_N(x) \geq 0$ must hold. The contact model described in this section is based on a special treatment of the contact forces and the relative velocities that arise during interaction between manipulandum, manipulator, and environment. Such a model has proved to be successful in solving trajectory planning problems for manipulation tasks with *no sliding*, in the presence of EC.

For normal contact forces, the underlying idea is borrowed from classical penalty-based approaches, where contact interactions are modeled by spring-dampers. In our model, no damping is introduced, while a nonlinear exponential spring relates the normal contact force and the normal gap through the constitutive equation (Fig. 1)

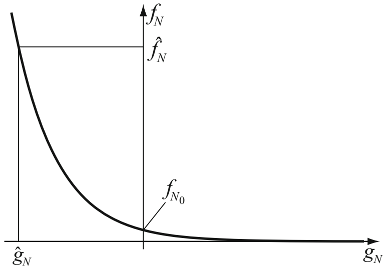


Fig. 1 Normal contact force as a function of the normal gap

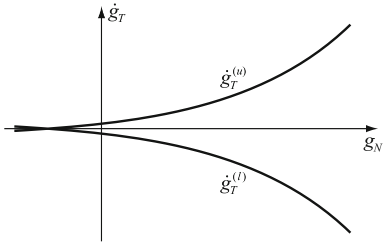


Fig. 2 Example of sliding velocity funnel

$$f_N(g_N(x)) = f_{N_0}\left(\frac{\hat{f}_N}{f_{N_0}}\right)^{g_N(x)/\hat{g}_N} \tag{2}$$

where $g_N(x)$ is the normal gap function and $\hat{g}_N$ the negative normal gap value (penetration) corresponding to the normal force value $\hat{f}_N$. The (fixed) parameters $(\hat{f}_N, \hat{g}_N)$ provide a straightforward way to properly "calibrate" the model and adapt it to the problem at hand. Relation (2) is a relaxation of the above-stated complementarity condition: it avoids discontinuities while being sufficiently representative of physical reality. In order to describe (unilateral) contacts with friction, the classical Coulomb model is adopted. The focus is placed here on point-contact with *static* friction, whereby normal force f_N, tangential force f_T and the coefficient of static friction μ_s are related by the well known relation

$$f_T \leq \mu_s f_N \tag{3}$$

No-sliding conditions must be enforced by requiring that sliding velocities at contact points be zero. The normal gap $g_N(x)$ and the sliding velocity $\dot{g}_T(\dot{x}, x)$ (i.e., the time derivative of the tangential gap [51]) would call for an additional complementarity condition. We devised a smooth relaxation of this discontinuous condition through the *sliding velocity funnel* shown in Fig. 2 and described by:

$$\dot{g}_T^{(l)}(g_N) \leq \dot{g}_T \leq \dot{g}_T^{(u)}(g_N) \tag{4}$$

where the functions $\dot{g}_T^{(l)}(g_N)$ and $\dot{g}_T^{(u)}(g_N)$ are lower and upper bounds, respectively, for the sliding velocity. It is reasonable to have a symmetric funnel, hence $\dot{g}_T^{(l)}(g_N) = -\dot{g}_T^{(u)}(g_N)$. The bounds are modeled here as:

$$\dot{g}_T^{(u)}(g_N) = \exp\left(c_1(g_N + c_2)\right) + c_3 \tag{5}$$

where the three parameters (c_1, c_2, c_3) are used for proper, problem-dependent calibration. No-sliding manipulation planning benefits from this approach as the sliding velocity funnel smoothly and gradually guides the fingers towards the object, eventually driving relative velocities to (nearly) zero at their contact points. The trajectory planning methods employed in this work are based on numerical optimization techniques that require a discrete-time model of the dynamic system, therefore discrete-time versions of Eqs. (1)–(4) are adopted in the following. In our implementation, the state vector $x_k = x(t_k)$ collects configuration and velocity of each body, while control vector $u_k = u(t_k)$ includes acceleration of each finger (or actuator) and contact forces at each candidate contact point. The dynamic equation (1) is used in a direct transcription scheme based on *collocation points*: using a single collocation point as midpoint in the interval $[t_k, t_{k+1}]$, and denoting $\bar{t}_k = (t_{k+1} + t_k)/2$, $\bar{x}_k = x(\bar{t}_k)$ and $h = t_{k+1} - t_k$ (fixed), the discretized dynamic equation becomes

$$x_{k+1} = x_k + hF(\bar{x}_k, u_k) \tag{6}$$

In the above implementation, states are linear and controls are constant over each discretization interval. The integration scheme (6) is known as implicit midpoint rule ($O(h^2)$), and it is the special one-point case of the Gauss–Legendre collocation method. As it is a symplectic integrator, it is suitable to cope with stiff, conservative mechanical systems. While Eqs. (3)–(4) are straightforward to discretize, Eq. (2) needs some attention: as it involves both states and controls, its discretization must adhere to the scheme dictated by Eq. (6), to wit

$$f_{N_k} = f_N\left(g_N(\bar{x}_k)\right) = f_{N_0}\left(\frac{\hat{f}_N}{f_{N_0}}\right)^{g_N(\bar{x}_k)/\hat{g}_N} \tag{7}$$

Failing to do so (e.g., evaluating g_N at x_k) would result in a "causality violation", with contact forces being inconsistent with the interpenetrations between bodies governed by (6).

3.2 Velocity-Based Time Stepping Scheme

In this section, we present the complementarity formulation of the time-stepping scheme employed for the dynamic modeling of manipulation systems exploiting *sliding over* EC. To keep the treatment general enough, we refer to a 3D system of m bodies with c contacts. We assume a polyhedral approximation of the friction cone, with d friction directions uniformly distributed to positively span the contact tangent plane.[1] For ease of notation, we write $M_k = M(q_k)$ and likewise for other vector/matrix functions. Let h be again the time step, and let $\Delta v := v_{k+1} - v_k$. For $k \in \{0, \ldots, N-1\}$, we adopt a Backward-Euler transcription scheme by writing the *kinematic reconstruction* and the *dynamic* equations as

$$q_{k+1} - q_k - h\, v_{k+1} = 0 \tag{8a}$$

$$M_{k+1}\Delta v - h\big[\kappa(q_{k+1}, v_k) + B_{k+1}u_{k+1}\big] - G_{k+1}\lambda_{k+1} = 0, \tag{8b}$$

where: $q \in \mathbb{R}^{6m}$ and $v \in \mathbb{R}^{6m}$ represent system configuration and velocity, respectively, $M \in \mathbb{R}^{6m\times 6m}$ is the generalized mass matrix, $\kappa \in \mathbb{R}^{6m}$ collects centrifugal, Coriolis and gravitational forces, $u \in \mathbb{R}^{t}$ is the control torque vector, $B \in \mathbb{R}^{6m\times t}$ is the actuation matrix, $G = [N\ \ T]$ is a *generalized grasp* matrix, where $N \in \mathbb{R}^{6m\times c}$ and $T \in \mathbb{R}^{6m\times nd}$ are normal and tangential wrench bases, and $\lambda = [\lambda_N^\top\ \ \lambda_T^\top]^\top$ is the generalized wrench impulse vector, wherein λ_N and λ_T are the normal and tangential contact wrench impulses. Matrix N appears as $N = [N^{(1)} \cdots N^{(c)}]$, and each column $N^{(i)} \in \mathbb{R}^{6m}$ corresponds to contact i and contains, for each body ℓ connected to contact i, a block of rows of the form $\pm[n_i^\top\ \ (p_{\ell,i} \times n_i)^\top]^\top$.[2] Since there are at most two bodies connected to a contact, each $N^{(i)}$ has at most 12 non-zero elements. Similarly, $T = [T^{(1)} \cdots T^{(c)}]$, and in the generic block $T^{(i)} \in \mathbb{R}^{6m\times d}$, each column $T^{(i,j)} \in \mathbb{R}^{6m}$ contains, for each body ℓ connected to contact i, a block of rows of the form $\pm[t_{i,j}^\top\ \ (p_{\ell,i} \times t_{i,j})^\top]^\top$, where $t_{i,j}$ denotes friction direction j at contact i. Opposite signs must be selected for each of the two bodies connected to contact i, and each column of T will contain, at most, 12 non-zero elements.

In partial accordance to [47], *unilateral* contacts with friction can be described by the following set of *inequality* and *complementarity* conditions

$$0 \le \lambda_{N_{k+1}} \perp\ g_N(q_{k+1}) \ge 0 \tag{9a}$$

$$0 \le \lambda_{T_{k+1}} \perp\ \big(\dot g_T(q_{k+1}, v_{k+1}) + E\gamma_{k+1}\big) \ge 0 \tag{9b}$$

$$0 \le \gamma_{k+1} \perp\ \big[\mu\lambda_{N_{k+1}} - (\lambda_{T_{k+1}}^\top H\, \lambda_{T_{k+1}})^{\frac{1}{2}}\big] \ge 0, \tag{9c}$$

where $g_N(\cdot)$ is the normal gap function and $\dot g_T(\cdot)$ is the time derivative of the tangential gap function [51], γ represents, in most cases,[3] an approximation to the magnitude of the relative contact velocity, matrix $E := BlockDiag(\mathbf{1}, \ldots, \mathbf{1}) \in \mathbb{R}^{(dc)\times c}$, with $\mathbf{1} \in \mathbb{R}^d$, $\mu \ge 0$ is the coefficient of friction, and $H := BlockDiag(H^{(1)}, \ldots, H^{(c)})$,

[1] For simplicity of description, we assume that the number of friction directions d is the same at each contact, although this is not necessary.

[2] The positive/negative sign must be chosen if, considering equilibrium of body ℓ, the unit normal vector n_i is facing into/away from body ℓ.

[3] In situations where the relative contact velocity and the friction vector are both zero, $\gamma \ge 0$ can be arbitrary and has no physical meaning.

where the $H_{lm}^{(i)} = t_{i,l}^{\top} t_{i,m}$ $(l, m \in \{1, \ldots, d\})$ is the metric form [13, Sects. 2–5] of the basis $t_{i,l}$ that positively spans the tangent plane at contact i. Equation (9a) state that bodies cannot interpenetrate ($g_N(q_{k+1}) \geq 0$), normal impulses can only push objects away ($\lambda_{N_{k+1}} \geq 0$), and that, in order for the impulse to be non-zero in the interval $[t_k, t_{k+1}]$, the normal gap must be closed at t_{k+1}. This condition also implies that collisions are approximated here as inelastic ones, and interacting bodies may end up sticking together. Equation (9b) require tangential impulses to be directed along the positive tangential directions ($\lambda_{T_{k+1}} \geq 0$). The complementarity condition in (9b) selects, for sliding contacts, the tangential impulse that opposes the sliding velocity. This constraint is tightly coupled with the complementarity condition in Eq. (9c), and it ensures that, if a contact is sliding, the tangential force will lie on the boundary of the friction cone. It is worth noting that the bracketed term in Eq. (9c) allows one to correctly define the Coulomb friction constraints even in sticking conditions, as it is robust to the physiological failure of Eq. (9b) in selecting only one non-zero component in each $\gamma_{k+1}^{(i)}$ for adhesive contacts.[4]

The choice of fully implicit integration schemes and nonlinear complementarity formulations, as described in Eqs. (8) and (9), can be justified in view of the increased numerical stability and modelling accuracy they bring about, while not hindering the general structure of the problem.[5]

4 Trajectory Planning as an Optimization Problem

4.1 Penalty-Based Contact Model

Within our direct transcription framework, for $k \in \{0, \ldots, N-1\}$, Eqs. (6), (7), and the discretized versions of Eqs. (3) and (4) constitute a set of (equality and inequality) *nonlinear constraints* for the optimal control problem (OCP) we set out to formulate. Additional constraints include the (fixed and known) initial state values $x_0 = x(0)$ as well as a *terminal equality constraint* on (some) components of x_N: the latter provides a direct way to specify the required final state of the manipulated object at the final time $T = t_N$. Generally, no terminal constraints on the manipulation system configuration are imposed. Lower and upper bounds for $(x_k, \bar{x}_k, u_k)$ are also included: they act as operational constraints for actuators and the system's workspace, and they are useful to restrain contact forces within safety limits. Other constraints can be introduced to shape emergent behaviors and to render them intrinsically more robust or desirable for several reasons. As an illustrative example, in order

[4] Replacing the bracketed term in (9c) with $[\mu\lambda_N - E^{\top}\lambda_T]$, as commonly performed in literature [48], would call for unrealistically strict and physically unmotivated conditions to ensure adhesive friction.

[5] Embedding contact dynamics into the numerical optimization problem as nonlinear constraints, where many other implicit constraints are already present, does not justify explicit or semi-implicit discretization schemes, which are, instead, legitimate when building fast simulators [21, Sect. 5].

to guarantee that any two fingers make always contact with an object in a three-fingered manipulation task, we add the following set of inequalities to the problem: $f_{N_k}^{(1)} + f_{N_k}^{(2)} \geq \varepsilon$, $f_{N_k}^{(2)} + f_{N_k}^{(3)} \geq \varepsilon$, and $f_{N_k}^{(3)} + f_{N_k}^{(1)} \geq \varepsilon$, with $\varepsilon > 0$.

We introduce the vector of decision variables $\boldsymbol{v} \in \mathbb{R}^n$, which collects the sequence of unknown $(x_k, \bar{x}_k, u_k)$ (i.e., configurations and velocities in $(x_k, \bar{x}_k)$, contact forces and actuator accelerations in u_k). All equality and inequality constraints can be written compactly as

$$g_{\min} \leq g(\boldsymbol{v}) \leq g_{\max}, \qquad \boldsymbol{v}_{\min} \leq \boldsymbol{v} \leq \boldsymbol{v}_{\max} \tag{10}$$

The general structure of the cost function takes the form

$$f(\boldsymbol{v}) = \sum_{k=0}^{N-1} \sum_{i \in \mathscr{I}} w_i \phi_i(x_k, u_k), \tag{11}$$

where each $\phi_i(\cdot)$ represents a peculiar type of cost. These have to be carefully selected according to the character of the manipulation action we desire to perform, along with the corresponding *weights* w_i (also acting as important scaling factors). Multiple cost terms $\phi_i(\cdot)$ can be used to shape different manipulation behaviors. The following terms (specifically, their squared 2-norm) have proved to be decisive in directing the optimization process: contact forces, and their variations from one interval to the next (to minimize jerk); accelerations of actuators; deviations of actual object trajectories from ideal, smooth trajectories (task-specific).

4.2 Velocity-Based Time Stepping Scheme

Similarly to the penalty-based contact model scheme, the discrete formulation of the dynamics of systems with contacts expressed by Eqs. (8) and (9) can be used as a set of nonlinear constraints in an optimal control problem (OCP), involving the sequence of unknown $(q_k, v_{k+1}, \lambda_{k+1}, \gamma_{k+1})$. Additional equality constraints are introduced: for the manipulated object, both initial and final configurations and velocities are imposed, whereas only the initial conditions are specified for the manipulation system.

Inequality constraints are also introduced with similar intentions as in the penalty-based scheme. It is worth noting that, since in our applications the hand is velocity controlled, the hand dynamics is not included in the optimization constraints, and the hand velocities will play the role of control actions. Therefore, limited control authority is imposed as bounds on hand velocities and accelerations in the form $v_{\min}^{(l)} \leq v^{(l)} \leq v_{\max}^{(l)}$ and $a_{\min}^{l} h \leq \Delta v^{(l)} \leq a_{\max}^{(l)} h$, respectively, with l belonging to the index set corresponding to the hand.

Defining $\boldsymbol{v} \in \mathbb{R}^n$ as the multi-stage sequence of configurations, velocities, contact impulses and inputs, $(q_k, v_{k+1}, \lambda_{k+1}, \gamma_{k+1})$, all constraints are still expressed in the form (10), whereas the cost function takes the form

$$f(\boldsymbol{v}) = \sum_{k=0}^{N-1} \sum_{i \in \mathscr{I}} w_i \phi_i (q_k, v_{k+1}, \lambda_{k+1}, \gamma_{k+1}) \tag{12}$$

4.3 Final Optimization Problem

From the previous discussion, we now present the nonlinear program (NLP) that has to be solved to generate optimal trajectories. With $\boldsymbol{v} \in \mathbb{R}^n$ previously defined, we consider the following optimization problem

$$\min_{\boldsymbol{v}} f(\boldsymbol{v}), \qquad \text{subject to} \quad g_{\min} \leq g(\boldsymbol{v}) \leq g_{\max} \quad \boldsymbol{v}_{\min} \leq \boldsymbol{v} \leq \boldsymbol{v}_{\max} \tag{13}$$

in which: $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $g : \mathbb{R}^n \to \mathbb{R}^m$ is the nonlinear constraint function, $g_{\min} \in [-\infty, \infty)^m$ and $g_{\max} \in (-\infty, \infty]^m$ (with $g_{\min} \leq g_{\max}$) are, respectively, lower and upper bound vectors of the nonlinear constraints, $\boldsymbol{v}_{\min} \in [-\infty, \infty)^n$ and $\boldsymbol{v}_{\max} \in (-\infty, \infty]^n$ (with $\boldsymbol{v}_{\min} \leq \boldsymbol{v}_{\max}$) are, respectively, lower and upper bound vectors of the decision variables. Problem (13) is a *large-scale*, but *sparse* NLP, that should be solved by structure-exploiting solvers. To this end, as detailed in the next section, we resorted to the IPOPT [50] implementation of the interior-point method within the CasADi framework. As initial guess required by the algorithm, we somewhat crudely mapped the initial state x_0 and a rough estimate of the controls to all the $(N-1)$ variable instances. Obviously, better initial guesses should be provided whenever possible. With the penalty-based approach, (partial) solutions of the NLP (13) have also been used as initial guesses for a subsequent optimization according to a *homotopy* strategy [40, Sect. 11.3], thereby maximizing physical realism while facilitating convergence.

5 Nonlinear Programming via an Interior-Point Algorithm

5.1 The Barrier Problem Formulation

Problem (13) is equivalently rewritten as

$$\min_{x} f(x), \qquad \text{subject to} \tag{14a}$$

$$c(x) = 0 \tag{14b}$$

$$x_{\min} \leq x \leq x_{\max} \tag{14c}$$

in which x is formed by augmenting the decision variable vector $\boldsymbol{v}$ with suitable slack variables that transform inequality constraints in (13) into equality constraints.[6]

Let $I_{\min} = \{i : x_{\min}^{(i)} \neq -\infty\}$, and $I_{\max} = \{i : x_{\max}^{(i)} \neq \infty\}$. We consider the barrier function

$$\varphi_\mu(x) = f(x) - \mu \left(\sum_{i \in I_{\min}} \ln(x^{(i)} - x_{\min}^{(i)}) + \sum_{i \in I_{\max}} \ln(x_{\max}^{(i)} - x^{(i)}) \right)$$

in which $\mu > 0$ is a (small) barrier parameter. Instead of solving (14), IPOPT performs iterations to achieve an approximate solution of the equality constrained NLP

$$\min_x \varphi_\mu(x), \qquad \text{subject to: } c(x) = 0 \tag{15}$$

Note that $\varphi_\mu(x)$ is well defined if and only if $x_{\min} < x < x_{\max}$, i.e. if x is in the *interior* of its admissible region. The value of μ is progressively reduced so that $\varphi_\mu(x) \to f(x)$, and in this way solving (15), in the limit, becomes equivalent to solving (14). Clearly, as $\mu \to 0$, any component of x can approach its bound if this is required by optimality.

5.2 Interior-Point Approach to NLP

Any local minimizer to (15) must satisfy the following Karush–Kuhn–Tucker (KKT) conditions [40, Sect. 12.2]

$$\nabla f(x) + \nabla c(x)\lambda - \underline{z} + \overline{z} = 0 \tag{16a}$$

$$c(x) = 0 \tag{16b}$$

$$\underline{z}^{(i)}(x^{(i)} - x_{\min}^{(i)}) - \mu = 0 \quad \forall i \in I_{\min} \tag{16c}$$

$$\overline{z}^{(i)}(x_{\max}^{(i)} - x^{(i)}) - \mu = 0 \quad \forall i \in I_{\max} \tag{16d}$$

for some vectors $\lambda \in \mathbb{R}^m$, $\underline{z} \in \mathbb{R}^n$, and $\overline{z} \in \mathbb{R}^n$ (for completeness: $\underline{z}^{(i)} = 0 \quad \forall i \notin I_{\min}$, $\overline{z}^{(i)} = 0 \quad \forall i \notin I_{\max}$). Notice that, if $\mu = 0$, then (16) together with $\underline{z} \geq 0$ and $\overline{z} \geq 0$ represent the KKT conditions for NLP (14). The KKT conditions (16) form a nonlinear algebraic system $F(\xi) = 0$ in the unknown $\xi = (x, \lambda, \underline{z}, \overline{z})$, which is solved in interior-point algorithms via Newton-like methods. If we denote by $E_\mu(\xi)$ the maximum absolute error of the KKT equations (16) (appropriately scaled), the basic algorithm implemented in IPOPT is summarized in Table 1 (in which j is the index of the outer loop, k is the index of the inner loop, and $\varepsilon > 0$ is a user-defined convergence tolerance).

[6] With a slight abuse of notation, we still use n and m to denote the dimension of x and $c(x)$, respectively, and $f(x)$ to denote $f(\boldsymbol{v})$.

Table 1 Basic Algorithm implemented in IPOPT

1.	Define $\mu_0 > 0$, x_0 ($x_{\min} \leq x_0 \leq x_{\max}$), λ_0, $\underline{z}_0 \geq 0$, $\bar{z}_0 \geq 0$, and form ξ_0 accordingly. Set: $j = 0, k = 0$
2.	Given the current iterate ξ_k, compute a Newton step p_k for $F(\xi) = 0$. Compute the new iterate performing a line search: $\xi_{k+1} = \xi_k + \alpha_k p_k$ (for some $\alpha_k > 0$)
3.	If $E_0(\xi_{k+1}) \leq \varepsilon$, exit: ξ_{k+1} is a local solution to NLP (14). Otherwise, proceed to Step 4
4.	If $E_{\mu_j}(\xi_{k+1}) \leq \kappa\mu_j$ (for some $\kappa > 0$) proceed to Step 5. Otherwise, update $k \leftarrow k + 1$ and go to Step 2
5.	Set $\mu_{j+1} = \mu_j / \rho$ (for some $\rho > 1$), update $j \leftarrow j + 1, k \leftarrow k + 1$ and go to Step 2

5.3 *Main Computational Aspects: Calculating Derivatives and Solving (Sparse) Linear Systems*

The most expensive computation step in the basic interior-point algorithm is the computation of the Newton step p_k for the KKT system $F(\xi) = 0$, i.e. Step 2. We first note that evaluation of $F(\xi)$, at each iteration, involves the computation of the cost function gradient $\nabla f(x) \in \mathbb{R}^n$ and of the constraint Jacobian $\nabla c(x) \in \mathbb{R}^{n \times m}$. Then, the Newton step is found from the solution of the following linear system:

$$\begin{bmatrix} W_k & A_k & -I & I \\ A_k^T & 0 & 0 & 0 \\ \underline{Z}_k & 0 & \underline{X}_k & 0 \\ -\overline{Z}_k & 0 & 0 & \overline{X}_k \end{bmatrix} \begin{bmatrix} p_k^x \\ p_k^\lambda \\ p_k^{\underline{z}} \\ p_k^{\bar{z}} \end{bmatrix} = - \begin{bmatrix} \nabla\varphi_{\mu_j}(x_k) + A_k\lambda_k \\ c(x_k) \\ \underline{X}_k\underline{Z}_k\mathbf{1} - \mu_j\mathbf{1} \\ \overline{X}_k\overline{Z}_k\mathbf{1} - \mu_j\mathbf{1} \end{bmatrix} \tag{17}$$

in which: $W_k = \nabla^2_{xx}\mathscr{L}(x_k, \lambda_k, \underline{z}_k, \bar{z}_k)$, with $\mathscr{L}(x, \lambda, \underline{z}, \bar{z}) = f(x) + c(x)^T\lambda - (x - x_{\min})^T\underline{z} - (x_{\max} - x)^T\bar{z}$ the Lagrangian function associated with NLP (14); $A_k = \nabla c(x_k)$ the constraint Jacobian; $\underline{Z}_k = \mathrm{diag}(\underline{\mathrm{z}}_\mathrm{k})$, $\overline{Z}_k = \mathrm{diag}(\bar{\mathrm{z}}_\mathrm{k})$, $\underline{X}_k = \mathrm{diag}(\mathrm{x_k} - \mathrm{x_{min}})$, and $\overline{X}_k = \mathrm{diag}(\mathrm{x_{max}} - \mathrm{x_k})$ diagonal matrices; $\nabla\varphi_{\mu_j}(x_k) = \nabla f(x_k) - \underline{z}_k + \bar{z}_k$. In order to generate the entries of system (17), it is necessary to evaluate the cost function gradient, the constraint Jacobian, as well as the Hessian of the Lagrangian (or a suitable approximation to it). Partial derivatives can be computed numerically by finite differentiation or analytically (for simple functions). A third approach is by means of so-called *Automatic Differentiation* (or Algorithmic Differentiation) techniques, which generate a numerical representation of partial derivatives by exploiting the chain rule in a numerical environment. Different approaches exist for AD, which are tailored to the computation of first-order and second-order derivatives. The interested reader is referred to [22]. A final computation observation is reserved to the numerical solution of system (17). First, it is transformed into a symmetric (indefinite) linear system via block elimination. Then, symmetry can be exploited by symmetric LDL factorizations. Furthermore, it should be noted that in trajectory planning problems considered here (and in general in optimal control problems) the Hessian W_k and the constraint Jacobian A_k are significantly sparse and structured.

Exploiting these features can reduce the solution time significantly. To this effect, the MA57 multifrontal solver [15] from the Harwell Software Library [24] is used.

5.4 The CasADi Framework

The transcribed optimal control problem is coded in a scripting environment using the Python [41] interface to the open-source CasADi framework [2], which provides building blocks to efficiently formulate and solve large-scale optimization problems.

In the CasADi framework, symbolic expressions for objective and constraints are formed by applying overloaded mathematical operators to symbolic primitives. These expressions are represented in memory as computational graphs, in contrast to tree representations common to computer algebra systems. The graph is sorted into an in-memory algorithm which can be evaluated numerically or symbolically with an efficient stack-based virtual machine or be exported to C code. Forward and backward source-code transforming AD can be performed on such algorithm at will, such that derivatives of arbitrary order can be computed. The sparsity pattern of the constraint Jacobian is computed using hierarchical seeding [19] and its unidirectional graph coloring is used to obtain the Jacobian with a reduced number of AD sweeps [18]. Regarding expressions and algorithm inputs and outputs, everything is a sparse matrix in CasADi. Yet the underlying computational graphs may be of either a type with scalar-valued (SX) nodes or a type with matrix-valued (MX) nodes. The combined usage of these two types amounts to a checkpointing scheme [22]: low-level functions are constructed with the SX type algorithm, which is optimized for speed. These algorithms are in turn embedded into a graph of the MX type, which is optimized for memory usage, to form the expression of objective and constraints.

In the context of optimal control problems, the CasADi framework offers several advantages over other AD tools: it comes bundled with common algorithms that can be embedded into an infinitely differentiable computational graph (e.g. numerical integrators, root-finding and linear solvers), and takes care of constructing and passing sensitivity information to various NLP solvers backends. Since CasADi does not impose an OCP solution strategy and allows fine-grained speed-memory trade-offs, it is suited more than black-box OCP solvers to explore non-standard optimal control problem formulations or efficient solution strategies.

6 Application Examples

6.1 Environment-Aware Manipulation

6.1.1 Penalty-Based Approach with Disk and Two Independent Fingers

Figure 3 shows a first example of EC-exploiting manipulation. In a vertical plane, the two independent fingers H_0 and H_1, initially away from the circular object, must interact with the object and have it interact with the environment (edges e_0 and e_1) so that it will be in the shown final position, with any orientation but zero velocity, at the end of a prescribed time horizon T. All contact interactions must occur without slippage (static friction). The object's initial state corresponds to a configuration of static equilibrium. The fingers have limitations on their horizontal workspace: as a result, grasping and lifting of the object is inhibited, and an environment-exploiting policy needs to be discovered in order to accomplish the task. Also, object-passing between fingers needs to emerge. This planning problem has been formulated and solved using the penalty-based approach. The resulting trajectories in terms of normal contact forces are shown in the first two plots of Fig. 4: finger H_0 approaches the object first (whose weight is symmetrically supported by e_0 and e_1), then rolls it on edge e_1 (without slipping) until it reaches its workspace limit and hands it over to finger H_1, which completes the task. Friction forces (not shown for brevity) satisfy constraint (3), where $\mu_s = 2$ was used. The third plot shows the actual x-component trajectory of the object versus a suggested trajectory, included as a hint in the objective function to facilitate convergence of the algorithm, but with a low weight (to avoid forcing such trajectory against dynamic constraints). With $N = 180$ discretization intervals (time step $h = 45\,\text{ms}$) and considering the prescribed initial and terminal conditions, the problem size is $n = 8810$ decision variables. To obtain a solution for

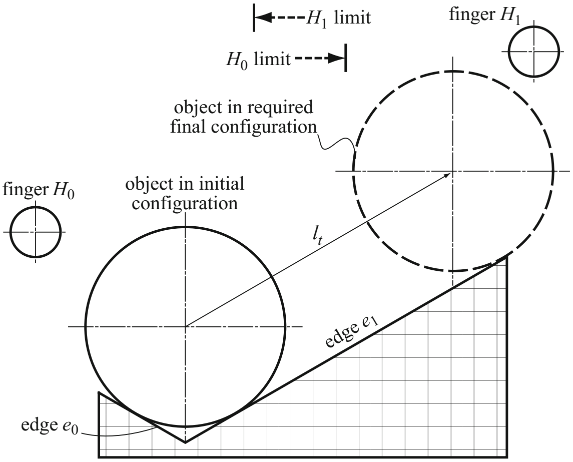


Fig. 3 EC manipulation scenario: the object must reach its final configuration at the prescribed final time $T = 8\,\text{s}$

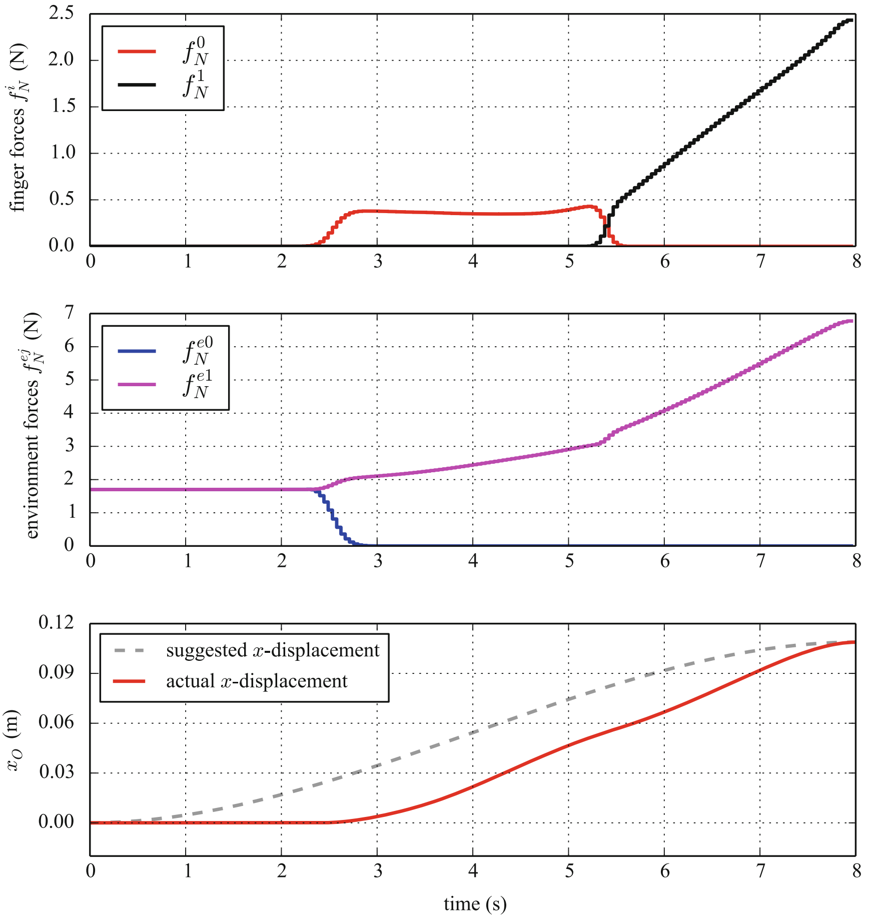


Fig. 4 Trajectories for a circular object manipulated by two independent fingers: normal contact forces f_N^i applied by the fingers; normal contact forces f_N^{ej} applied by the environment (segments e_0, e_1); suggested and actual x-displacement x of the object

this example, starting from an initial guess built by repeating the initial condition for all the time steps, Ipopt took 1062 iterations, which correspond to ∼23 min on a 2.70 GHz Intel(R) Core(TM) i7-4800MQ CPU with 32 GB of RAM. An animation of the obtained results can be found in part A.1 of the accompanying video [17] which also shows the grasping-and-lifting behavior that is discovered if finger workspace limitations are removed and the contact force exerted by edge e_1 is penalized.

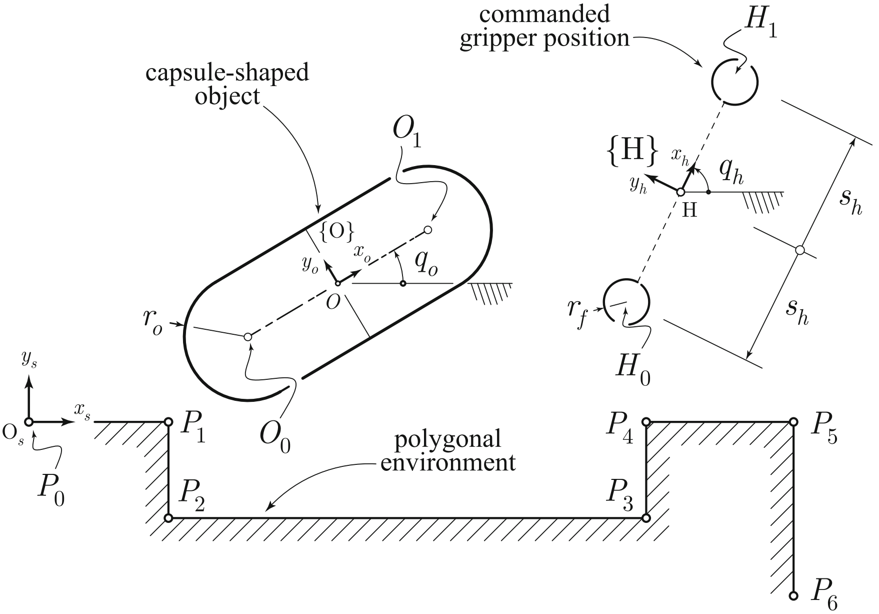


Fig. 5 EC manipulation scenario. Starting at $q_O(0) = \pi/2$, in contact with segment P_2P_3, the capsule must be placed, at time $T = 5\,\mathrm{s}$, in the same position but with $q_O(T) = -\pi/2$

6.1.2 Velocity-Based Time-Stepping Scheme with Capsule and Two-Fingered Underactuated Gripper

With reference to Fig. 5, a capsule-shaped object, starting from an equilibrium configuration in contact with segment P_2P_3 (of a six-edged polygonal environment), has to find itself rotated by 180° at the end of the planning horizon T. Since the object is passive, a manipulation gait has to emerge for the gripper. Moreover, since we penalize high contact impulses and the gripper has a reduced mobility due to underactuation — it has symmetrically closing jaws — it turns out that convenient EC-exploiting behaviors are indeed automatically synthesized by the optimizer. In fact, with reference to Fig. 6, beside finger impulses (first plot), which represent standard grasping/manipulation actions, contact interactions generated by collisions of the object with the environment (second plot) play a role of paramount importance in shaping the object motion. More in detail, with reference to part A.2 of the accompanying video [17], the object is initially grasped and lifted, then it is gently dropped so that it lays on segment P_2P_3 after hitting segment P_3P_4 (see the corresponding bumps in λ_N^{23} and λ_N^{34}). Then, with the circular part of the capsule pushed to corner P_3, the object is rotated with only one finger by sliding it on edges P_2P_3 and P_3P_4. Finally, both fingers grasp the object and, slightly lifting it up, they slide it on edge P_2P_3 to the initial position. It is worth noting that, with a wise exploitation of EC, the actual rotation of the object can closely follow the desired one (third plot). This

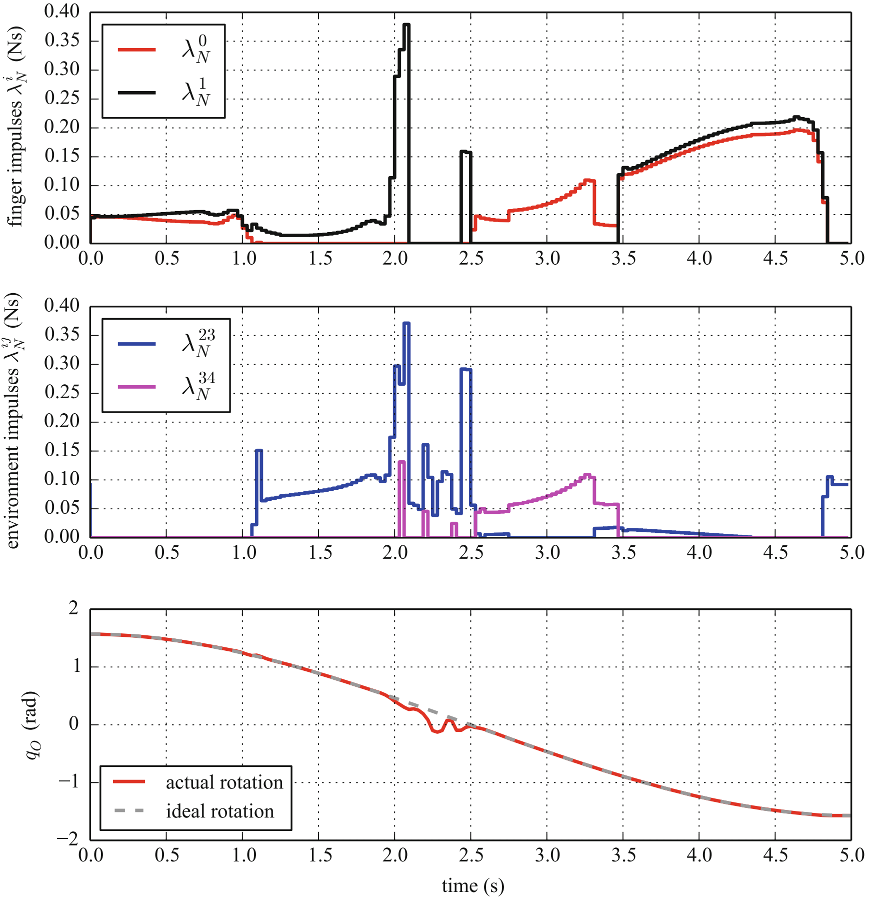


Fig. 6 Motion trajectories of a capsule-shaped object manipulated by an underactuated gripper: contact normal impulses λ_N^0 and λ_N^1 applied by the fingers; contact normal impulses λ_N^{ij} applied by the environment through segment $P_i P_j$; ideal and actual rotation q_O of the object-fixed frame

condition can be violated in general, since the trajectory prescribed from the outset only constitutes a suggested behavior for some components of the system. Regarding the underlying numerical OCP, at each time step $k \in \{0, \dots, N-1\}$, the problem variables have the following dimensions: $q_k \in \mathbb{R}^{4+3}$, $v_{k+1} \in \mathbb{R}^{4+3}$, $\lambda_{N_{k+1}} \in \mathbb{R}^{6+2}$, $\lambda_{T_{k+1}} \in \mathbb{R}^{12+4}$, $\gamma_{k+1} \in \mathbb{R}^{6+2}$. With $N = 160$ discretization intervals ($h = 30\,\text{ms}$) and considering the prescribed boundary conditions on the object/gripper, the problem size is $n = 7375$. To obtain a solution for this example, starting from an initial guess built by repeating the initial condition for all the time steps, Ipopt took 920 iterations, which correspond to ∼25 min on a 2.70 GHz Intel(R) Core(TM) i7-4800MQ CPU with 32 GB of RAM. An animation of the results can be found in part A.2 of the accompanying video [17].

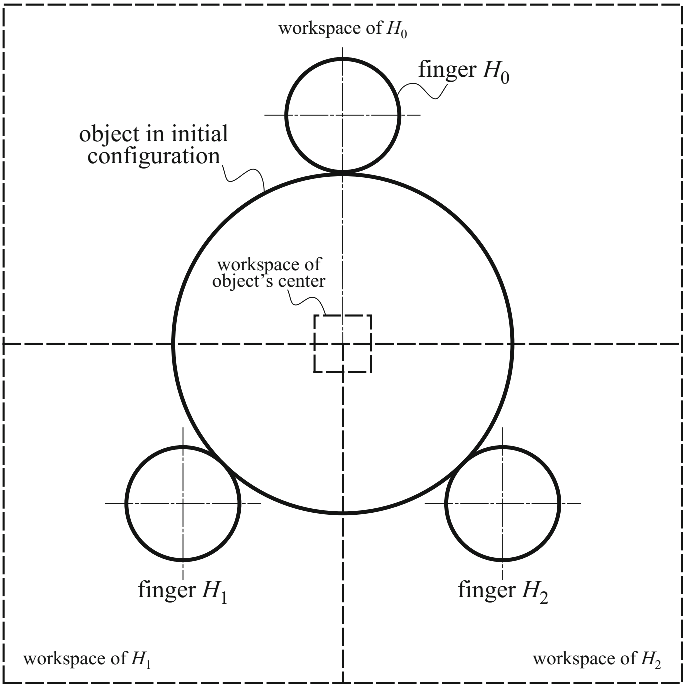


Fig. 7 Dexterous manipulation scenario: the object must find itself rotated by 360° at the final time $T = 6\,\mathrm{s}$

6.2 Dexterous Manipulation

With reference to Fig. 7, a circular object, starting from a configuration where it is held in equilibrium by three independent fingers in a force-closure grasp, has to find itself rotated by 360° at the end of the planning horizon T, with zero velocity. Since, again, the object itself is passive and each finger has workspace limitations (see Fig. 7), a relatively complex (dexterous) manipulation gait has to be discovered for the fingers. To obtain an in-place manipulation, a box constraint has also been assigned on the position of the object center. In order to obtain a relatively robust manipulation, the constraint described in Sect. 4.1 has been included to guarantee that any two fingers are always in contact. As we want no slipping between the fingers and the object during manipulation, the penalty-based approach has been used (with a coefficient of friction $\mu_s = 1.5$). The resulting optimal trajectories in terms of finger forces are shown in the first two plots of Fig. 8: intermittent contacts due to the discovered manipulation gait can be clearly seen. The third plot shows the object's actual rotation versus a smooth (third-order), suggested rotation trajectory. With $N = 200$ discretization intervals ($h = 30\,\mathrm{ms}$) and accounting for the fixed initial and terminal conditions, the problem size is $n = 12585$. To obtain a solution for this

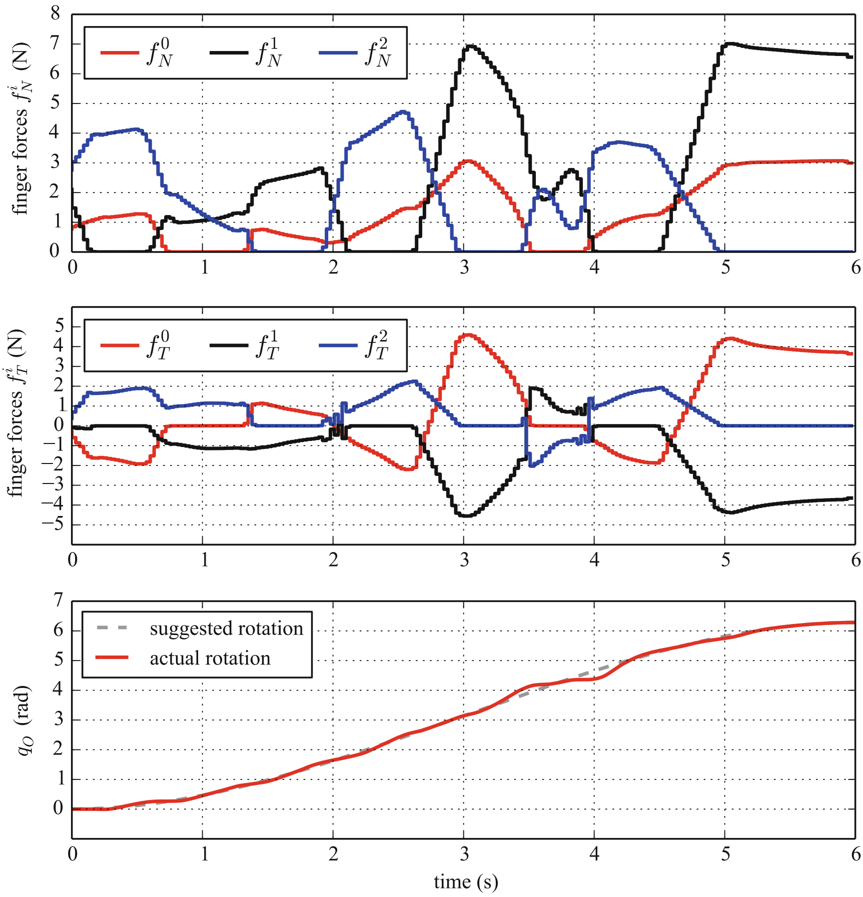


Fig. 8 Trajectories for a circular object manipulated by three independent fingers. Shown are: normal contact forces f_N^i and tangential contact forces f_T^i applied by the fingers; suggested and actual rotation q_O of the object

example, starting from an initial guess built by repeating the initial condition for all the time steps, Ipopt took 1661 iterations, which correspond to ~81 min on a 2.70 GHz Intel(R) Core(TM) i7-4800MQ CPU with 32 GB of RAM. An animation is provided in part B.1 of the accompanying video [17], while part B.2 shows the results obtained by solving the same problem with the velocity-based time-stepping scheme, where sliding between fingers and object is allowed and exploited.

7 Conclusions and Future Work

This paper proposed a computational framework to plan environment-aware manipulation behaviors that do not rely on an a-priori defined sequences of contacts. To this end, we framed the problem as a numerical optimal control one, including contact forces among the optimization variables as a key factor, and we sharpened the algorithmic pipeline by exploiting structural sparsity and leveraging Automatic Differentiation. Two contact models were proposed that best fit manipulation scenarios where sliding primitives need to be avoided or sought, respectively. These proved effective in solving manipulation planning problems where essential interactions with the environment had to be synthesized to accomplish a task (Sect. 6.1). The results presented in Sect. 6.2 demonstrated that the very same method is able to perform successfully in discovering non-trivial gaits also in dexterous manipulation tasks. Current research is devoted to extending the method to 3D scenarios, the major thrust being the synthesis of EC-exploiting, whole-body manipulation strategies for humanoid platforms. Injection of motion primitives/synergies into the model are also being considered, and proper model scaling and tuning of IPOPT convergence parameters are under way to maximize computational efficiency.

Acknowledgements This work is supported by the grant no. 600918 “PACMAN” - Probabilistic and Compositional Representations of Object for Robotic Manipulation - within the FP7-ICT-2011-9 program “Cognitive Systems”. European Research Council and under the ERC Advanced Grant no. 291166 SoftHands (A Theory of Soft Synergies for a New Generation of Artificial Hands).

References

1. Ambler, A.P., Popplestone, R.J.: Inferring the positions of bodies from specified spatial relationships. Artif. Intell. **6**, 157–174 (1975)
2. Andersson, J.: A General-Purpose Software Framework for Dynamic Optimization. Ph.D. thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium (2013)
3. Betts, J.T.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM (2001)
4. Biegler, L.T., Zavala, V.M.: Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. Comput. Chem. Eng. **33**(3), 575–582 (2009)
5. Bonilla, M., Farnioli, E., Piazza, C., Catalano, M., Grioli, G., Garabini, M., Gabiccini, M., Bicchi, A.: Grasping with soft hands. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids) (2014)
6. Boothroyd, G., Poli, C., Murch, L.E.: Handbook of feeding and orienting techniques for small parts. University of Massachusetts at Amherst, Dept. of Mechanical Engineering, Automation Project (1981)
7. Boyle, I., Rong, Y., Brown, D.C.: A review and analysis of current computer-aided fixture design approaches. Robot. Comput. Integr. Manuf. **27**(1), 1–12 (2011)
8. Cohen, B., Phillips, M., Likhachev, M.: Planning single-arm manipulations with n-arm robots. In: Robotics Science and Systems (RSS) (2014)

9. Dafle, N.C., Rodriguez, A., Paolini, R., Tang, B., Srinivasa, S.S., Erdmann, M., Mason, M.T., Lundberg, I., Staab, H., Fuhlbrigge, T.: Regrasping objects using extrinsic dexterity. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 2560–2560 (2014)
10. Deacon, G.E.: An attempt to raise the level of software abstraction in assembly robotics through an apposite choice of underlying mechatronics. J. Intell. Rob. Syst. **28**(4), 343–399 (2000)
11. De Schutter, J., De Laet, T., Rutgeerts, J., Decré, W., Smits, Ruben, Aertbeliën, Erwin, Claes, Kasper, Bruyninckx, Herman: Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. Int. J. Robot. Res. **26**(5), 433–455 (2007)
12. Diankov, R.: Automated Construction of Robotic Manipulation Programs. Ph.D. thesis, Carnegie Mellon University, Robotics Institute (2010)
13. Do Carmo, M.P.: Differential Geometry of Curves and Surfaces. Prentice Hall, Englewood Cliffs (1976)
14. Dogar, M.R., Srinivasa, S.S.: A planning framework for non-prehensile manipulation under clutter and uncertainty. Autonom. Robots **33**(3), 217–236 (2012)
15. Duff, I.S.: Ma57 - a code for the solution of sparse symmetric definite and indefinite systems. ACM Trans. Math. Softw. (TOMS) **30**(2), 118–144 (2004)
16. Eppner, C., Brock, O.: Planning grasp strategies that exploit environmental constraints. In: IEEE International Conference on Robotics and Automation (ICRA) (2015)
17. Gabiccini, M., Artoni, A., Pannocchia, G., Gillis, J.: ISRR 2015 submission accompanying video (2015). http://youtu.be/ozDwPnS00fI
18. Gebremedhin, A.H., Manne, F., Pothen, A.: What color is your Jacobian? Graph coloring for computing derivatives. SIAM Rev. **47**, 629–705 (2005)
19. Gillis, J., Diehl, M.: Hierarchical seeding for efficient sparsity pattern recovery in automatic differentiation. In: CSC14: The Sixth SIAM Workshop on Combinatorial Scientific Computing (2014)
20. Glassman, E., Tedrake, R.: A quadratic regulator-based heuristic for rapidly exploring state space. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 5021–5028, Anchorage, Alaska, USA, 5 (2010)
21. Glocker, C., Studer, C.: Formulation and preparation for numerical evaluation of linear complementarity systems in dynamics. Multibody Sys. Dyn. **13**, 447–463 (2005)
22. Griewank, A., Walther, A.: Evaluating Derivatives, 2nd edn. SIAM, Philadelphia (2008)
23. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Frontiers in Applied Mathematics, vol. 19. SIAM, Philadelphia (2000)
24. HSL. A collection of fortran codes for large scale scientific computation (2014)
25. Ijspeert, A., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives (2003)
26. Jonschkowski, R, Brock, O: State representation learning in robotics: Using prior knowledge about physical interaction. In: Proceedings of Robotics Science and Systems, Berkeley, USA (2014)
27. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
28. Kazemi, M., Valois, J.-S., Andrew Bagnell, J., Pollard, N.: Human-inspired force compliant grasping primitives. Autonom Robots, pp. 1–17 (2014)
29. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: a survey. Int. J. Robot. Res. (IJRR) **32**(11), 1238–1274 (2013)
30. Koga, Y., Latombe, J.-C.: On multi-arm manipulation planning. In: IEEE International Conference on Robotics and Automation (ICRA) (1994)
31. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: IEEE International Conference on Intelligent Robots and Systems (IROS) (2004)
32. Levine, S., Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In: Neural Information Processing Systems (NIPS) (2014)
33. Levine, S., Wagener, N., Abbeel, P.: Learning contact-rich manipulation skills with guided policy search. In: IEEE International Conference on Robotics and Automation (ICRA) (2015)

34. Lozano-Pérez, T., Mason, M.T., Taylor, R.H.: Automatic synthesis of fine-motion strategies for robots. Int. J. Robot. Res. (IJRR) **3**(1), 3–24 (1984)
35. Mason, M.T.: The mechanics of manipulation. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 544–548 (1985)
36. Miller, A.T., Allen, P.K.: Graspit! a versatile simulator for robotic grasping. IEEE Robot. Autom. Mag. **11**(4), 110–122 (2004)
37. Mordatch, I., Popović, Z., Todorov, E.: Contact-invariant optimization for hand manipulation. In: Eurographics/ACM Symposium on Computer Animation (2012)
38. Mordatch, I., Todorov, E., Popović, Z.: Discovery of complex behaviors through contact-invariant optimization. In: ACM Transactions on Graphics (2012)
39. Mordatch, I., Todorov, E.: Combining the benefits of function approximation and trajectory optimization. In: Proceedings of Robotics Science and Systems, Berkeley, USA (2014)
40. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)
41. Oliphant, T.E.: Python for scientific computing. Comput. Sci. Eng. **9**(3), 10–20 (2007)
42. Perez, A., Platt, R., Konidaris, G., Kaelbling, L., Lozano-Perez, T.: LQR-RRT*: optimal sampling-based motion planning with automatically derived extension heuristics. In: 2012 IEEE International Conference on Robotics and Automation (ICRA) (2012)
43. Peters, J., Schaal, S.: Reinforcement learning of motor skills with policy gradients. Neural Netw. **21**(4), 682–697 (2008)
44. Posa, M., Cantu, C., Tedrake, R.: A direct method for trajectory optimization of rigid bodies through contact. Int. J. Robot. Res. (IJRR) **33**(1), 69–81 (2014)
45. Samson, C., Le Borgne, M., Espiau, B.: Robot Control, The Task Function Approach. Clarendon Press, Oxford (1991)
46. Schultz, G., Mombaur, K.: Modeling and optimal control of human-like running. IEEE/ASME Trans. Mechatron. **15**(5), 783–792 (2010)
47. Stewart, D.E., Trinkle, J.C.: An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. Int. J. Numer. Methods Eng. **39**, 2673–2691 (1996)
48. Stewart, D, Trinkle, J.C.: An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 1, pp. 162–169. IEEE (2000)
49. Tedrake, R., Zhang, T. Seung, H.: Stochastic policy gradient reinforcement learning on a simple 3d biped. In: IEEE International Conference on Intelligent Robots and Systems (IROS) (2004)
50. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006)
51. Wriggers, P.: Computational Contact Mechanics. Springer, Berlin (2006)
52. Xi, W., Remy, C.D.: Optimal gaits and motions for legged robots. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 3259–3265 (2014)

Small and Adrift with Self-Control: Using the Environment to Improve Autonomy

M. Ani Hsieh, Hadi Hajieghrary, Dhanushka Kularatne, Christoffer R. Heckman, Eric Forgoston, Ira B. Schwartz and Philip A. Yecko

1 Introduction

The motions of small vehicles are more significantly impacted by the environments they operate in, allowing the environment to be leveraged to improve vehicle control and autonomy, particularly for aerial and marine vehicles [15]. Consider the example of a small autonomous marine vehicle (AMV) operating in the turbulent ocean. The tightly coupled vehicle and environmental dynamics makes control challenging, but

M. Ani Hsieh (✉) · H. Hajieghrary · D. Kularatne
Scalable Autonomous Systems Lab, Drexel University, Philadelphia, PA 19104, USA
e-mail: mhsieh1@drexel.edu

H. Hajieghrary
e-mail: hadi.hajieghrary@exel.edu

D. Kularatne
e-mail: dnk32@drexel.edu

C.R. Heckman
Autonomous Robotics & Perception Group, University of Colorado, Boulder, CO 80309, USA
e-mail: christoffer.heckman@colorado.edu

E. Forgoston
Department of Mathematical Sciences, Montclair State University, Montclair, NJ 07043, USA
e-mail: eric.forgoston@montclair.edu

I.B. Schwartz
Nonlinear Systems Dynamics Section, Plasma Physics Division, Code 6792, U.S. Naval Research Laboratory, Washington, DC 20375, USA
e-mail: ira.schwartz@nrl.navy.mil

P.A. Yecko
Albert Nerken School of Engineering, The Cooper Union, Newyork city, NY 10003, USA
e-mail: yecko@cooper.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_22

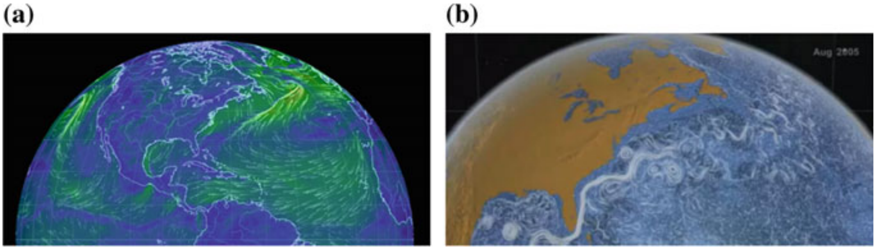


Fig. 1 **a** Visualization of atmospheric currents for January 2015 using data provided by Global Forecasting System, NCEP, National Weather Service, and NOAA. **b** Visualization of ocean surface currents for June 2005 through December 2007 using NASA/JPLs Estimating the Circulation and Climate of the Ocean, Phase II (ECCO2) ocean model

offers nearly limitless environmental forces to be exploited to extend the power budgets of small, resource constrained vehicles.

Recent work in the marine robotics community has verified that AMV motion planning and adaptive sampling strategies can be improved when accounting for dynamics of the fluidic environment [10, 12, 15, 16, 27, 28]. Further progress is hindered by the immense complexity of the atmospheric and/or the ocean dynamics, which involves the interplay of rotation, stratification, complex topography and variable thermal and atmospheric/oceanic forcing, not to mention thousands of biological, chemical, and physical inputs. Theoretical and experimental efforts to model atmospheric and ocean flows have made progress with the help of simpler, so-called "reduced" models, but these models are too idealized and limited in applicability for use in the field. On the other hand, atmosphere and ocean hindcasts, nowcasts, and forecasts provided by National Oceanic and Atmospheric Administration (NOAA) and Naval Coastal Ocean Model (NCOM) [22] and Regional Ocean Model Systems (ROMS) [27] include data assimilated from satellite and field observations. The overall quality of these data is highly dependent on how well a given region of interest is instrumented [25, 26], stymieing attempts to incorporate historical and/or forecasted flows into vehicle motion planning and control strategies.

Fortunately, while geophysical flows are physically complex and naturally stochastic, they also exhibit coherent structure. Figure 1a, b[1] show examples of coherent structures that are easily discerned from the snapshots of atmospheric and ocean surface currents. Knowledge of coherent structures enables prediction of flow properties, including transport, where they are known to play a key role. The Gulf Stream is a prominent example of a mesoscale coherent jet whose heat transport is a critical component of global weather and climate. In addition to large mesoscale eddies and jets, smaller, sub-mesoscale, coherent features such as fronts, often grow from instabilities of the larger scale motions [14]. These features impact both transport and the ocean's *primary production*, and thus storage, of organic matter. Using geophysical

[1] For full animation visit http://earth.nullschool.net/ and http://svs.gsfc.nasa.gov/vis/a000000/a003800/a003827/.

fluid models, details from flows such as the Gulf Stream can be used to diagnose the underlying geophysical fluid dynamics. This, in turn, enables the prediction of various physical, chemical, and biological processes in general geophysical flows. More importantly, predictions based on coherent structures may be exploited more effectively than the detailed predictions offered by state-of-the-art numerical models. Coherent structures can thus provide a basis from which one can construct a vastly reduced order description of the fluid environment.

The challenge of robotic control in a fluidic medium is closely tied to the dual problems of mixing and optimal sensing. G.I. Taylor described and quantified how even simple steady shear flows enhance the mixing of a contaminant in a fluid, a process now known as *Taylor dispersion* [29]. In its simplest form, Taylor dispersion is due to the creation of small scales which locally enhance contaminant gradients and thus optimize diffusion. While achieving better mixing over long times, for shorter time scales, the ability of flows to form structures may also inhibit mixing, as when contaminant is trapped in a coherent vortex.

In this work we examine autonomous sampling of a dispersive event in a geophysical flow. In particular, we consider the problem of finding the source of a contaminant leak in a turbulent flow and discuss how search strategies can be significantly improved from a GFD perspective. The GFD framework is based on a specific class of coherent structures, called Lagrangian coherent structures (LCS). LCS are similar to separatrices that divide a flow into dynamically distinct regions; hyperbolic LCS can be understood as extensions of stable and unstable manifolds to general time-dependent flows [4–6].

In our ongoing work we have tracked LCS using teams of autonomous robots in geophysical fluidic environments, while Inanc et al. showed that time and fuel optimal paths in the ocean can coincide with LCS boundaries [12, 23]. As such, knowledge derived from geophysical fluid dynamics (GFD) can significantly improve the overall quality of vehicle autonomy. For example, planning energy efficient trajectories, maintaining sensors in their desired monitoring regions [7, 10, 16], and enabling computationally tractable and efficient estimation and prediction of surrounding environmental dynamics. We claim that LCS based flow knowledge can be used to improve contaminant tracking strategies and support our claims using geophysical fluid and dispersion models. We conclude with a discussion of major challenges and opportunities ahead.

2 Contaminant Source Localization in Turbulent Mediums

To illustrate these ideas, consider the problem of finding the source of a contaminant or hazardous waste plume in a turbulent medium. Turbulence poses significant challenges for the localization and tracking of material dispersion sources since it breaks up continuous patches of material into seemingly random moving disconnected components. As such, gradient-based search strategies based on chemical concentrations become highly unreliable in turbulent mediums since the mixing dynamics renders any estimation of chemical gradients ineffective [21].

In this section, we formulate the source seeking/plume source localization problem as an information theoretic search strategy. The proposed strategy builds upon [30] where the search strategy consists of making moves that maximize the change in entropy of the posterior distribution of the source location. Similar to [1, 8], we rely on a particle filter representation of the posterior belief distribution to make the strategy more computationally viable in large complex spaces and distributable for mobile sensing teams. The main contribution is extending existing state-of-the-art information theoretic search strategies for robots operating in a turbulent flows.

Background and Assumptions We assume the mean rate of chemical detection at position r resulting from a leakage at r_0 follows a Poisson distribution given by:

$$\mathscr{R}(r|r_0) = \frac{R}{\ln(\frac{\lambda}{a})} e^{\frac{(y_0-y)V}{2D}} K_0\left(\frac{|r-r_0|}{\lambda}\right), \tag{1}$$

where $\lambda = \sqrt{D\tau/\left(1 + V^2\tau/4D\right)}$, R is the emission rate of the source, τ is the finite lifetime of a chemical patch before its concentration falls out of the detectable range, D is the isotropic effective diffusivity of the medium, V is the mean velocity of the current, and $K_0(\cdot)$ is the modified Bessel function of the second kind [30]. The probability of registering the presence of a chemical patch by a sensor depends on the distance and the angle of the sensor from the source, i.e., $(r - r_0)$.

In the search for the source, the set of chemical detection encounters along the search trajectory carries the only available information about the relative location of the source with respect to the robot. Using this information, a robot can construct the probability distribution of the source location using Bayes' rule $P(r_0|\mathscr{T}_t) = P(\mathscr{T}_t|r_0)P(r_0)/\int P(\mathscr{T}_t|r)P(r)dr$. Here, $\mathscr{T}_t$ encapsulates the history of the uncorrelated material encounters along the robot's search trajectory, with $P(\mathscr{T}_t|r_0)$ denoting the probability of experiencing such a history if the source of the dispersion is at r_0. We assume the probability of detecting a material or chemical plume at each step is independent and use Poisson's law to estimate the probability of detecting such a history of the material presence along the search trajectory as:

$$P(\mathscr{T}_t|r_0) = \exp\left(-\int_0^t \mathscr{R}(r(\bar{t})|r_0)\, d\bar{t}\right) \prod_i \mathscr{R}(r(t_i)|r_0), \tag{2}$$

where $r(t)$ is the search trajectory, and $r(t_i)$ is the position of each detection along the trajectory [30]. We note that the assumption of the independence of detections holds since the location of the source is unknown. Furthermore, chemical plumes quickly disintegrate into disconnected patches in turbulent mediums whose dispersion dynamics are highly nonlinear and stochastic. Rather than attempt to model the complex process dynamics, we assume detection events are independent.

We note that (1) provides an observation model and there is no process model beyond what we assume of the environmental dynamics and the propagation of the source in the medium. As such, the current estimate of the belief distribution of the source location, $P_t(r)$, is used to determine the expected number of positive sensor measurements at the new position, i.e., $\mathscr{R}(r(t)|r_0)$. We assume robots move on a grid within the workspace and at each time step can travel an upper bound of n moves. The maximum number of moves on the grid is determined based on the limitations and bounds on the control input and is based on the vehicle's dynamics. The workspace is assumed to be bounded but obstacle-free.

The objective is to localize the source of the dispersion in the workspace. Since the rate of chemical encounter is dependent on the robot's position with respect to the source, the proposed strategy should result in robots maximizing the expected rate of information acquisition on the source's position.

The Single Robot Search Strategy The information gathered through the chemical encounters shapes the probability distribution function that describes the possible source locations, denoted by $P_t(r)$. Thus, it makes sense to consider a search strategy that drives the robot in a direction that promises the steepest decrease in the entropy of this distribution. The expected rate of information gain at each search step is the expected change in the entropy of the estimated field is given by:

$$\mathbf{E}[\Delta S_t(r \mapsto r_j)] = P_t(r_j)[-S_t] + [1 - P_t(r_j)][(1 - \rho(r_j))\Delta S_0 + \rho(r_j)\Delta S_1], \quad (3)$$

where, $S_t = \int P_t(r)\log(P_t(r))dr$ is the Shannon entropy of the estimated field. The first term of (3) corresponds to the change in entropy upon finding the source at the very next step. If the robot successfully localizes the source at the next step, then the change in entropy would be zero and thus end the search.

Using the current belief distribution as the best estimate of the source location, the expected number of positive sensor hits at location r is given by:

$$h(r) = \int P_t(r_0)\mathscr{R}(r|r_0)dr_0 \quad (4)$$

and the probability of a single positive detection follows the Poisson law $\rho(r) = h(r)\exp(-h(r))$. Thus, the second term of (3) accounts for the case when the source is not found at r_j and computes the expected value of the information gain by the robot moving to this new position. At each new location, we assume the robot takes one measurement with its sensor resulting in two possibilities – the robot will have either a positive sensor hit or not. To find the expected value of the change in the utility function, i.e., the entropy, (3) calculates the change in entropy for each possible move and each possible sensor outcome after the move.

The belief distribution for the source location $P_t(r)$ is maintained over all possible source positions. At each step, the robot chooses to move in a direction that enables it to acquire more information and decreases the uncertainty of the source position estimate. Storing and representing the belief distribution becomes computationally challenging especially when the search space spans large physical scales and/or contains complex geometry. This is especially true if robots rely on a fine grid map to calculate the log-likelihood of the belief distribution. Furthermore, since the search hypotheses are often spread across the workspace, closed form descriptions for the belief distribution may not be accessible, especially in geometrically complex spaces. Thus, we employ a particle filter approach to the representation of the belief distribution with limited numbers of randomly drawn particles.

In this work, we assume each robot stores the estimated belief distribution of the source location, $P_t(r)$, using a manageable number of particles, $\{\hat{r}_i, \omega_i\}$, with $\hat{r}_i$ representing the hypothesis for the state (position) and ω_i representing the corresponding weight (probability) of hypothesis i. The probability mass function represented by these set of particles is mathematically equivalent to the sum of the weighted spatial impulse functions [2]:

$$\hat{P}_r(t) \approx \sum_i \omega_i \delta(\hat{r}_i - r), \tag{5}$$

where $\hat{r}_i$ is a hypothesis that survived the re-sampling procedure of the previous step, and the weights, ω_i, of the particles are modified as follows:

$$\omega_i(t) = \omega_i(t-1)e^{-\left(R(r(t)|r_0)\right)}\left(R(r(t)|r_0)\right)^{hit} \tag{6}$$

with $hit = 1$ in the case of a positive detection and $hit = 0$ otherwise. To calculate the entropy of the particle representation of the belief distribution, we use the approach presented in [8] where $S \approx -\sum_{k=1}^{N} w_{(t-1),k}^{(i)} \log\left(w_{(t-1),k}^{(i)}\right)$.

A robot's control decision is determined by the expected change in entropy of the source position estimate. At every time step, the expected information gain from an observation at the next probable robot position is calculated. Since robots are constrained to a maximum of n moves on the grid, they can quantify the expected information gain on the source's position as it moves. If no particles are within the robot's set of reachable points on the grid, then any position that is n moves away from the robot's current position can be chosen as the next step.

Two approaches are used to represent the information gathered during the search. The first one assumes the robot has a limited field of view with no knowledge of the search area. The particles are placed in the robot's coordinate frame, and the control decision is to move in the direction the robot expects to acquire more information. Although the source location may not initially be within the robot's field of view, we assume its sensing range is large enough to provide a good enough measurement for it to determine a direction to move in. The second approach assumes the robot knows the boundary of the search area and can localize itself within the space. Under this scenario, the particles used to estimate the source positions are spread over the entire workspace. Figure 2 depicts the two representations.

It is important to note that we assume robots can measure the flow direction at its current position. This is important for the robot to discern the direction the material plumes are coming from. We note that the weight update given by (6) is different from most particle filter implementations. This is because when localizing the source

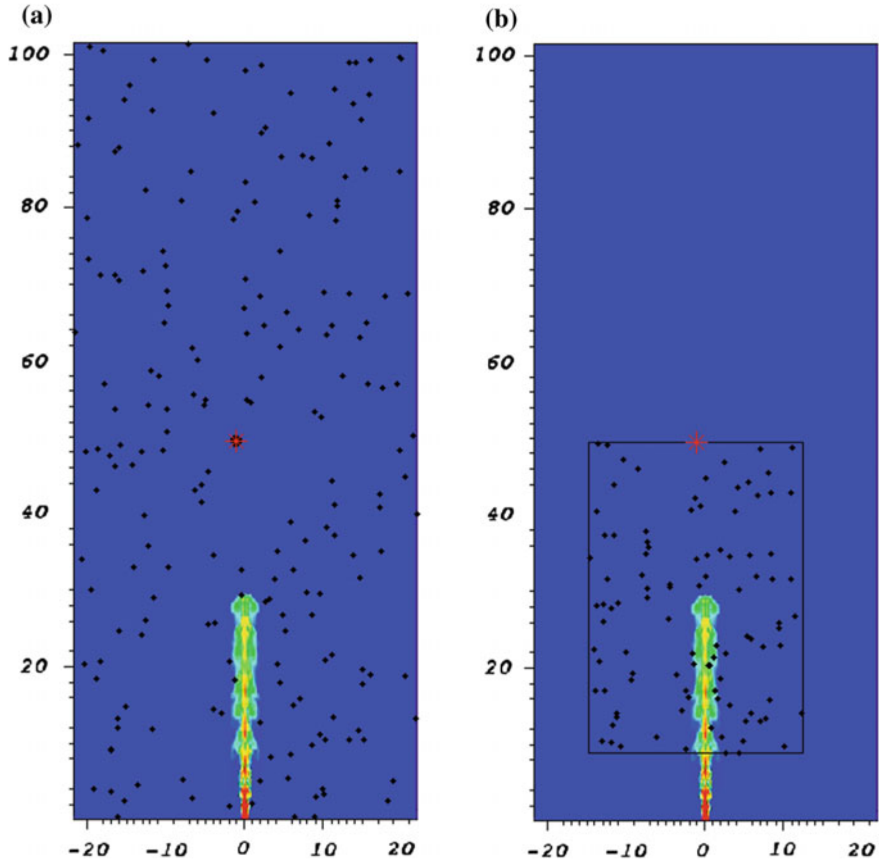


Fig. 2 The *particles* representing the hypotheses for the source position. The *background* shows the gas volume concentration in the water due to an oil spill. *Blue* denotes low concentration and *red* denotes high concentration. Details of the plume simulation can be found in [3]

of a dispersion, re-sampling serves the role of integrating past information into the current estimate of the source position. Therefore, one must take the likelihood of the detection history, (2), into account during the update and eliminate the less probable hypotheses when re-sampling. This is the fundamental difference between the proposed strategy and existing information theoretic strategies for aerial and ground vehicles operating in static environments. The single robot search strategy is summarized in Algorithm 1.

The Multi-Robot Search Strategy To speed up the search strategy, we extend the proposed single robot search strategy to a robot team. This effectively increases the chances of positive encounters and decreases the expected time needed to localize

```
input  : CURRENT estimate of source position belief distribution
output: UPDATED estimate of source position belief distribution
for All particles within the robot's reachable set do
    Update particle weights if the robot moves to the position of the particle, and;
    (1) The robot detects a plume in the new position;
    (2) The robot does not detect a plume in the new position;
    Calculate entropy of the particle filter for (1) and (2);
    Calculate expected reduction in entropy if robot moves to each particle position;
end
Move to the location of the particle with steepest expected reduction in entropy;
Obtain sensor reading and compute new particle weights for the samples;
Re-sample;
```

Algorithm 1: Single robot search strategy

the source. To achieve this, we assume robots can communicate with one another and thus the team can build a shared belief distribution of the source position. While communications can be severely bandwidth limited in a fluidic medium, e.g., underwater environments, the amount of data that must be communicated is low. Robots are only required to exchange their position information when they have detected a material plume at their current location. Since robots initialize with the same belief distribution, each robot moves in a direction that reduces the entropy of this common belief distribution. Such a coordination strategy ensures every individual in the team searches for a single source location. To take into account the impact of robot motion uncertainties on particle positions, we represent each particle position with a Gaussian distribution and thus employ a Gaussian Mixture Model to represent the probability distribution of the source given by (5). As such, we employ the steps described in [1, 8, 11] to calculate the information utility function:

$$S \approx -\int_{\theta} \left\{ \sum_{k=1}^{N} w_{(t-1),k}^{(i)} p(\theta_t | \theta_{t-1} = \hat{\theta}_{(t-1),k}^{(i)}) \log \left(\sum_{k=1}^{N} w_{(t-1),k}^{(i)} p(\theta_t | \theta_{t-1} = \hat{\theta}_{(t-1),k}^{(i)}) \right) \right\},$$

where $p(\theta_t | \theta_{t-1} = \hat{\theta}_{(t-1),k}^{(i)})$ is the probability distribution over the possible position of the ith particle with respect to the agent after it moves and is a Gaussian.

Simulation results validating the proposed search strategy for a team of three robots are shown Fig. 3. For a video of the simulation, we refer the reader to https://youtu.be/5maQPeEcyf8. The robots are shown localizing the source of a 2D multi-scale simulation of the Deepwater Horizon oil spill [3]. The entropy of the belief distribution for the source position at every time step is shown in Fig. 4a. We note that the temporary increase in entropy at 60^{th} time step corresponds to one of the robots loosing track of the plume.

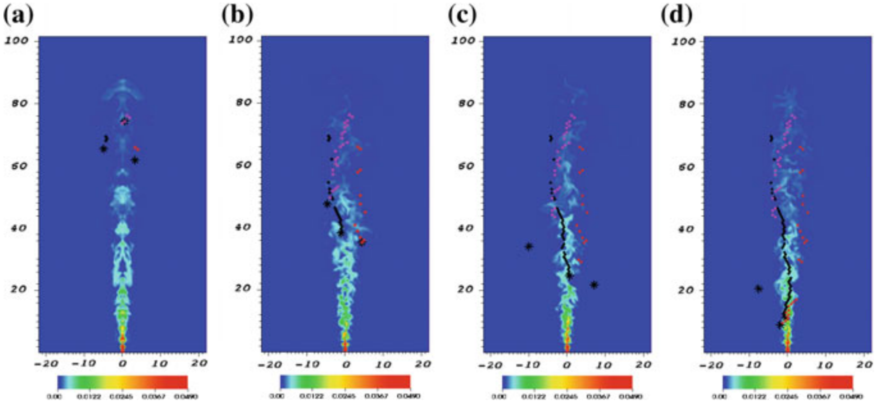


Fig. 3 A team of three robots executing the proposed collaborative search strategy to localize the source of an oil spill. The *background* shows the gas volume concentration in the water due to the spill. The robots are denoted by *black* ∗. The *red*, *magenta*, and *black dots* denote the robot trajectories. For more details on the plume simulation, we refer the interested reader to [3]

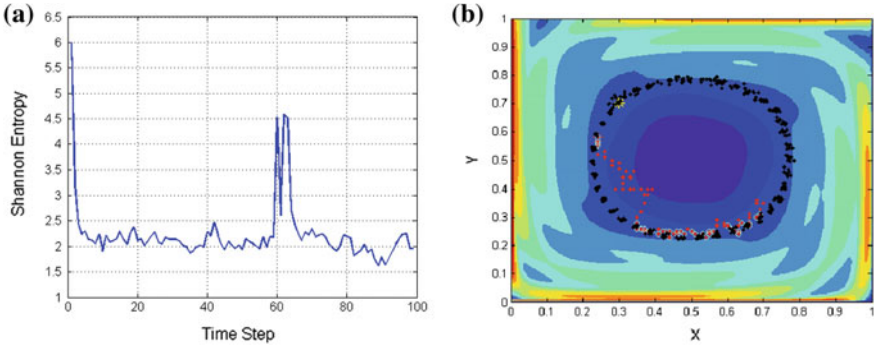


Fig. 4 **a** The particles representing the hypotheses over the position of the source. **b** Single agent search using Algorithm 1 in a gyre flow. The *background* shows the FTLE field of the fluidic environment with LCS boundaries denoted in *red*. The source of the spill is denoted by the *yellow* ∗ and the *black dots* denote contaminants emanating from the *yellow* ∗. The *red* ∗ denotes the robot's current position, the *red dots* denote the robot's search trajectory. The *cyan circles* denote portions of the robot's trajectory where it detected the contaminants

3 Lagrangian Coherent Structures

While the proposed information theoretic search strategy described in the previous section is able to find and localize the source of a spill in a turbulent medium, the strategy can be significantly improved with GFD knowledge of the environmental dynamics. Coherent structures exhibited in GFD flows provide reduced-order description of the complex fluid environment and enable the estimation of the underlying geophysical fluid dynamics. In particular, Lagrangian coherent structures are important since they quantify transport and control the stretching and folding that underpins kinematic mixing.

Lagrangian coherent structures are material lines that organize fluid-flow transport and as mentioned may be viewed as the extensions of stable and unstable manifolds to general time-dependent systems [6]. In two-dimensional (2D) flows, LCS are one-dimensional separating boundaries analogous to ridges defined by local maximum instability, and can be quantified by local measures of Finite-Time Lyapunov Exponents (FTLE) [5, 24]. In this section, we briefly explain the computation of FTLE fields and the identification of LCS boundaries via local FTLE measures. We limit our discussions to 2D planar flows, however all concepts discussed in this section are readily extended to higher dimensions.

Consider a 2D flow field given by

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}, t) \tag{7}$$

where $\mathbf{x} = [x, y]^*$ gives the position in the plane and * denotes the transpose of the vector. Such a continuous time dynamical system has quantities, known as Lyapunov exponents, which are associated with the trajectory of the system in an infinite time limit. The Lyapunov exponents measure the growth rates of the linearized dynamics about the trajectory. To find the FTLE, one computes the Lyapunov exponents on a restricted finite time interval. For each initial condition, the exponents provide a measure of its sensitivity to small perturbations. Therefore, the FTLE is a measure of the local sensitivity to initial data. The position of a fluid particle advected by the flow field given by (7), is a function of time t, the starting point of the particle $\mathbf{x_0}$ and starting time t_0, i.e., $\mathbf{x} = \mathbf{x}(t; \mathbf{x_0}, t_0)$. Using the notation by Shadden et al. [24], the solution to the dynamical system given in (7) can be viewed as a flow map which takes points from their initial position $\mathbf{x_0}$ at time t_0 to their position at time t. This map, denoted by $\phi_{t_0}^{t}$, satisfies $\phi_{t_0}^{t}(\mathbf{x_0}) = \mathbf{x}(t; \mathbf{x_0}, t_0)$, and has the properties: $\phi_{t_0}^{t_0}(\mathbf{x}) = \mathbf{x}$ and $\phi_{t_0}^{s+t}(\mathbf{x}) = \phi_{s}^{s+t}(\phi_{t_0}^{s}(\mathbf{x}))$.

The FTLE with a finite integration time interval T, associated with a point $\mathbf{x}$ at time t_0 is given by,

$$\sigma_{t_0}^{T}(\mathbf{x}) = \frac{1}{|T|} \ln \sqrt{\lambda_{max}\,(\varDelta)} \tag{8}$$

where $\lambda_{max}\,(\varDelta)$ is the maximum eigenvalue of the finite-time version of the Cauchy-Green deformation tensor $\varDelta$, given by,

$$\varDelta = \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}}^{*} \frac{d\phi_{t_0}^{t_0+T}(\mathbf{x})}{d\mathbf{x}}. \tag{9}$$

The value of $\varDelta$ is computed numerically by discretizing the domain into a regular grid and computing the trajectories of each point and its immediate neighbors in the grid from time t_0 to $t_0 + T$. For each point in the grid, the trajectories are computed by numerically integrating (7) from t_0 to $t_0 + T$. The FTLE value gives a measure of the maximum expansion of two initially nearby particles when they are advected

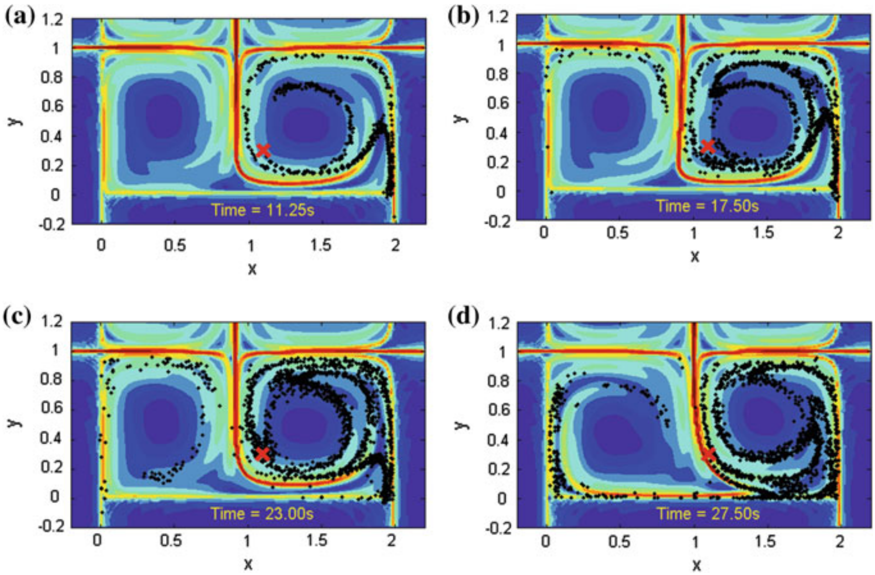


Fig. 5 Simulation of a spill in a time-varying wind-driven *double-gyre* flow. The *background* denotes the FTLE field for flow and the *red x* denotes the source position of the spill. The *black particles* denotes particulates emanating from the source

by the flow. Therefore, particles initiated on opposite sides of an LCS will have much higher FTLE values than their neighbors since LCS are essentially boundaries between two dynamically distinct regions in the flow.

By calculating the FTLE values for the entire flow field, it is possible to identify the LCS boundaries by tracing out regions with the highest FTLE values. As such, LCS are equivalent to ridges in the FTLE field with maximal FTLE values as defined by Shadden et al. [24]. The *forward-time* FTLE field calculated by advecting fluid particles forward in time ($T > 0$), reveals repelling LCS which are analogous to the stable manifolds of saddle points in a time independent flow field. Conversely, the *backward-time* FTLE field ($T < 0$) reveals attracting LCS which are analogous to unstable manifolds of a time independent flow field.

While one can easily extend the source seeking strategies described in Sect. 2 to single gyre-like flows as shown in Fig. 4b, knowing the LCS boundaries can significantly improve search strategies when operating in complex environments like those shown in Fig. 1a, b. Figure 5 shows a simulation of the dispersion of particulates in a time-varying wind-driven double-gyre flow [31] with the FTLE field shown in the background. The LCS boundaries are denoted by red and the center vertical boundary oscillates about $x = 1$. From the simulations, we note that: (1) LCS boundaries behave as basin boundaries and thus fluid from opposing sides of the boundary do not mix; (2) in the presence of noise,[2] particulates can cross the

[2] Noise can arise from uncertainty in model parameters and/or measurement noise.

LCS boundaries and thus LCS denotes regions in the flow field where more escape events occur [4]; and (3) knowledge of LCS locations can improve the existing search strategy since this is akin to having a map of the workspace.

However, one downside to finding attracting and repelling LCS by computing the FTLE fields is due to the fact that one needs global velocity field information for the region of interest. It is often the case when operating in the ocean that this information is sparse. Therefore, it would be useful to find the LCS in an alternate way. To this end, we have developed a novel method that allows for collaborative robotic tracking of LCS. In particular, the robots perform the tracking based on local velocity measurements, thus ameliorating the need for global information. This collaborative LCS tracking is described in the following section.

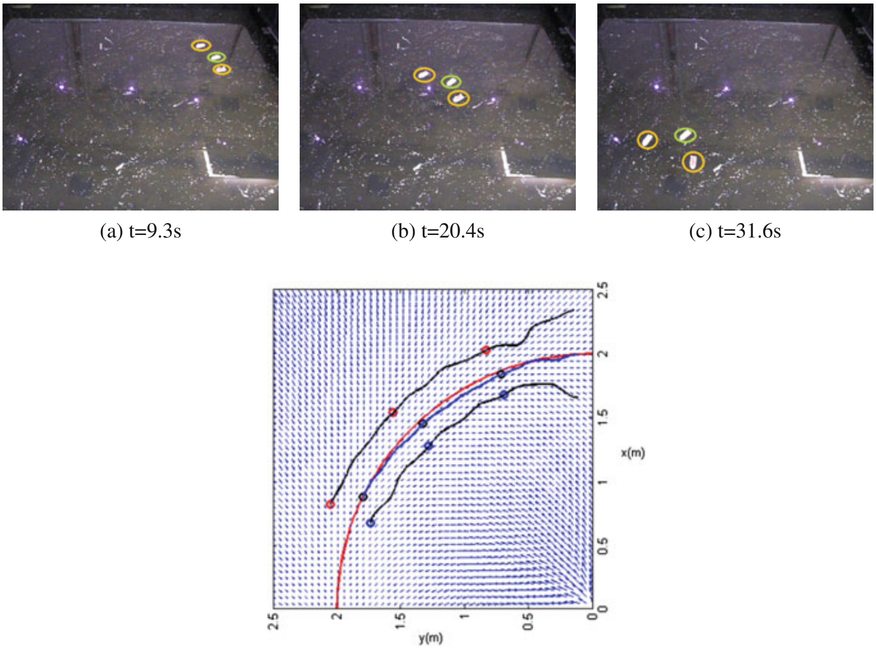
4 Collaborative LCS Tracking

The tracking of coherent structures in fluids is challenging since they are global structures that are generally unstable and time-dependent. We briefly describe existing and on-going work in developing collaborative strategies for teams of robots to track the stable and unstable manifolds and LCS boundaries in 2D flows. It is important to note that tracking repelling LCS is achieved without relying on explicit computations of FTLE values. In fact, the identification of repelling LCS boundaries requires the computation of forward FTLEs, making real-time FTLE-based tracking of LCS boundaries extremely challenging. In contrast to this, tracking attracting LCS can be performed using backward FTLE fields calculated using the local velocity data acquired by the robots.

Repelling LCS and Stable Manifolds The tracking of repelling LCS boundaries or stable manifolds in flows relies on robots maintaining a boundary straddling formation while collecting local measurements of the flow field. Since LCS correspond to regions in the flow field with extremal velocities, LCS tracking is achieved by fusing data obtained on opposite sides of the boundary to identify the location of the extremal velocity. The strategy was first proposed for a team of three robots where the center robot is responsible for locating the boundary using the flow measurements provided by the team. The remaining robots would then maintain a saddle straddling formation, i.e., remain on opposite sides of the boundary, at all times. The strategy has been extensively validated using analytical models, experimental data, and actual ocean data [9, 17, 18] and extended for larger team sizes [19].

It is important to note that these strategies achieve tracking of *global* fluidic features, i.e., the LCS, using only *local* measurements of the flow field and an initial estimate of the location of the repelling LCS.

Attracting LCS and Unstable Manifolds Different from repelling LCS, attracting LCS are quantified by maximal backward FTLE measures. As such, collaborative tracking of attracting LCS or unstable manifolds can be achieved through on-board calculation of local FLTE fields using previously acquired flow velocity data. In this


(a) t=9.3s

(b) t=20.4s

(c) t=31.6s

(d)

Fig. 6 **a**–**c** Actual ASV tracking experiment. **d** ASVs trajectories for the experiment in (**a**)–(**c**). The *red line* is the actual boundary and the *blue line* is the trajectory of the center vehicle

case, the tracking strategy utilizes the FTLE field along with instantaneous local flow field measurements to resolve the attracting LCS boundary. Similar to tracking repelling LCS, agent-level control policies leveraging underlying flow dynamics were developed to maintain the formation of the team as they track the attracting boundary. The formation control strategy ensures vehicles do not collide with one another while maintaining the necessary boundary straddling formation.

The attracting LCS tracking strategy was also validated using a combination of analytical models and experimental flow tanks using micro autonomous surface vehicles (ASVs) [13]. Figure 6 shows an experiment using three real and four virtual ASVs to track a simulated static flow field. The three ASVs were initially arranged in a saddle straddling formation with the center ASV tasked with tracking the boundary. The virtual agents were placed at the four corners of the grid.

Collaborative tracking of coherent structures allows one to gain knowledge of the geophysical flow of interest. This knowledge can be used to improve or optimize a variety of sensing and control objectives. For example, in previous work we have demonstrated how GFD-based knowledge allows for an increase in loitering time of vehicles operating in the ocean [4, 7, 10, 16]. Another example involves the localization of a contaminant source as described previously.

5 Conclusions and Future Outlook

In this work, we showed how GFD-derived knowledge can significantly improve the autonomy of the vehicles that operate within them. Coherent structures have the potential to provide computationally efficient forecasts of current and wind patterns since they enable a much lower order description of the environmental dynamics. The ability to better quantify transport behaviors in natural fluid environments will enable the synthesis of energy-efficient motion planning and control strategies, the detection and tracking of contaminant and hazardous waste dispersions, and more effective allocation of mobile sensing resources for search and rescue operations. This makes intuitive sense since this is akin to planning and motion control of autonomous vehicles using a suitable map of the environment.

Different from traditional maps, most features of interest in flows are unstable and non-stationary. While this renders the problem of leveraging dynamic transport controlling structures for improved underwater vehicle autonomy highly challenging, it also opens up new opportunities in planning, control, and perception in these highly dynamic and uncertain environments. Some specific directions for future investigations include how do we construct, maintain, and update such a fluidic map? Since LCS boundaries exist at various length scales and are often seasonal, how does a vehicle leverage the structures for energy efficient monitoring and navigation? Is it possible to use LCS information to identify unsafe regions in the fluidic environment where vehicles may be trapped by the environmental dynamics? Can we manipulate the fluid for fluid-mediated underwater manipulation of objects?

Beyond these challenging problems is the extension of all the previously discussed tracking, planning, control, search, etc. problems to the fully 3D environment. One major challenge associated with autonomous vehicles operating at depth are the issues involved with communication. It is known that communication time delay can destabilize formations of vehicle groups [20]. But one also must consider communication drop-outs and even the complete loss of communication especially when operating in the ocean. Overcoming the difficulties of operating in a 3D environment will allow for greatly improved environmental monitoring and forecasting.

In this work, we presented information theoretic search strategies for localizing contaminant sources in turbulent flows. We also presented on-going work in distributed sensing of geophysical fluid dynamics. Much of this work has been motivated by the insight that the environment may be effectively exploited for vehicle control and autonomy especially when the vehicle and environmental dynamics are tightly coupled. By looking at the changing environment through a geophysical perspective, there are significant GFD features that can be leveraged for predicting and estimating the environmental dynamics. The challenge lies in overcoming the theoretical and technological challenges needed to robustly and autonomously collect, process, and interpret data about the geophysical flows.

Acknowledgements MAH, HH, and DK are supported by ONR Award Nos. N000141211019 and N000141310731 and National Science Foundation (NSF) grant IIS-1253917. EF and PAY are supported by National Science Foundation (NSF) grant DMS-1418956. IBS is supported by

ONR contract No. N0001412WX2003 and NRL 6.1 program contract No. N0001412WX30002. We would like to thank Alex Fabregat Tomas (CUNY) and Andrew Poje (CUNY) in providing the plume data.

References

1. Charrow, B., Michael, N., Kumar, V.: Cooperative Multi-robot Estimation and Control for Radio Source Localization. Springer International Publishing, New York (2013)
2. Crisan, D., Doucet, A.: A survey of convergence results on particle filtering methods for practitioners. IEEE Trans. Signal Process. **50**(3), 736–746 (2002)
3. Fabregat, A., Dewar, W.K., Ozgokmen, T.M., Poje, A.C., Wienders, N.: Numerical simulations of turbulent thermal, bubble and hybrid plumes. Ocean Model. **90**, 16–28 (2015)
4. Forgoston, E., Billings, L., Yecko, P., Schwartz, I.B.: Set-based corral control in stochastic dynamical systems: making almost invariant sets more invariant. Chaos **21**, 013116 (2011)
5. Haller, G.: A variational theory of hyperbolic Lagrangian coherent structures. Phys. D **240**, 574–598 (2011)
6. Haller, G., Yuan, G.: Lagrangian coherent structures and mixing in two-dimensional turbulence. Phys. D **147**, 352–370 (2000)
7. Heckman, C.R., Hsieh, M.A., Schwartz, I.B.: Controlling basin breakout for robots operating in uncertain flow environments. In: International Symposium on Experimental Robotics (ISER 2014), Marrakech/Essaouira, Morocco (2014)
8. Hoffmann, G., Tomlin, C.: Mobile sensor network control using mutual information methods and particle filters. IEEE Trans. Autom. Control **55**(1), 32–47 (2010)
9. Hsieh, M.A., Forgoston, E., Mather, T.W., Schwartz, I.: Robotic manifold tracking of coherent structures in flows. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA2012), Minneapolis, MN USA (2012)
10. Hsieh, M.A., Mallory, K., Schwartz, I.B.: Distributed allocation of mobile sensing agents in geophysical flows. In: Proceedings of the 2014 American Controls Conference, Portland, OR (2014)
11. Huber, M., Bailey, T., Durrant-Whyte, H., Hanebeck, U.: On entropy approximation for gaussian mixture random vectors. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008, pp. 181–188 (2008). doi:10.1109/MFI.2008.4648062
12. Inanc, T., Shadden, S., Marsden, J.: Optimal trajectory generation in ocean flows. In: Proceedings of the American Control Conference, pp 674–679 (2005)
13. Kularatne, D., Hsieh, A.: Tracking attracting lagrangian coherent structures in flows. In: Proceedings of Robotics: Science and Systems, Rome, Italy (2015)
14. Levy, M., Ferrari, R., Franks, P.J.S., Martin, A.P., Riviere, P.: Bringing physics to life at the submesoscale. Geophys. Res. Lett. **39**(14) (2012)
15. Lolla, T., Ueckermann, M.P., Haley, P., Lermusiaux, P.F.J.: Path planning in time dependent flow fields using level set methods. In: Proceedings of IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA (2012)
16. Mallory, K., Hsieh, M.A., Forgoston, E., Schwartz, I.B.: Distributed allocation of mobile sensing swarms in gyre flows. Nonlin. Process. Geophys. **20**(5), 657–668 (2013)
17. Michini, M., Hsieh, M.A., Forgoston, E., Schwartz, I.B.: Experimental validation of robotic manifold tracking in gyre-like flows. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2014, Chicago, IL, USA (2014)
18. Michini, M., Hsieh, M.A., Forgoston, E., Schwartz, I.B.: Robotic tracking of coherent structures in flows. IEEE Trans. Robot. **30**(3), 593–603 (2014)
19. Michini, M., Rastgoftar, H., Hsieh, M.A., Jayasuriya, S.: Distributed formation control for collaborative tracking of manifolds in flows. In: Proceedings of the 2014 American Control Conference (ACC 2014), Portland, OR (2014)

20. Mier-y Teran-Romero, L., Forgoston, E., Schwartz, I.: Coherent pattern prediction in swarms of delay-coupled agents. IEEE Trans. Robot. **28**(5), 1034–1044 (2012)
21. Russell, R.: Locating underground chemical sources by tracking chemical gradients in 3 dimensions. In: Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004), vol. 1, pp 325–330 (2004). doi:10.1109/IROS.2004.1389372
22. SCRIPPS: Naitonal HF RADAR network - surface currents (2014). URL http://cordc.ucsd.edu/projects/mapping/maps/
23. Senatore, C., Ross, S.: Fuel-efficient navigation in complex flows. Am. Control Conf. **2008**, 1244–1248 (2008)
24. Shadden, S.C., Lekien, F., Marsden, J.E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. Phys. D Nonlin. Phenom. **212**(3–4), 271–304 (2005)
25. Shchepetkin, A., McWilliams, J.: Quasi-monotone advection schemes based on explicit locally adaptive dissipation. Mon. Weather Rev. **126**, 1541–1580 (1998)
26. Shchepetkin, A.F., McWilliams, J.C.: The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. Ocean Model. **9**, 347–404 (2005)
27. Smith, R.N., Chao, Y., Li, P.P., Caron, D.A., Jones, B.H., Sukhatme, G.S.: Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model. Int. J. Robot. Res. **29**(12), 1475–1497 (2010)
28. Smith, R., Kelly, J., Sukhatme, G.: Towards improving mission execution for autonomous gliders with an ocean model and kalman filter. In: Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN (2012)
29. Taylor, G.: Dispersion of soluble matter in solvent flowing slowly through a tube. Proc. R Soc. Lond. A **A219**, 186–203 (1953)
30. Vergassola, M., Villermaux, E., Shraiman, B.I.: Infotaxis as a strategy for searching without gradients. Nature **445**, 406–409 (2007)
31. Veronis, G.: Wind-driven ocean circulation, part I and part II. Deep-Sea Res. **13**, 31 (1966)

Unifying Robot Trajectory Tracking with Control Contraction Metrics

Ian R. Manchester, Justin Z. Tang and Jean-Jacques E. Slotine

1 Introduction

Design of tracking controllers for robotic systems is a classical problem, as old as robotics itself. Most industrial robots are fully-actuated, i.e. for each configuration coordinate (degree of freedom) there is an associated actuator. Additionally, many walking robots are controlled so as to remain in a fully-actuated state, i.e. at least one flat foot firmly planted on the ground (see, e.g., [1] and references therein). For fully-actuated systems, there are several generally applicable methods for tracking control: independent joint control using PID, computed torque, sliding control, and passivity/energy-based methods, to name but a few [2, 3].

For underactuated systems, i.e. those with fewer independent control inputs than dynamical degrees of freedom, the situation is quite different. Many robotic systems of current interest are underactuated, including quadrotors (e.g. [4]), dynamic walking robots (e.g. [5]), and lightweight robots with flexible links (e.g. [6]).

Several nonlinear control approaches have been developed for underactuated systems [7, 8], but there is currently no generally applicable method that stabilizes underactuated systems over their maximal basin of attraction.

One recent approach is to design a controller based on a local linearization, e.g. using the linear quadratic regulator (LQR) [9], and then to verify a basin of attraction using computational methods [10]. These basins can then be pieced together to

I.R. Manchester (✉) · J.Z. Tang
Australian Centre for Field Robotics & School of Aerospace, Mechanical and Mechatronic Engineering, University of Sydney, Camperdown, Australia
e-mail: ian.manchester@sydney.edu.au

J.Z. Tang
e-mail: j.tang@acfr.usyd.edu.au

J.-J.E. Slotine
Nonlinear Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: jjs@mit.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_23

extend the region of validity of the controller [11, 12]. Another recent approach uses dual representations of Lyapunov functions to compute an outer-approximation the maximal basin of attraction [13, 14].

In this paper, we explore the newly-developed concept of a "control contraction metric" (CCM) as a unifying framework for classical physics-based methods and emerging optimization-based methods for control of nonlinear robots. The idea of a CCM was introduced in [15, 16], extending the analysis method of [17].

We show that CCMs can be thought of as unifying the above, seemingly unconnected, approaches. For fully actuated systems, particular metrics can be chosen so that the CCM design reduces exactly to sliding and energy-based designs. On the other hand, the CCM approach extends that of [10–12] in some key ways: firstly, it generates controllers that stabilize *every* feasible trajectory in a region, not just a single target that must be known a priori. Secondly, the control synthesis method can be represented as a convex optimization problem.

Convexity of conditions has benefits beyond tractability of numerical optimization. For example, one can can mix-and-match different specifications such as local optimality and global stability [16] – providing an interesting alternative to switching between global and local controllers (e.g. [18, 19]) – or build motions by combining motor primitives with contracting or limit cycle behaviour [20, 21].

2 System Description and Problem Setup

Our development is motivated by controlling a mechanical system with configuration $q \in \mathscr{Q}$, an n-dimensional smooth manifold (e.g. $\mathbb{R}^n$), and dynamics derived from the Euler-Lagrange equations and expressed in the standard "manipulator" form [2, 3]:

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = Ru. \tag{1}$$

Here $H(q)$ is the mass/inertia matrix, $C(q, \dot{q})\dot{q}$ contains Coriolis and centrifugal terms, and $g(q)$ is the gradient of a potential field. The control inputs are forces/torques $u \in \mathbb{R}^m$. For a fully-actuated robot, $m = n$ and R is a full-rank square matrix, while for an underactuated robot R is $n \times m$ with $m < n$. Recall that (1) can be constructed so that $\dot{H} - 2C$ is skew-symmetric [2, p. 400].

We will illustrate our results on three classic systems: the single and double pendulum (both fully actuated) and the underactuated cart-pole system. These systems, their configuration coordinates, and relevant parameters are depicted in Fig. 1.

For our purposes it will frequently be convenient to rewrite the dynamical equations (1) as a $2n$-dimensional set of first-order differential equations:

$$x := \begin{bmatrix} q \\ \dot{q} \end{bmatrix}, \quad \dot{x} = f(x) + B(x)u, \tag{2}$$

where

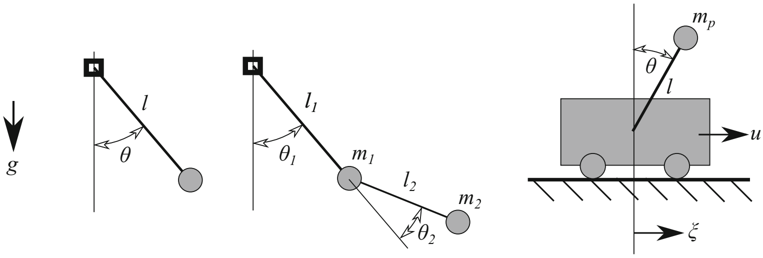


Fig. 1 The single pendulum, double pendulum, and cart-pole systems

$$f(x) = \begin{bmatrix} \dot{q} \\ H(q)^{-1}(-C(q,\dot{q})\dot{q} - g(q)) \end{bmatrix}, \quad B(x) = \begin{bmatrix} 0 \\ H(q)^{-1}R \end{bmatrix}. \tag{3}$$

To define a nonlinear control problem precisely, we must be specific about the type of convergence desired. If the configuration space is $\mathbb{R}^n$, then a feasible solution $x^\star(t), u^\star(t)$ of (2) on $t \in [0, \infty)$ is said to be globally exponentially stabilized by a feedback controller $u = k(x, t)$ if positive constants K, λ exist such that $|x(t) - x^\star(t)| \leq Ke^{-\lambda t}|x(0) - x^\star(0)|$ for all $x(0)$, where $x(t)$ is the solution of the closed-loop system. The constant λ is referred to as the *rate* of convergence. For nonlinear manifolds, similar definitions can be constructed using an appropriate distance metric.

Global exponential stability is defined with respect to a *particular* solution $x^\star$, $u^\star$. For robot tracking applications, a more appropriate requirement is that *every* solution of the system can be stabilized. System (2) is said to be *universally exponentially stabilizable with rate* λ if there exists a control law $u(t) = k(x(t), x^\star(t), u^\star(t), t)$ that globally exponentially stabilizes *any* feasible solution $(x^\star, u^\star)$ with rate λ.

3 Differential Dynamics and Control Contraction Metrics

Riemannian geometry extends familiar Euclidean notions of distances and angles to smooth nonlinear manifolds [22]. The central idea is that at each point on a manifold, a local Euclidan geometry is defined on the tangent space by way of a "metric": a smoothly varying inner product on tangent vectors. In coordinates, this can be written as the inner product $\langle \delta_1, \delta_2 \rangle_x = \delta_1' M(x) \delta_2$ for some positive-definite matrix $M(x)$ that smoothly depends on x. Lengths of parameterized curves $\gamma(s)$, $s \in [0, 1]$ can then be computed as

$$l(\gamma) = \int_0^1 \sqrt{\langle \gamma_s, \gamma_s \rangle_\gamma} ds$$

where $\gamma_s = \frac{\partial \gamma}{\partial s}$. The Riemannian distance between two points is the length of the shortest path (a "geodesic") joining them. If the manifold is $\mathbb{R}^n$ and the metric M

is independent of x, i.e. "flat", then all geodesics are straight lines and the distance between two points x_1, x_2 is just $\sqrt{(x_1 - x_2)'M(x_1 - x_2)}$.

Contraction analysis is based on the relative motion of "nearby" solutions of a dynamical system [17]. If there exists some metric in which nearby solutions get closer (differential variations shrink), then all solutions converge globally. This is analogous to the obvious statement that if all the links in a chain get shorter, then the chain itself must get shorter. A contraction metric is a Riemannian metric on the state space for which all distances shrink under the flow of the system.

A smooth nonlinear system $\dot{x} = f(x, t)$ has differential (a.k.a. variational, linearized) dynamics $\dot{\delta}_x = \frac{\partial f}{\partial x}\delta_x$ along any particular solution. One of the key advantages of contraction analysis is that the differential dynamics are *linear time-varying*, and therefore many tools from analysis of linear systems can be applied to nonlinear systems to obtain global results, without approximation.

In [15, 16] this idea was extended from analysis to control design. Note that a Riemannian metric defines not only a local measure of length $\sqrt{\delta' M(x)\delta}$, but also a local notion of orthogonality: two tangent vectors δ_1, δ_2 at a point x are orthogonal if $\langle \delta_1, \delta_2 \rangle_x = \delta_1' M(x)\delta_2 = 0$. The main results of [16] is the following: if a control system of the form (2) is contracting in all directions *orthogonal* to the actuated directions, then all feasible trajectories are stabilizable. This can be considered a contraction version of the classical notion of a control Lyapunov function.

To make the statement precise, we first note that the control system (2) has the following differential dynamics:

$$\dot{\delta}_x(t) = A(x, u)\delta_x(t) + B(x)\delta_u(t) \tag{4}$$

where $A(x, u) = \frac{\partial}{\partial x}(f(x) + B(x)u)$. The main result of [16] was the following.

Theorem 1 ([16]) *Consider the system* (2) *with differential dynamics* (4). *If there exists a symmetric matrix function $M(x)$ and constants $\alpha_2 \geq \alpha_1 > 0$ and $\lambda > 0$ such that $\alpha_1 I \leq M \leq \alpha_2 I$ for all x and the following implication is true for all x, u:*

$$\delta_x' M B = 0 \Longrightarrow \delta_x'(\dot{M} + A'M + MA + 2\lambda M)\delta_x < 0, \tag{5}$$

then the system is universally exponentially stabilizable with rate λ.

This result also holds for time-varying systems and metrics [16]. To understand (5), consider that the rate of change of a differential squared-length $\delta_x' M(x)\delta_x$ is given by

$$\frac{d}{dt}(\delta_x' M(x)\delta_x) = \delta_x'(\dot{M} + A'M + MA)\delta_x + 2\delta_x MB\delta_u. \tag{6}$$

Depending on the values of x, u and δ_x in (6), there are two possibilities: If $\delta_x' MB \neq 0$, then the right-hand-side of (6) can be made negative by appropriate choice of differential control input δ_u. Indeed, since (6) is affine in δ_u, it can be made negative with as large a magnitude as desired. On the other hand, if $\delta_x' MB = 0$, then the control input has no effect on (6), but by (5) the system is "naturally" contracting, i.e. differential lengths decrease with rate λ, since (5) implies $\frac{d}{dt}\sqrt{\delta_x' M(x)\delta_x} < -\lambda\sqrt{\delta_x' M(x)\delta_x}$. Hence it can be shown [16] that there exists a "differential feedback" $\delta_u(x, \delta_x, u, t)$ that stabilizes the differential dynamics.

To construct the actual control signal u, one constructs a minimal path (geodesic) γ, parameterized by $s \in [0, 1]$, joining x and $x^\star$, i.e. $\gamma(0) = x^\star$, $\gamma(1) = x$, and the differential control signal δ_u is integrated along this smooth path:

$$\mathfrak{u}(t, s) = u^\star(t) + \int_0^s \delta_u \left(\gamma(\mathfrak{s}), \tfrac{\partial\gamma}{\partial s}(\mathfrak{s}), \mathfrak{u}(t, \mathfrak{s})\right) d\mathfrak{s}, \tag{7}$$

and the actual control signal that is applied is $u(t) = \mathfrak{u}(t, 1)$, see [16] for details and proofs. In general, this involves an on-line optimization reminiscent of (but simpler than) model predictive control to find γ. However, as mentioned above, if the metric is independent of x then geodesics are just straight lines.

3.1 Solution via Convex Optimization

A significant advantage of the CCM methodology is that, unlike a control Lyapunov function, the search for a CCM can be written as a convex optimization problem.

Making the change of variables $\eta = M(x)\delta$ and $W(x) = M(x)^{-1}$, (5) is rewritten as $\eta' B = 0$ implies $\eta'(-\dot{W} + AW + WA' + 2\lambda W)\eta < 0$. This can be expressed in the reduced form:

$$B_\perp'(-\dot{W} + AW + WA' + 2\lambda W)B_\perp < 0, \tag{8}$$

where $B_\perp(x)$ is a matrix function satisfying $B_\perp(x)'B(x) = 0$ for all x. In tensor analysis, the matrix W is a *dual* metric (contravariant rather than covariant), and the bijection $\delta \leftrightarrow \eta$ is related to the "musical isomorphisms".

Both the constraint (8) and the uniform boundedness of $W(x)$ are convex in the elements of $W(x)$. Both take the form of pointwise linear matrix inequalities (LMI) and generalize the standard LMI condition for stabilizability of a linear time-invariant system [23]. While convex, these conditions are infinite-dimensional: the elements of W are smooth functions of x, and there are infinitely-many constraints: the LMI conditions must hold for all x.

To solve this numerically we require a finite-dimensional approximation. Standard methods include choosing some basis for the functions (e.g. polynomials or trigonometric polynomials) and then enforcing the constraints by gridding over a region of state space, or by using a tractable (e.g. LMI) approximation to the set of positive polynomials, such as sum-of-squares or related approximations [10, 12]. The use of sum-of-squares was examined for contraction *analysis* in [24].

A more explicit condition is the following:

$$-\dot{W} + AW + WA' - \rho BB' + 2\lambda W \leq 0. \tag{9}$$

This condition clearly implies (8), since after left and right multiplying by $B_\perp'$ and $B_\perp$, respectively, the term involving ρ vanishes. Furthermore, this leads to an explicitly constructed differential control $\delta_u = -K(x)\delta_x$, with $K = -\frac{1}{2}\rho B' W^{-1}$.

Note that if the metric, ρ, and B are independent of x, then the resulting control law is just a linear feedback gain: $u(t) = u^\star(t) - \frac{1}{2}\rho B' W^{-1}(x(t) - x^\star(t))$.

4 CCM Designs for Fully-Actuated Robots

The conditions for a CCM can be expressed as a convex optimization problem. However, for the purpose of understanding the method, it is interesting to consider cases in which a CCM can be constructed analytically. In this section, we show that the CCM conditions in the previous section reduce to a very simple form for systems which are “second-order” and fully-actuated. Furthermore, the resulting designs turn out to be closely related to sliding control.

The differential dynamics of (2), (3) have the form:

$$\frac{d}{dt}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix} = \underbrace{\begin{bmatrix}0 & I\\ \star & \star\end{bmatrix}}_{A(x)}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ H(q)^{-1}R\end{bmatrix}}_{B(x)}\begin{bmatrix}\delta_1\\ \delta_2\end{bmatrix}$$

where $\star$ denotes an element that will not affect the results of this section. The only information we need for $B(x)$ is that $H(q)^{-1}R$ is full-rank, which is true for fully-actuated systems.

Consider the dual-CCM condition (8) for this system. The annihilator matrix $B_\perp(x)$ can be any matrix such that $B_\perp(x)'B(x) = 0$, which does not specify it uniquely, but a simple and natural choice is $B_\perp = \begin{bmatrix}I & 0\end{bmatrix}'$.

In general, we may choose a state-dependent metric $M(x)$ or dual metric $W(x) = M(x)^{-1}$, but for this class of systems it turns out to be sufficient to consider a constant metric, which we decompose into four $n \times n$ blocks:

$$W = \begin{bmatrix}W_{11} & W_{12}\\ W_{12}' & W_{22}\end{bmatrix}.$$

Condition (8) then states simply that W is a dual CCM if the upper-left block of $AW + WA' + 2\lambda W$ is negative definite. But due to the simple structure of $A(x)$, this reduces to the statement

$$W_{12} + W_{12}' < -2\lambda W_{11}. \tag{10}$$

In other words, any positive definite block matrix with off-diagonals that have a negative-definite symmetric part is a dual-CCM for some $\lambda > 0$. This is true for any second-order fully-actuated system, regardless of the dynamics.

Let us examine the geometrical meaning of this statement for a one-degree of freedom system with states $q, \dot{q}$. Then W is a two-by-two matrix. Since W_{11} is required to be positive, the condition $W_{12} + W'_{12} < -2\lambda W_{11}$ means that W_{12} is negative. From the formula for a two-by-two matrix inverse, the metric $M = W^{-1}$ must have all positive entries.

Now, since $B = [0, 1]'$, the uncontrolled differentials (those for which $\delta'_x MB = 0$) must satisfy $\delta_q m_{12} + \delta_{\dot{q}} m_{22} = 0$. i.e. $\delta_{\dot{q}} = \dot{\delta}_q = -\frac{m_{12}}{m_{22}}\delta_q$. But since all entries of M are positive, any non-zero δ_q and $\dot{\delta}_q$ are of opposite sign. i.e. the lengths of coordinate differentials δ_q are shrinking. Meanwhile, the statement $\delta'_x MB = 0$ implies that differentials in accelerations, which inhabit the second row of the differential dynamics, have no effect on the squared-length $\delta'_x M\delta_x$.

4.1 Connection to Sliding Control

We have shown that for second-order, fully-actuated systems it is easy to construct a flat control contraction metric. Interestingly, when taken to a certain limit, this construction transitions to another method for guaranteeing convergence of second-order systems: sliding control. In sliding control, one chooses some desired first-order dynamics for the configuration coordinates, e.g.

$$\frac{d}{dt}(q(t) - q^\star(t)) = -a(q(t) - q^\star(t)) \tag{11}$$

for some constant $a > 0$. For the second-order system, this means the state should move rapidly to the line in phase space given by $\frac{d}{dt}(q - q^\star) + a(q - q^\star) = 0$, known as a "sliding surface". This objective can be achieved applying high-gain or switching force feedback to drive the joint velocity towards $\dot{q}^\star(t) - a(q(t) - q^\star(t))$.

Let us construct our dual-CCM W to be of the form

$$W = \begin{bmatrix} I & -\lambda I \\ -\lambda I & (1+\varepsilon)\lambda^2 I \end{bmatrix},$$

where ε and λ are positive constants. By construction, this W satisfies (10), and it is also positive-definite, since taking the Schur complement with respect to the upper-right block gives $(1+\varepsilon)\lambda^2 I - \lambda^2 I = \varepsilon\lambda^2 I$, which is positive-definite. Taking the block matrix inversion, and setting $a = \frac{1}{\lambda}$, we have the metric $M = W^{-1}$:

$$M = \begin{bmatrix} (1+\frac{1}{\varepsilon})I & \frac{1}{\lambda\varepsilon}I \\ \frac{1}{\lambda\varepsilon}I & \frac{1}{\lambda^2\varepsilon}I \end{bmatrix} \approx \frac{1}{\varepsilon}\begin{bmatrix} I & aI \\ aI & a^2 I \end{bmatrix} =: \hat{M},$$

where the approximation $M \approx \hat{M}$ holds for $\varepsilon \ll 1$. Now, the eigenvalue/eigenvector pairs of $\hat{M}$ are of the form:

$$\lambda_i = 0, v_i = \begin{bmatrix} ae_i \\ -e_i \end{bmatrix}, \quad \lambda_{n+i} = \frac{1}{\varepsilon}(1+a^2), v_2 = \begin{bmatrix} e_i \\ ae_i \end{bmatrix}, \quad i = 1, 2, ..., n, \qquad (12)$$

where e_i are vectors with all zeros except for 1 at the i^{th} component. So as $\varepsilon \to 0$, level sets of the metric approach long valleys in the direction of v_1, with very steep sides in the direction v_2.

A feedback controller that makes $\delta_x' M \delta_x$ decrease exponentially will necessarily make state deviations in the direction of v_2 shrink very rapidly, but is under no obligation to make those in the direction of v_1 shrink. So the differentials will, after short transients, lie almost upon the line $a\delta_q + \delta_{\dot{q}} = 0$. If all the differentials line up like this, then $\dot{q} - \dot{q}^\star = -a(q - q^\star)$, which is exactly the target dynamics of the sliding controller described above.

4.2 Example: The Inverted Pendulum

The simple pendulum provides a useful illustration because all details can be worked out by hand and the results plotted in full. The state space of the system is cylindrical, which allows us to illustrate the geometrical and physical meaning of the geodesic-based control law. Normalising physical parameters, we can write the pendulum dynamics as

$$\ddot{q} + d\dot{q} + \sin q = u.$$

Taking the state space $x = [q, \dot{q}]'$ we have the differential dynamics (4) with

$$A(x) = \begin{bmatrix} 0 & 1 \\ -\cos q & -d \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

For this system, W is a 2×2 matrix, and condition (10) is just $w_{12} < -\lambda w_1$. So any positive-definite matrix with negative off-diagonal is a dual-CCM for the simple pendulum. By inverting, any constant positive-definite matrix with *positive* off-diagonal is a CCM for the simple pendulum.

Let us take the particular metric $m_{11} = 5, m_{12} = 2, m_{22} = 1$, and consider the problem of swingup to the upright position. Figure 2 represents the cylindrical phase space “unwrapped” onto $\mathbb{R}^2$. The figure depicts the open-loop phase portrait with the two equilibria recurring periodically. It also depicts level sets of the Riemannian distance to the target (upright) equilibrium. Notice that there are certain diagonal “bands” in this unwrapped state space: all states in each such band share a particular “unwrapping” of the target equilibrium as their closest. In Riemannian geometry the separatrix these bands are known as the *cut locus* [22, Chap. 13].

The shape of the bands can be explained as follows: for positive $\dot{\theta}$ the system is moving to the right in the phase portrait, so a target equilibrium to the right will be “closer” in terms of the metric. So, e.g., in closed-loop with a very large initial $\dot{\theta}$, the pendulum will spin around several times before coming to rest.

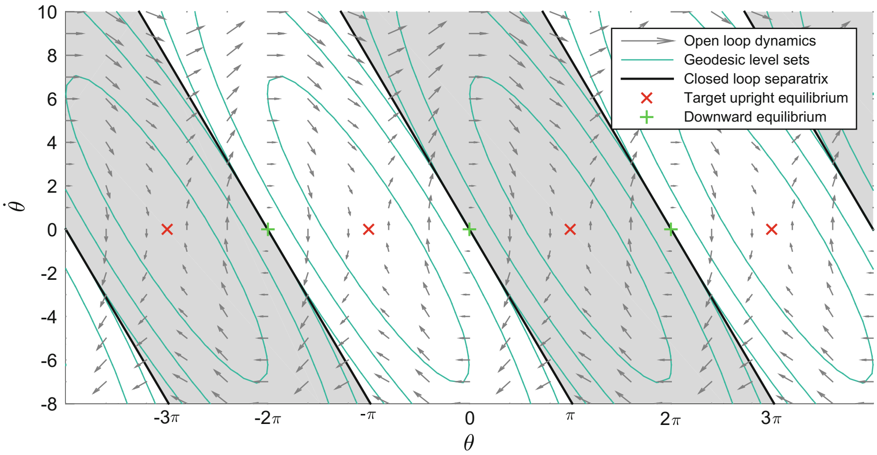


Fig. 2 Level sets of the distance metric, and bands of states that are closest to a particular "unwrapped" upright equilibrium

So far we have a metric but not the control gain, so we apply the more constructive condition (9)

$$\begin{bmatrix} 2(w_{12} + \lambda w_1) & w_2 - w_1 \cos q - w_{12} d + 2\lambda w_{12} \\ \star & 2(\lambda w_2 - w_{12} \cos q - w_2 d) - \rho(x) \end{bmatrix} \leq 0.$$

Assuming $w_{12} < -\lambda w_1$, this is equivalent via Schur complement to the statement

$$\rho(x) \geq \frac{(w_2 - w_1 \cos q - w_{12} d + 2\lambda w_{12})^2}{2(w_{12} + \lambda w_1)} - 2(\lambda w_2 - w_{12} \cos q - w_2 d),$$

which could be enforced pointwise by $\rho(x)$. At the expense of more aggressive control, one can give a fixed-gain feedback: since $w_1, w_2 > 0$ and $w_{12} < 0$ the right-hand side has a maximum when $\cos(x) = -1$, i.e. $x = \pi \pm 2k\pi, k = 1, 2, 3, \ldots$. Therefore it is sufficient to take the constant multiplier:

$$\rho = \frac{(w_2 + w_1 - w_{12} d + 2\lambda w_{12})^2}{2(w_{12} + \lambda w_1)} - 2(\lambda w_2 + w_{12} - w_2 d)$$

and the constant linear feedback $u = -\frac{1}{2}\rho B' W^{-1}(x - x^\star)$.

Figure 3 shows closed-loop responses for two nearby initial conditions, on either side of the cut locus. Note that in the unwrapped representation they converge to different equilibria, whereas in the (true) cylindrical phase space they both converge to the upright equilibrium but with different winding around the cylinder.

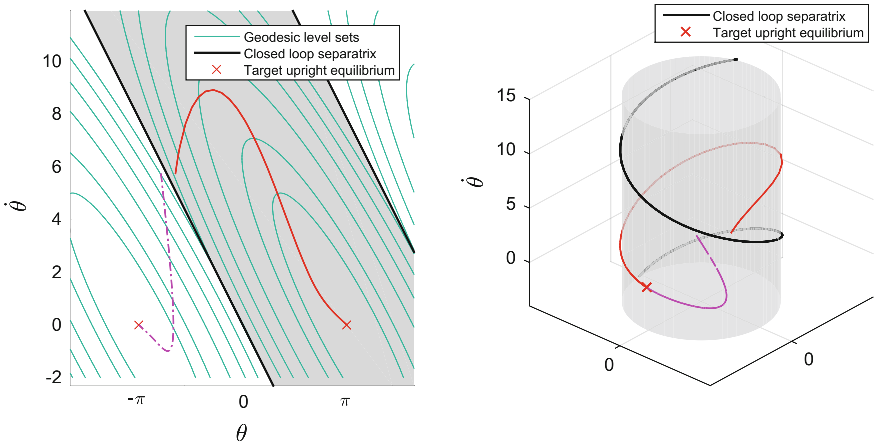


Fig. 3 Example trajectories of two nearby initial conditions. In unwrapped representation (*left*) and the cylindrical phase space (*right*)

5 Energy-Based Control and Virtual Systems

The design in the previous section made use of only very limited physical knowledge: the fact that the second set of state variables are the derivatives of the first set of state variables. In this section, we show that an alternative approach based on a so-called "virtual control system" makes use of more physical information and results in a classical energy-based tracking controller. We will also show how these designs can be applied for assigning desired dynamics for systems on nonlinear manifolds.

Virtual systems were introduced for convergence analysis in [25, 26], and here we extend this technique to control design. Given the system (2), a *virtual system* is a new system of the form

$$\dot{y} = \bar{f}(y, x) + \bar{B}(y, x)u \tag{13}$$

with the property that $\bar{f}(x, x) = f(x)$ and $\bar{B}(x, x) = B(x)$, where $y, x \in \mathbb{R}^n$. The variable x, taken as an exogenous input, is the state of the true system (2).

Theorem 2 *Suppose condition* (5) *holds for the* virtual *system, i.e. in terms of differentials in y, for some metric $M(y, x, t)$, then any trajectory $x^\star(t)$ of* (2) *for which there exists a control law $\bar{u}(x, x^\star)$ such that*

$$\dot{x}^\star = \bar{f}(x^\star, x) + \bar{B}(x^\star, x)\bar{u}(x, x^\star), \tag{14}$$

for all x, t can be globally exponentially stabilized.

Proof A controller is constructed using the method of [15] for the virtual system, treating the true state x as a time variation, except that in (7) except with $\bar{u}(t)$ replacing $u^\star(t)$ as the boundary condition. By construction, there is a path of states of the virtual system γ which shrinks in length exponentially. Furthermore, the use of $\bar{u}(t)$ as a boundary condition ensures that one end of this path ($s = 0$) follows the desired trajectory of the true system $x^\star(t)$, while the other end ($s = 1$) follows the actual trajectory of the true system $x(t)$, by the property that $\bar{f}(x, x) = f(x)$ and $\bar{B}(x, x) = B(x)$. Therefore the distance between x and $x^\star$ decreases exponentially. Note that intermediate trajectories $s \in (0, 1)$ do not necessarily correspond to trajectories of the true system. ■

Mechanical systems are an important example where virtual systems can simplify control design. For the system (1), we can construct the associated virtual system (similar to [26]):

$$H(q)\dot{y} + C(q, \dot{q})y + g(q) = Ru$$

where $(q, \dot{q})$ is the true state. Since the system is fully actuated, assume without loss of generality that $R = I$, which can always be achieved by change of variables on the control input. Note the virtual system is linear in y. The differential dynamics of the virtual system are given by $H(q)\dot{\delta}_y + C(q, \dot{q})\delta_y = \delta_u$. Now consider the metric $\delta_y' H(q)\delta_y$. Its derivative is $\frac{d}{dt}\left(\delta_y' H(q)\delta_y\right) = \delta_y'(\dot{H} - 2C)\delta_y + 2\delta_y'\delta_u = 2\delta_y'\delta_u$. The second equality follows from the skew-symmetry of $\dot{H} - 2C$. So any differential feedback of the form $\delta_u = -K(q)\delta_y$ with $K(q)$ positive-definite stabilizes the differential dynamics. Since the metric $H(q)$ and gain $K(q)$ are independent of y, geodesics are straight lines and the feedback controller takes the form $u = \bar{u} - K(q)(\dot{q} - \dot{q}^\star)$ with

$$\bar{u} = H(q)\ddot{q}^\star + C(q, \dot{q})\dot{q}^\star + g(q).$$

This controller will stabilize the generalized velocities to $\dot{q}^\star$. One can then use $\dot{q}^\star$ to design desired first-order dynamics, as in sliding control [27].

5.1 Example: Swingup of a Double Pendulum

We will illustrate the above method by application to swingup control of a (fully-actuated) double pendulum. The configuration space is toroidal in topology: $q = [\theta_1, \theta_2]'$, with θ_1 the angle of the "shoulder" link, relative to downward position, and θ_2 the "elbow" angle. The equations of motion can be found in, e.g., [3, 8].

A common choice of sliding surface is (11), but an alternative choice that respects the identity of angles offset by 2π is $\frac{d}{dt}(\theta - \theta_d) = -\lambda \sin(\theta - \theta_d)$, where θ_d is some desired angle. This assigns smooth and almost-globally stable dynamics to each coordinate. Note that this involves subtracting velocities which are attached at different points of the manifold, which depends on a particular choice of coordinate system. For systems on nonlinear manifolds an alternative is to incorporate the vector transport method in [28].

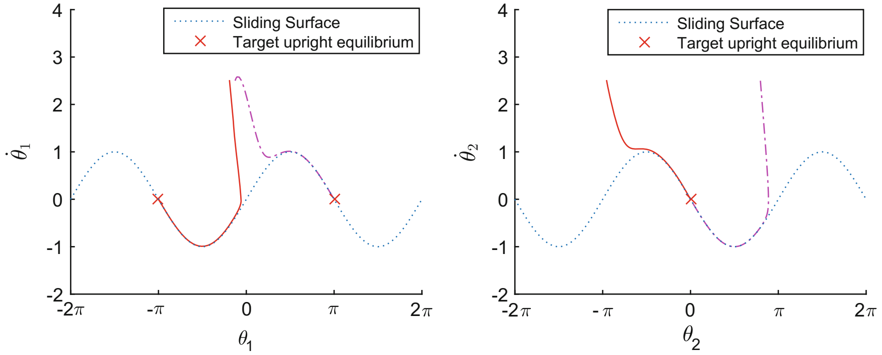
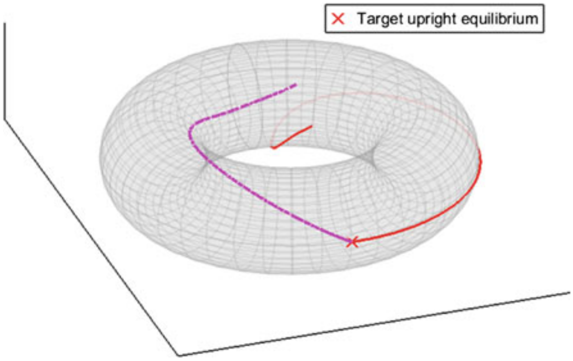


Fig. 4 Trajectories of the double pendulum during swingup

Fig. 5 Trajectories of the double pendulum plotted on the nonlinear configuration manifold



The results of simulations are shown in Fig. 4 in unwrapped coordinates. As before, the closed-loop trajectories from two nearby initial conditions separate and converge to apparently different equilibria. In Fig. 5 the configuration trajectories are plotted on the torus, and it is clear that both converge to the same equilibrium (both pendulums upright) but with a different winding around the torus.

6 Underactuated Robots and Optimization-Based Synthesis

For underactuated systems, i.e. those with fewer actuators than degrees of freedom, the techniques of Sects. 4 and 5 are not guaranteed to be applicable. Another recently developed approach is to use linear optimal control techniques (e.g. LQR) to locally stabilize a system, and then verify stability in a region using semialgebraic optimization [10–12]. In this section we discuss the connections between CCM and these optimization-based methods.

Consider the following family of LQR-like cost functions for tracking a desired trajectory $x^\star(t), u^\star(t), t \in [0, \infty)$:

$$J(x,u) = \int_0^\infty [(x - x^\star)'Q(x - x^\star) + (u - u^\star)'R(u - u^\star)]dt.$$

Suppose there exists a uniformly positive-definite matrix $W(x)$ such that

$$\begin{bmatrix} (\dot{W} - WA' - AW + BR^{-1}B') & WL' \\ LW & I \end{bmatrix} \geq 0, \tag{15}$$

for all x, u, with $Q = L'L$, then for any target trajectory and any initial condition $x(0)$, the CCM controller achieves the following upper bound on the cost: $J(x,u) \leq d_M(x(0), x^\star(0))^2$, where d_M is the Riemannian distance induced by $M = W^{-1}$.

Note that if W, B are constant, and (15) is evaluated at an equilibrium state, then it is equivalent (via Schur complement) to the condition $WA' + AW - BR^{-1}B' + WQW \leq 0$. Multiplying on either side by $M = W^{-1}$ and replace the inequality with equality recovers exactly the algebraic Riccati equation of LQR control [9]. It follows that if the system is locally exponentially stabilizable at an equilibrium point, then for some region around that point a CCM exists.

6.1 Relation to LQR-Trees and Model Predictive Control

Let us relate this approach to LQR-Trees [11]. Using CCMs, one can choose some region of state space $\mathscr{X}$, and then the search for a CCM valid for all $x \in \mathscr{X}$ is convex (though infinite dimensional). Practically, this problem can be solved via gridding or via sum-of-squares with Lagrange multipliers. In LQR-Trees, a locally-valid controller is found using LQR, and *then* its region of validity is estimated using sum-of-squares. Furthermore, the method of [11] constructs a controller which stabilizes a *particular* target equilibrium point or trajectory only, whereas the CCM approach produces a controller that stabilizes *any* feasible trajectory that remains in the verified region, without prior knowledge of the target trajectory.

Additionally, the CCM method can be thought of as a “middle ground” between LQR (or other methods based on linearization) – for which off-line synthesis is convex and on-line computation is cheap, but stabilization is local – and model predictive control – for which stabilization is in principle global, but on-line computation may be prohibitive. In the CCM approach, there is an off-line convex problem to find the CCM, and then the real-time controller is computed using an on-line geodesic optimization, which is simpler than MPC due to the lack of dynamic constraints. This relationship was explored in detail in [29].

6.2 Example: Underactuated Cart-Pole System

The cart-pole system is a classic example of an underactuated system: an unactuated inverted pendulum is mounted on an actuated cart. There are two configuration coordinates: the pendulum angle θ and the cart position ξ. The derivatives are denoted ω and v, respectively, and the control input is a force to the cart, u. See Fig. 1.

To simplify the computation we first apply the partial feedback linearization of [30] to obtain a state vector $[\theta, \omega, \xi, v]'$ with dynamics $\dot{\theta} = \omega, \dot{\omega} = \sin(\theta) - \cos(\theta)u, \dot{\xi} = v, \dot{v} = u$. As mentioned in Sect. 3, the CCM controller is linear if B, W and ρ are independent of x. With this in mind, we take the new coordinates $x = [\theta, \eta, \xi, v]'$ with $\eta = \sec\theta\omega$, which is an invertible transformation on $\theta \in (-\frac{\pi}{2}, \frac{\pi}{2})$. In these coordinates the dynamics are

$$\frac{d}{dt}\begin{bmatrix}\theta\\ \eta\\ \xi\\ v\end{bmatrix} = \begin{bmatrix}\eta\cos\theta\\ \eta^2\sin\theta + \tan\theta\\ v\\ 0\end{bmatrix} + \begin{bmatrix}0\\ -1\\ 0\\ 1\end{bmatrix}u,$$

and the differential dynamics are:

$$\dot{\delta}_x = \underbrace{\begin{bmatrix}-\eta\sin\theta & \cos\theta & 0 & 0\\ -\eta^2\cos\theta + \sec^2\theta & 2\eta\sin\theta & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix}}_{A(x)}\delta_x + \underbrace{\begin{bmatrix}0\\ -1\\ 0\\ 1\end{bmatrix}}_{B}\delta_u.$$

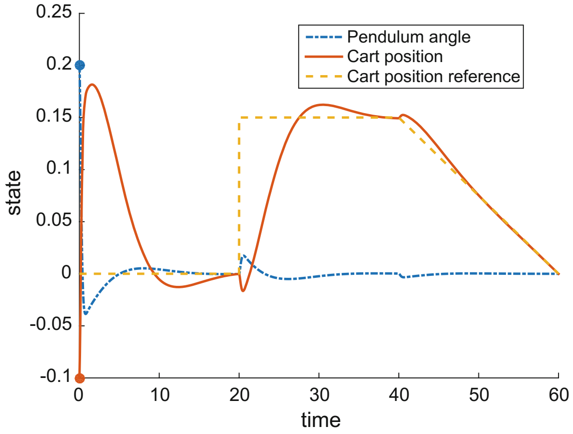
Note that these transformations are merely to simplify computation: it was proven in [16] that the existence of a CCM is invariant under smooth changes of variables and affine feedback transformations.

For this system, we used the LMI parser Yalmip [31] and solver Mosek to solve for W satisfying (15) with the cost weights $Q = \text{diag}[1, 1, 15, 1]$ and $R = 10$, over the region $\theta \in [-\pi/4, \pi/4]$ and $\eta \in [-0.5, 0.5]$ via gridding. The resulting optimization problem took about 1.6 seconds to solve on a standard desktop computer.

Tracking results are shown in Fig. 6. The system first responds to initial conditions $x = [0.2, 0, -0.1, 0]'$ and with the desired state at the origin (i.e. balancing upright). After 20 s there is a step change in the desired cart position $\xi^\star = 0.15$, all other desired states remain zero. The controller reacts to bring the cart to the new desired position. After 40 seconds, the desired cart position linearly shifts back to the origin, and the desired cart velocity is set accordingly. After a small transient, the controlled system converges to this path.

Note that in all of these simulations, the desired pendulum position remained upright: $\theta^\star = 0$. Therefore the reference trajectory was not actually feasible for the nonlinear system, since the unactuated pendulum would not stay upright if the cart accelerates.

Fig. 6 Response of the cart-pole system to initial conditions and a changing cart position reference



7 Conclusions

We have explored the use of control contraction metrics for the design of robot tracking controllers. The central theme of the paper is that, on the one hand, CCMs extend linearization/optimization-based methods (e.g. LQR-Trees) to tracking problems and can be found using convex optimization. On the other hand, for fully-actuated systems the CCM conditions can be solved analytically and reduce to classical methods of sliding and energy-based design. Thus CCMs can be seen to represent a bridge between previously disparate methods.

References

1. Vukobratović, M., Borovac, B.: Zero-moment point — thirty five years of its life. Int. J. Humanoid Robot. **1**(1), 157–173 (2004)
2. Slotine, J.-J.E., Li, W.: Applied Nonlinear Control. Prentice-Hall, USA (1991)
3. Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot Modeling and Control, vol. 3. Wiley, New York (2006)
4. Mellinger, D., Michael, N., Kumar, V.: Trajectory generation and control for precise aggressive maneuvers with quadrotors. Int. J. Robot. Res. **31**(5), 664–674 (2012)
5. Collins, S., Ruina, A., Tedrake, R., Wisse, Martijn: Efficient bipedal robots based on passive-dynamic walkers. Science **307**(5712), 1082–1085 (2005)
6. Albu-Schäffer, A., Ott, C., Hirzinger, G.: A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. Int. J. Robot. Res. **26**(1), 23–39 (2007)
7. Fantoni, I., Lozano, R.: Non-linear Control for Underactuated Mechanical Systems. Springer, Berlin (2001)
8. Tedrake, R.: Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832) (2014)
9. Anderson, B.D.O., Moore, J.B.: Optimal Control: Linear Quadratic Methods. Prentice-Hall, USA (1990)

10. Parrilo, P.A.: Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization. PhD thesis, California Institute of Technology (2000)
11. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, John W.: LQR-trees: feedback motion planning via sums-of-squares verification. Int. J. Robot. Res. **29**(8), 1038–1052 (2010)
12. Majumdar, A., Ahmadi, A.A., Tedrake, R.: Control and verification of high-dimensional systems via DSOS and SDSOS optimization. In: 53rd IEEE Conference on Decision and Control, Los Angeles, CA (2014)
13. Henrion, D., Korda, M.: Convex computation of the region of attraction of polynomial control systems. IEEE Trans. Autom. Control **59**(2), 297–312 (2014)
14. Majumdar, A., Vasudevan, R., Tobenkin, M.M., Tedrake, R.: Convex optimization of nonlinear feedback controllers via occupation measures. Int. J. Robot. Res. **33**(9), 1209–1230 (2014)
15. Manchester, I.R., Slotine, J.-J.E.: Control contraction metrics and universal stabilizability. In: Proceedings of the IFAC World Congress, Cape Town, South Africa (2014)
16. Manchester, I.R., Slotine, J.-J.E.: Control contraction metrics: convex and intrinsic criteria for nonlinear feedback design, IEEE Transactions on Automatic Control, in press (2017)
17. Lohmiller, W., Slotine, J.-J.E.: On contraction analysis for non-linear systems. Automatica **34**(6), 683–696 (1998)
18. Spong, M.W.: The swing up control problem for the acrobot. Control Systems, IEEE, **15**(1):49–55, (1995)
19. Åström, K.J., Furuta, K.: Swinging up a pendulum by energy control. Automatica **36**(2), 287–295 (2000)
20. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: learning attractor models for motor behaviors. Neural Comput. **25**(2), 328–373 (2013)
21. Manchester, I.R., Slotine, J.-J.E.: Transverse contraction criteria for existence, stability, and robustness of a limit cycle. Syst. Control Lett. **62**, 32–38 (2014)
22. Do Carmo, M.P.: Riemannian Geometry. Springer, Berlin (1992)
23. Hespanha, J.P.: Linear Systems Theory. Princeton University Press, USA (2009)
24. Aylward, E.M., Parrilo, P.A., Slotine, J.-J.E.: Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming. Automatica **44**(8), 2163–2170 (2008)
25. Wang, W., Slotine, J.-J.E.: On partial contraction analysis for coupled nonlinear oscillators. Biol. Cybern. **92**(1), 38–53 (2005)
26. Jouffroy, J., Slotine, J.-J.E.: Methodological remarks on contraction theory. In: 43rd IEEE Conference on Decision and Control, The Bahamas (2004)
27. Slotine, J.-J.E., Li, W.: On the adaptive control of robot manipulators. Int. J. Robot. Res. **6**(3), 49–59 (1987)
28. Bullo, F., Murray, R.M.: Tracking for fully actuated mechanical systems: a geometric framework. Automatica **35**(1), 17–34 (1999)
29. Leung, K., Manchester, I.R.: Nonlinear stabilization via control contraction metrics: a pseudospectral approach for computing geodesics. In: Proceedings of the 2017 American Control Conference, Seattle, WA (2017)
30. Spong, M.W.: Energy based control of a class of underactuated mechanical systems. In: Proceedings of the IFAC World Congress, San Francisco, CA (1996)
31. Löfberg, J.: Yalmip: a toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference, Taipei, Taiwan (2004)

Inference-Enabled Information-Theoretic Exploration of Continuous Action Spaces

Shi Bai, Jinkun Wang, Kevin Doherty and Brendan Englot

1 Introduction

We consider a mobile robot that has no prior knowledge of the contents of its environment and must make repeated decisions about where to travel next, comprising an autonomous exploration problem [1]. Specifically, we formulate an information-theoretic exploration problem in which the long-term goal is to reduce entropy throughout the robot's environment map, and the short-term goal is to perform the sensing action in each iteration that will maximize mutual information, along the lines of [2]. We assume the robot is equipped with a range sensor and uses an occupancy grid [3] to represent and reason about the environment.

Motivated by the recent work of [4], which proved that a controller driven by mutual information maximization attracts a robot to unexplored space, our aim is to implement a mutual information maximization approach that is amenable to real-time decision making and scalable to higher-dimensional systems. The approach of [4], although successful, requires a predictive evaluation of the mutual information achieved by performing every possible sensing action within a robot's finely discretized action space. We hope to cut down the complexity by evaluating only a

S. Bai (✉) · J. Wang · B. Englot
Department of Mechanical Engineering, Stevens Institute of Technology,
Castle Point on Hudson, Hoboken, NJ 07030, USA
e-mail: sbai1@stevens.edu

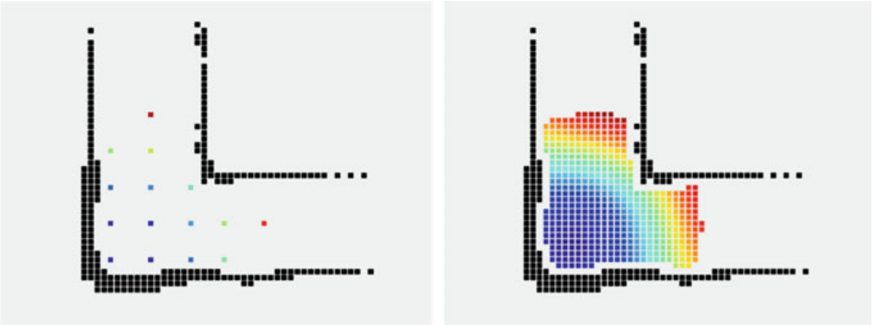
J. Wang
e-mail: jwang92@stevens.edu

B. Englot
e-mail: benglot@stevens.edu

K. Doherty
Department of Electrical and Computer Engineering, Stevens Institute of Technology,
Castle Point on Hudson, Hoboken, NJ 07030, USA
e-mail: kdoherty@stevens.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_24



(a) MI of discrete sensing actions. (b) MI predicted by GP regression.

Fig. 1 An illustration of two steps of the proposed decision-making process. In **a**, the current occupancy map is used to predict the mutual information at a set of discrete sensing locations. In **b**, a Gaussian process (GP) regression is performed over a local region of the space of sensing actions, using the data from (**a**) as training data. The *darkest* of the *blue cells* in (**a**) represents the current location of the robot, and the *black cells* represent known obstacles in the robot's occupancy grid. The color scale from *blue* to *red* indicates increasing mutual information (MI)

select number of actions, and using these as training data for a supervised learning procedure that will predict information gain throughout the continuous action space.

A key vehicle for capturing correlation among a discrete set of candidate actions will be Gaussian process regression [5], a method that has met with recent success in predicting outcomes within unknown regions of robot action [6] and observation [7] spaces. An example of the regression performed in a single decision-making step is illustrated in Fig. 1. The output of this regression is used to select the maximally informative sensing action from a continuous, local region of the robot's action space. Support vector regression [8] will also be investigated to determine whether similar results can be obtained with reduced computational effort.

1.1 Related Work

Among the earliest information-theoretic exploration strategies are those proposed by Whaite and Ferrie [9] and Elfes [2]. The former work proposes exploring an a priori unknown environment with the goal of minimizing entropy, and the latter work specifically proposes exploring to maximize the mutual information between sensor observations and an occupancy grid map. More recent works in information-theoretic exploration have considered the trade-off between maximizing mutual information and managing the localization uncertainty in a robot's simultaneous localization and mapping (SLAM) process [10–12], in addition to the selection of trajectories that maximize map accuracy [13]. Efforts to reduce the computational cost of evaluating mutual information over many possible future measurements have considered small, carefully selected sets of candidate trajectories, using a skeletonization of the known occupancy map [14] and the evaluation of information gain over a finite number of motion primitives [15, 16] or 3D viewpoints [17]. Limiting consideration to local neighborhoods of configurations permits efficient exploration by manipulators in 3D environments [18].

Gaussian process regression has been applied to the problem of robot action selection and control in a variety of contexts. It has been used to solve optimal control problems [6, 19], generate paths that reduce localization uncertainty [20], and select actions that are likely to observe physical objects of interest [21]. It has also been used to aid the inspection of structures by predicting the regions of variability in greatest need of additional measurement [22]. Gaussian process regression has also been used to generate maps that support the evaluation of informative actions [15, 23]. However, to the best of our knowledge, it has not been applied to the problem of action selection for the exploration of unknown environments modeled by occupancy maps, nor has support vector regression.

1.2 Paper Organization

We propose and describe below a methodology for choosing the most informative action from the continuous action space with the aid of supervised learning. A formal definition of the problem is given in Sect. 2, including brief introductions to Gaussian process regression and support vector regression. The proposed algorithm is given in Sect. 3, and the time complexity of the algorithm is analyzed in Sect. 4. Computational results are presented in Sect. 5, with conclusions and a discussion of areas for future work in Sect. 6.

2 Problem Definition

2.1 Information Gain

We define the space of mobile robot sensing actions to be the configuration space $\mathscr{C} \subseteq \mathbb{R}^d$, a subset of d-dimensional Euclidean space. We assume the robot's range sensor provides a 360-degree field of view, and that its occupancy grid map is discretized finely enough to represent the configuration space, in addition to serving as the robot's model of the environment. In the absence of obstacles, the robot is assumed capable of travel from any grid cell in the map to any other cell. A fundamental presumption in this formulation is that the robot's action space is a subset of the spatial configuration space; this, along with our other assumptions, are similar to those made in [4]. The implications of extending the proposed method to systems with more challenging topologies will be discussed in Sect. 6.

We define Shannon's entropy [24] over an occupancy grid map m as follows:

$$H(m) = -\sum_i \sum_j p(m_{i,j}) \log p(m_{i,j}) \tag{1}$$

where index i refers to the individual grid cells of the map and index j refers to the possible outcomes of the Bernoulli random variable that represents each grid cell, which is either free or occupied. Cells whose contents have never been observed are characterized as $p(m_{i,j}) = 0.5$, contributing one unit of entropy per cell. Cells whose contents are perfectly known contribute no entropy to the summation.

We use mutual information $I(m, x_i)$ to evaluate the expected information gain with respect to a specific configuration x_i, defined as follows:

$$I(m, x_i) = H(m) - H(m|x_i) \tag{2}$$

where $H(m)$ is the current entropy of the map, and $H(m|x_i)$ is the expected entropy of the map given a new sensor observation at configuration x_i. Our goal is to pick the optimal configuration x^* that maximizes the expected information gain.

$$x^* = \underset{x_i \in \mathscr{C}_{action}}{\text{argmax}}\ I(m, x_i) \tag{3}$$

In (3), $\mathscr{C}_{action}$ represents the subset of the configuration space from which the robot's next sensing action will be selected, typically within a short distance of the robot's current location.

2.2 Gaussian Process Regression

We assume a set of training data $\mathbf{x}$ represents the candidate sensing configurations x_i for which $I(m, x_i)$ has been computed. The values of $I(m, x_i)$ for all $x_i \in \mathscr{C}_{action}$ comprise the set of training outputs $\mathbf{y}$. Gaussian process regression [5] estimates the output values and corresponding covariance associated with a set of test configurations $\mathbf{x}_*$, according to Eqs. (4) and (5). The test configurations $\mathbf{x}_*$ will be finely discretized, with the same resolution as the occupancy grid map.

$$\bar{\mathbf{y}}_* = k(\mathbf{x}_*, \mathbf{x})[k(\mathbf{x}, \mathbf{x}) + {\sigma_n}^2 I]^{-1}\mathbf{y} \tag{4}$$

$$cov(\mathbf{y}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{x})[k(\mathbf{x}, \mathbf{x}) + {\sigma_n}^2 I]^{-1}k(\mathbf{x}, \mathbf{x}_*) \tag{5}$$

In the above equations, $\bar{\mathbf{y}}_*$ are the estimated values $I(m, x_{i*})$ for the test data $\mathbf{x}_*$, $cov(\mathbf{y}_*)$ is the covariance associated with these outputs, σ_n^2 is a vector of Gaussian noise variances associated with the observed outputs $\mathbf{y}$, and k($\mathbf{x}$,$\mathbf{x}$') is the kernel function, which gives a covariance matrix relating all pairs of inputs. The hyperparameters of the kernel function, which typically influence such characteristics as smoothness and length scales, can be trained using a preliminary set of representative training data.

We adopt a Matérn kernel function for this application, given by (6).

$$k(\mathbf{x},\mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}|\mathbf{x}-\mathbf{x}'|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}|\mathbf{x}-\mathbf{x}'|}{\ell}\right) \tag{6}$$

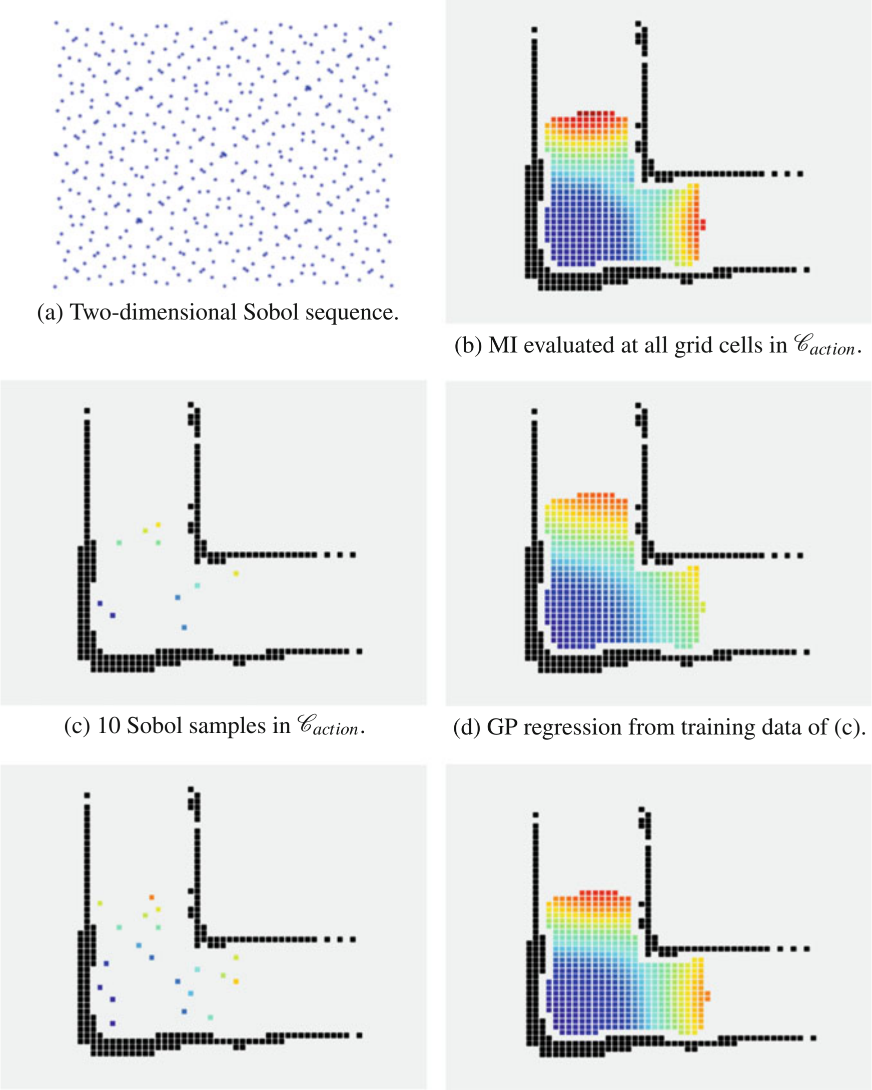
In (6), ν is a parameter used to vary the smoothness of the covariance, ℓ is a characteristic length, Γ is the gamma function, and K_ν is a modified Bessel function. In contrast with the squared exponential kernel function, which is more commonly used in Gaussian process regression [5], the Matérn kernel can be tuned to capture sharp variations in the estimated outputs. This has met with success in Gaussian process occupancy mapping, in which sharp and sudden transitions in occupancy probability due to obstacles are successfully modeled [23, 25]. Similarly, we anticipate sharp variations in mutual information due to the presence of obstacles, which will obstruct the visibility of some areas and permit the observation of others.

Example regressions performed over two sets of candidate training data are given in Fig. 2. The training data $\mathbf{x}$ is sampled from the two-dimensional Sobol sequence [26]. This pseudorandom sequence is selected to impose some regularity on the training data, as demonstrated by the field of samples shown in Subfigure (a). Subfigures (b), (c), and (e) represent the explicit evaluation of expected mutual information over several series of candidate views. Subfigures (d) and (f) represent the use of Gaussian process regression to predict the mutual information achieved by all actions within the finely discretized grid representing $\mathscr{C}_{action}$, the continuous action space. The boundaries of $\mathscr{C}_{action}$ are set according to the limits of the robot's field of view at its present location.

2.3 Support Vector Regression

Support vector regression [8] is an adaptation of the support vector machine [27] used for regression rather than classification. With the same training set $\mathbf{x}$ of candidate robot configurations employed as training inputs, and $I(m, x_i)$ for all $x_i \in \mathscr{C}_{action}$ as the set of training outputs $\mathbf{y}$, support vector regression estimates the output values associated with test configurations $\mathbf{x}_*$, but not their corresponding covariance, in contrast to Gaussian process regression.

Specifically, we adopt ε-support vector regression, in which we aim to find an approximate function with deviation less than ε from each output value at training time. The basic form for the hypothesis of a support vector regression estimator for test data $\mathbf{x}_*$ is as follows:



(a) Two-dimensional Sobol sequence.

(b) MI evaluated at all grid cells in $\mathscr{C}_{action}$.

(c) 10 Sobol samples in $\mathscr{C}_{action}$.

(d) GP regression from training data of (c).

(e) 20 Sobol samples in $\mathscr{C}_{action}$.

(f) GP regression from training data of (e).

Fig. 2 Gaussian process (GP) regression using samples from the Sobol sequence. In all images, the robot's current location is the same as depicted in Fig. 1, and the *black cells* represent known obstacles in the robot's occupancy grid. The color scale from *blue* to *red* indicates increasing mutual information (MI)

$$\bar{\mathbf{y}}_* = \sum_{i=1}^{l} (-\alpha_i + \alpha_i{}^*) k(\mathbf{x}_{*i}, \mathbf{x}) + b \tag{7}$$

where $k(\cdot)$ denotes the Matérn kernel function mapping from test inputs to features and ${\alpha_i}^* - \alpha_i$ denotes the learned weight for the i-th feature in $k(\cdot)$, from the solution to the dual problem described in [28]. Using this approach, results visually indistinguishable from those in Figs. 1 and 2 were obtained. This was achieved with reduced computational effort, with the only drawback being the lack of the empirical confidence in the results provided by the covariance of the test data, which would be supplied by Gaussian process regression. This method will be examined alongside Gaussian process regression to determine whether comparable results can be obtained with reduced computational effort in all problems of interest.

3 Algorithm Description

Algorithm 1 AutonomousExploration(x_{init}, m_{init}, InfoThreshold, $N_{samples}$)

```
$x_k \leftarrow x_{init}$; $m_k \leftarrow m_{init}$; $ActionHistory \leftarrow x_{init}$;
for $k \in \{1, 2, ..., NumIterations\}$ do
    $ActionSet \leftarrow \emptyset$;
    $MISet \leftarrow \emptyset$;
    for $x_i \in \mathscr{C}_{action}(x_k, N_{samples})$ do
        $MI \leftarrow ObservationPrediction(x_i, m_k)$;
        $MISet \leftarrow MISet \cup MI$;
        if $MI > InfoThreshold$ then
            $ActionSet \leftarrow ActionSet \cup x_i$;
        end if
    end for
    $ActionSet \leftarrow Regression(ActionSet, InfoThreshold, MISet, x_k, m_k)$;
    if $ActionSet \neq \emptyset$ then
        $x_{k+1} \leftarrow BestAction(ActionSet)$;
        $ActionHistory \leftarrow ActionHistory \cup x_{k+1}$;
    else
        $x_{k+1} \leftarrow ActionHistory(PreviousAction)$;
        $ActionHistory \leftarrow ActionHistory \setminus x_k$;
    end if
end for
$m_{k+1} \leftarrow MapUpdate(x_{k+1})$;
```

The exploration process proceeds according to Algorithm 1. On each iteration, an action set $\mathscr{C}_{action}$ is formulated within the sensor field of view at the robot's current location, and a designated number of sampled actions within the set is evaluated per Eq. (2), drawn from a Sobol sequence. Actions whose mutual information surpasses a designated threshold $InfoThreshold$ are added to a set of approved candidate actions $ActionSet$. All of the mutual information data are then used to perform the regression of choice to estimate the information gain of all other members of $\mathscr{C}_{action}$ whose mutual information was not explicity computed. Actions whose esti-

Algorithm 2 MI = ObservationPrediction(x_i, m_i)

```
$m \leftarrow m_i$;
for $beam_j \in SensorBeams(x_i)$ do
    $IntersectCell \leftarrow IntersectionDetected(beam_j, m)$;
    if $IntersectCell \neq \emptyset$ then
        $r_j = knnsearch(x_i, IntersectCell)$;
    else
        $r_j = MaxSensorRange$;
    end if
    $m \leftarrow EntropyUpdate(x_i, r_j)$;
end for
$MI = Entropy(m_i) - Entropy(m)$;
return $MI$;
```

mated mutual information exceeds the $InfoThreshold$ are also added to the set of candidate actions $ActionSet$. If at least one action is identified whose information gain surpasses the threshold, the robot performs the maximally informative sensing action. However, if none of the actions evaluated surpasses the threshold, the robot takes a step backwards and considers the actions at a previous location along the route traveled, where there may have been informative candidate actions that were not yet performed. The algorithm repeats until the designated number of iterations is performed, or map entropy drops below a user-designated lower limit.

Algorithm 2 gives the specific steps required to explicitly evaluate the mutual information at a designated sample action x_i. This entails a ray tracing computation along each of the robot's sensor beams, returning the ranges to the nearest obstacles intersected, if any. New entropy values are estimated in all cells that are anticipated to be intersected by a sensor beam, and the new expected map entropy is used to compute the expected mutual information after performing the designated sensing action. Algorithm 2 is a sub-routine of Algorithm 1 used to evaluate every Sobol sample from the action space whose mutual information is explicitly computed.

4 Analysis

When using Gaussian process regression, the computational complexity of Algorithm 1 is given in (8):

$$O(N_{steps}(N_{samples}N_{beams}N_{cells} + N_{samples}^3 + N_{samples}^2 N_{actions})) \tag{8}$$

where N_{steps} is the total number of sensing actions taken by the robot in the course of exploration, $N_{samples}$ is the number of designated configurations whose mutual information is explicitly evaluated, $N_{actions}$ is the total number of actions comprising $\mathscr{C}_{action}$ that are estimated using Gaussian process regression, N_{beams} is the number of beams emitted by the robot's range sensor, and N_{cells} is the worst-case number of occupancy grid cells that a beam may intersect. The term $N_{samples} N_{beams} N_{cells}$ represents the cost of explicitly evaluating mutual information in all cells intersected by the robot's sensor, for all designated actions $N_{samples}$. The term $N_{samples}^3 + N_{samples}^2 N_{actions}$ represents the cost of performing the subsequent Gaussian process regression, which requires the inversion of a matrix that is square in $N_{samples}$, and its subsequent multiplication with cross-covariance terms that scale with $N_{actions}$, the total number of sensing actions recovered from the "test data" of the Gaussian process regression. In practice, we have worked with $10 \leq N_{samples} \leq 20$, $N_{actions} \sim 300$, $N_{beams} = 360$, and $N_{cells} \sim 25$, and we have found that in this range, the complexity of the procedure is dominated by the first term, with the cost of the Gaussian process regression relatively minor in comparison to the cost of the mutual information computation. Hence, a much larger number of sensing actions can be evaluated approximately for a small additional cost on top of the initial evaluation of information gain over the original set of samples. Specific examples will be highlighted in the following section.

When using support vector regression, the complexity of Algorithm 1 is given in (9):

$$O(N_{steps}(N_{samples} N_{beams} N_{cells} + N_{samples}^3 + N_{actions})) \tag{9}$$

where training still takes on worst-case cubic complexity (which occurs when the upper bound on the coefficients α_i is large) but testing is linear in the number of candidate sensing actions. Once again, the complexity of the procedure is dominated by the $N_{samples} N_{beams} N_{cells}$ term, and the cost of the regression is minor in comparison to the cost of mutual information computation.

5 Computational Results

5.1 Experimental Setup

We explored the performance of our algorithm using two different maps: (1) a "maze" map representing an indoor environment (shown in Fig. 3) and (2) an "unstructured" map representing a forest-like environment (shown in Fig. 4). In our simulations, we assume the robot is equipped with a laser scanner with a 360° field of view and 1° resolution. The range of the laser scanner was set to 1 m, and all sensing actions considered were within a 0.5 m range of the robot, ensuring that the next sensing action lies within the robot's current field of view to the extent that its outcome can be reasonably predicted by a mutual information evaluation over the existing map. The exploration process was simulated using MATLAB.

We initialized the robot randomly within each map and simulated 100 instances of exploration for each of the six following cases: (a) choosing the best action among 10 Sobol samples (as depicted in Fig. 2c), (b) choosing the best action among 20

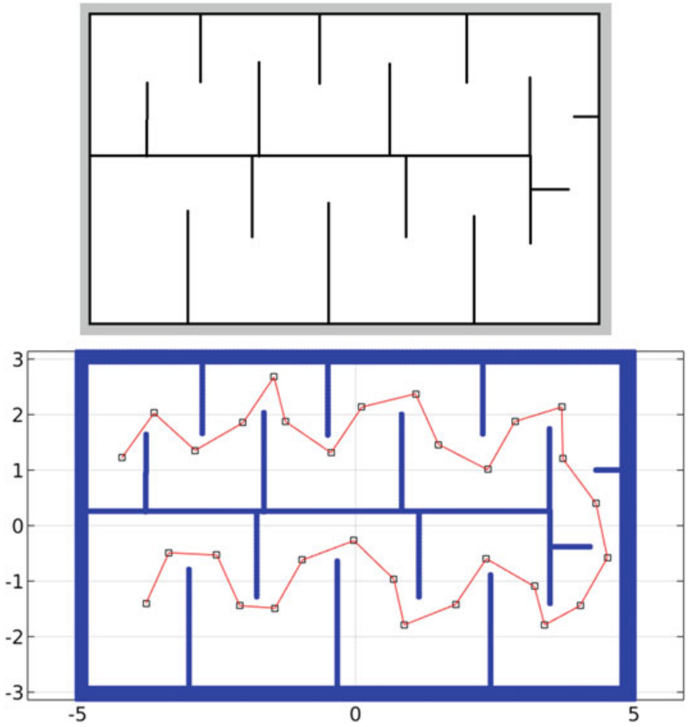


Fig. 3 The “maze” map used in our experiments is shown at *top*. The dimensions of the map are approximately 6 by 9 m. An occupancy map produced by exploration with GP regression, showing the trajectory of the robot, is given at *bottom*

Sobol samples (as depicted in Fig. 2e), (c) using 10 Sobol samples as the basis for Gaussian process regression and (d) support vector regression, then choosing the best action from the approximately continuous action space, and (e) using 20 Sobol samples as the basis for Gaussian process regression and (f) support vector regression, again choosing the best action from the approximately continuous action space. The robot is permitted to explore until its map entropy falls below a designated threshold, after which the simulation terminates. The Gaussian process regression computations were performed with the aid of the Gaussian processes for machine learning (GPML) MATLAB library [29], and support vector regression computations were performed in MATLAB using LIBSVM [28], with a precomputed Matérn kernel. The computation required for each trial was distributed across four cores of an Intel Xeon 5 3.0 GHz processor using the MATLAB Parallel Computing Toolbox, and a computer equipped with 4GB RAM.

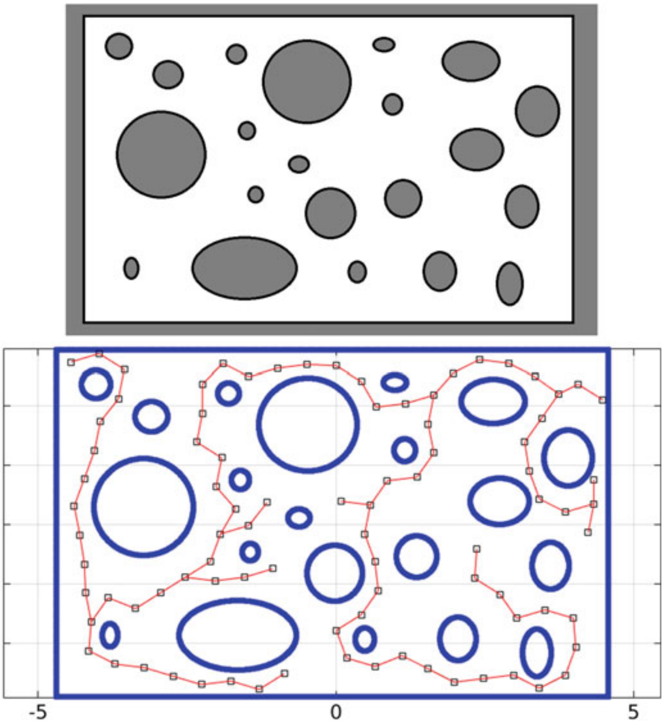
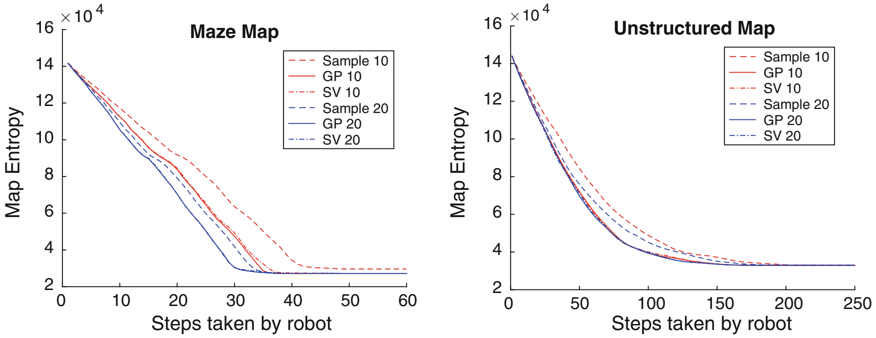


Fig. 4 The "unstructured" map used in our experiments is shown at *top*. The dimensions of the map are approximately 7 by 9 m. An occupancy map produced by exploration with GP regression, showing the trajectory of the robot, is given at *bottom*

5.2 Results

As noted in Sect. 4, the time consumed by the regression computations across the entirely of the action space is substantially less than evaluating the mutual information across the much smaller designated set of actions drawn from a Sobol sequence. Figure 5 gives results showing the performance of the six problem parameterizations over the maps of Figs. 3 and 4. In the maze map, both supervised learning methods drive down entropy faster than each respective case that chooses the most informative action from the explicitly evaluated sample set. In this case, all exploration methods nearly always select sensing actions from the same homotopy class, but choosing the approximately continuous action that is expected to be most informative, with the aid of supervised learning, tends to point the robot in a more advantageous direction than the most informative Sobol sample.

In the unstructured map, all parameterizations using Gaussian process and support vector regression perform better across the board, even when less computational effort is invested in establishing a training data set. In this case, the learning-based



(a) Results from the maze map of Fig. 3.

(b) Results from the unstructured map of Fig. 4.

Fig. 5 The results of 100 exploration trials randomly initialized in their respective maps, for each of six parameterizations. The mean entropy reduction is given over the number of sensing actions performed by the robot for all test cases considered

methods occasionally select actions from a different homotopy class than the competing method using explicitly computed mutual information only, resulting in fundamentally different paths among the different parameterizations. The use of supervised learning to select moves from the continuous space of sensing actions accumulates a more significant advantage, such that regression over 10 samples performs better than explicity evaluating the mutual information at 20 samples. Hence, more informative outcomes are selected with substantially less computational effort. Representative trajectories of the robot when using Gaussian process regression over 20 Sobol samples are given in Figs. 3 and 4, for each map. These trajectories represent full exploration of their respective environments, reaching the lower allowed limit on map entropy. Figure 6 gives representative examples of different exploration outcomes resulting from Gaussian process exploration, versus exploration using the sampled configurations only. Finally, Table 1 gives details on the computation time required, and the number of steps taken by the robot in the exploration process, for all examples implemented over the unstructured map of Fig. 4.

6 Conclusions and Future Work

We have proposed a novel approach to evaluate the mutual information throughout a robot's continuous action space, for the purpose of exploring a priori unknown environments. In the examples considered, supervised learning facilitates the selection of more informative sensing actions, in some cases selecting more informative actions with less overall computational effort.

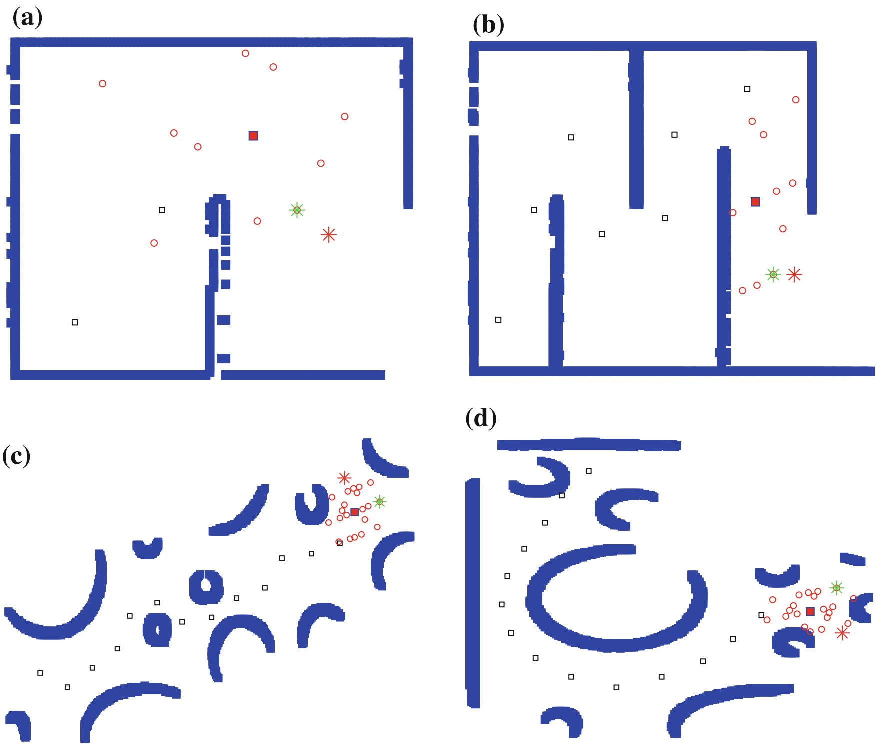


Fig. 6 Representative cases when Gaussian Process regression makes a more informative decision than the best sample whose information gain was explicitly evaluated. The *green star* represents the most informative Sobol sample, and the *red star* represents the action expected to be most informative from the Gaussian process regression. In cases **c** and **d** from the unstructured map, the candidate actions lie in different homotopy classes

6.1 Complex Action Spaces

Extending the approach of this paper to higher-dimensional systems and non-Euclidean action spaces is a compelling area for future work, in which we intend to consider regression over a robot's rotational degrees of freedom, as well as for differential systems capable of aggressively exploring their environments. The Matèrn kernel function shows promise in capturing sharp variations in expected mutual information across obstacle boundaries, and we intend to test its applicability to even more sharply varying action spaces in future work.

Acknowledgements The authors would like to thank Chengjiang Long from the Computer Science Department and Kiril Manchevski from Mechanical Engineering Department of Stevens Institute of Technology for help with computational resources.

Table 1 The results shown here are the average of 100 computational trials over the unstructured map shown in Fig. 4. For the six test cases examined (in which "DA" refers to the deterministic approach, derived from Sobol samples, "GP" refers to the Gaussian process approach, and "SV" refers to the support vector approach), at *top* we show a comparison of the mean and standard deviation of computation time required per sensing action, and at *bottom* we show a comparison of the mean and standard deviation of the total number of steps taken by the robot in the course of driving its entropy to the minimum designated value

Time cons. per step (s)	10 GP	10 SV	10 DA	20 GP	20 SV	20 DA
μ	2.18	2.18	2.19	3.83	3.79	3.79
σ	0.03	0.02	0.03	0.04	0.04	0.04
Steps taken per trial	10 GP	10 SV	10 DA	20 GP	20 SV	20 DA
μ	177.2	177.4	222.7	175.3	174.9	199.9
σ	4.87	5.06	5.65	5.11	5.14	5.58

References

1. Thrun, S., Burgard, W., Fox, D.: Exploration. Probabilistic Robotics, pp. 569–605. MIT Press, Cambridge (2005)
2. Elfes, A.: Robot Navigation: integrating perception, environmental constraints and task execution within a probabilistic framework. In: Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics, pp. 93–129 (1995)
3. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. Computer **22**(6), 46–57 (1989)
4. Julian, B.J., Karaman, S., Rus, D.: On mutual information-based control of range sensing robots for mapping applications. Int. J. Robot. Res. **33**(10), 1375–1392 (2014)
5. Rasmussen, C.E., Williams, C.K.I.: Gaussian Process for Machine Learning. MIT Press, Cambridge (2006)
6. Deisenroth, M.P., Fox, D., Rasmussen, C.E.: Gaussian processes for data-efficient learning in robotics and control. IEEE Trans. Pattern Anal. Mach. Intell. **37**(2), 408–423 (2015)
7. O'Callaghan, S.T., Ramos, F.T.: Gaussian process occupancy maps. Int. J. Robot. Res. **31**(1), 42–62 (2012)
8. Smola, A., Schlkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)
9. Whaite, P., Ferrie, F.P.: Autonomous exploration: driven by uncertainty. IEEE Trans. Pattern Anal. Mach. Intell. **19**(3), 193–205 (1997)
10. Bourgault, F., Makarenko, A.A., Williams, S.B., Grocholsky, B., Durrant-Whyte, H.F.: Information based adaptive robotic exploration. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 540–545 (2002)
11. Stachniss, C., Grisetti, G., Burgard, W.: Information gain-based exploration using rao-blackwellized particle filters. In: Proceedings of the Robotics: Science and Systems Conference, pp. 65–72 (2005)
12. Kim, A., Eustice, R.M.: Perception-driven navigation: active visual SLAM for robotic area coverage. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3196–3203 (2013)
13. Kollar, T., Roy, N.: Trajectory optimization using reinforcement learning for map exploration. Int. J. Robot. Res. **27**(2), 175–196 (2008)

14. Kollar, T., Roy, N.: Efficient optimization of information-theoretic exploration in SLAM. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1369–1375 (2008)
15. Yang, K., Gan, S.K., Sukkarieh, S.: A Gaussian process-based RRT planner for the exploration of unknown and cluttered environment with a UAV. Adv. Robot. **27**(6), 431–443 (2013)
16. Charrow, B., Kumar, V., Michael, N.: Approximate representations for multi-robot control policies that maximize mutual information. Auton. Robot. **37**(4), 383–400 (2014)
17. Wurm, K.M., Hennes, D., Holz, D., Rusu, R.B., Stachniss, C., Konolige, K., Burgard, W.: Hierarchies of octrees for efficient 3D mapping. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4249–4255 (2011)
18. Quin, P., Paul, G., Alempijevic, A., Liu, D., Dissanayake, G.: Efficient neighbourhood-based information gain approach for exploration of complex 3D environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1343–1348 (2013)
19. Deisenroth, M.P., Rasmussen, C.E., Peters, J.: Gaussian process dynamic programming. Neurocomputing **72**(7–9), 1508–1524 (2009)
20. Martinez-Cantin, R., de Freitas, N., Doucet, A., Castellanos, J.A.: Active policy learning for robot planning and exploration under uncertainty. In: Proceedings of the Robotics: Science and Systems Conference (2007)
21. Velez, J., Hemann, G., Huang, A.S., Posner, I., Roy, N.: Modelling observation correlations for active exploration and robust object detection. J. Artif. Intell. Res. **44**, 423–453 (2012)
22. Hollinger, G.A., Englot, B., Hover, F.S., Mitra, U., Sukhatme, G.S.: Active planning for underwater inspection and the benefit of adaptivity. Int. J. Robot. Res. **32**(1), 3–18 (2013)
23. Jadidi, M.G., Miro, J.V., Valencia, R., Andrade-Cetto, J.: Exploration on continuous Gaussian process frontier maps. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 6077–6082 (2014)
24. Shannon, C.E., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press, Champaign (1949)
25. Kim, S., Kim, J.: Continuous occupancy maps using overlapping local Gaussian processes. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4709–4714 (2013)
26. Sobol, I.M.: On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput. Math. Math. Phys. **7**(4), 86–112 (1967)
27. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
28. Chang, C., Lin, C.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), (2011)
29. Rasmussen, C.E., Nickisch, H.: Gaussian processes for machine learning (GPML) toolbox. J. Mach. Learn. Res. **11**, 3011–3015 (2010)

Relative Topometric Localization in Globally Inconsistent Maps

Mladen Mazuran, Federico Boniardi, Wolfram Burgard and Gian Diego Tipaldi

1 Introduction

Mobile robot localization is a well studied field in robotics and several robust approaches to localization have been proposed in the past [11, 12, 14, 19]. The majority of those approaches, however, assume that a globally consistent map of the environment is built beforehand by a mapping process. This map is then used to estimate the pose of the robot in a single absolute reference frame, often without taking into account the map uncertainties arising from the robot pose estimates during mapping.

Such maps are normally built by a separate mapping process, solving what is called the simultaneous localization and mapping (SLAM) problem. Many modern techniques for this problem are based on a least squares minimization approach over a graph of measurements [13], whose error is assumed to be normally distributed. In the presence of data association outliers, the Gaussian error assumption leads to maps that are not globally consistent. The need of a globally consistent map for localization led many researchers to either apply robust statistics in the minimization step [1, 18, 24, 30], or to measure the global consistency of the resulting maps [20].

In this paper we propose a novel paradigm to robot localization that relaxes the assumption of a globally consistent map and a single absolute reference frame. We believe that, for the majority of navigation tasks, a globally consistent map is not

M. Mazuran (✉) · F. Boniardi · W. Burgard · G.D. Tipaldi
University of Freiburg, 79110 Freiburg, Germany
e-mail: Mazuran@cs.uni-freiburg.de

F. Boniardi
e-mail: Boniardi@cs.uni-freiburg.de

W. Burgard
e-mail: Burgard@cs.uni-freiburg.de

G.D. Tipaldi
e-mail: Tipaldi@cs.uni-freiburg.de

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_25

necessary and one only needs global topological information and local metrical consistency. Our paradigm, *relative topometric localization* (RTL), is based on a graph representation of the environment, where each node represents a pose in the map and each edge their relative transformation. The graph implicitly defines a manifold structure with a chart associated to each node that parametrizes its local neighborhood. Such maps are not unusual in robotics, since they are the internal representation of modern mapping approaches based on pose-graph estimation or bundle adjustment. We also do not assume this map to be the result of any optimization process, nor to be globally embeddable in a single Euclidean space.

We formulate the localization problem as jointly estimating the current reference frame and the relative pose of the robot within its chart. Our paradigm has a set of advantages with respect to approaches based on a single absolute reference frame: (1) it is inherently robust to maps that are not globally consistent; (2) it includes uncertainties in the map estimate during localization; and (3) it operates on unoptimized maps, removing the need for a SLAM back-end.

We thoroughly evaluated our approach on a large set of experiments in fully simulated environments, environments sourced from real data, as well as on a real robot tracked with a motion capture system. To verify our claims, we compared our method with a Monte Carlo localization approach [26] on maps that have different levels of global inconsistency. In an additional experiment, we also evaluated our paradigm when the map is globally consistent but the environment changed its appearance (e.g., furniture was added or removed). The experimental results demonstrate that a relative topometric approach is indeed resilient to global inconsistencies in the map. Further, the method provides localization accuracy in the order of millimeters even under substantial changes in the environment or inconsistencies.

2 Related Work

Autonomous localization has been mainly addressed within probabilistic state estimation frameworks, with solutions based on extended Kalman filters (EKF) [19], histogram filters [12] or particle filters, often referred to as Monte-Carlo localization (MCL) [11]. Such approaches assume knowledge of a globally consistent map, without considering any uncertainty on it apart from the error induced by grid-based approximations. In this work, we relax the assumption of global consistency and we explicitly consider uncertainties in the map stemming from the SLAM process.

In the context of SLAM and bundle adjustment, several authors explored the concept of relative estimates. Howard et al. [2] introduced the idea of a manifold mapping for multiple robots. In their approach, the map is represented as a two-dimensional manifold embedded in a higher-dimensional space. They introduce key ideas of the manifold structure and present an application to multi-robot mapping. Sibley et al. [27] proposed the relative bundle adjustment paradigm. They claim that bundle adjustment is costly due to the choice of a single privileged reference frame and propose to optimize in a metric space defined by a manifold instead of a single Euclidean space. Their paradigm has been first extended by Blanco et al. [6], to consider a set of possible sparsification strategies, and by Strasdat et al. [29], to consider a second optimization window and to enforce a metric consistency within this optimization window. We differ from those works for the fact that we address a localization problem and also estimate a distribution over the reference frame. This is needed in order to include possible multiple localization hypotheses.

Churchill and Newman [9] introduced the concept of *navigation experiences*, i.e., robot paths with relative metrical information. They localize the robot in the experiences by first using appearance-based data association methods to estimate the initial node in the experience graph and then track the robot position using visual odometry techniques. In contrast to our framework, their approach does not consider uncertainties in the map, nor does it track the index of the reference frame over time.

Recently, the need of a globally consistent map for navigation has been questioned by some researchers. Konolige et al. [15] proposed a navigation system based on a hybrid metric-topological map. They employ a laser scanner and localize the robot with respect to one reference node in the graph. Dayoub et al. [10] extended the approach to cameras and considered a set of image features associated at each node in the graph. Our approach is closely related to these last two. The main difference is that we do not assume the graph to be the result of an optimization algorithm. Additionally, we are able to consider uncertainties in the map estimates.

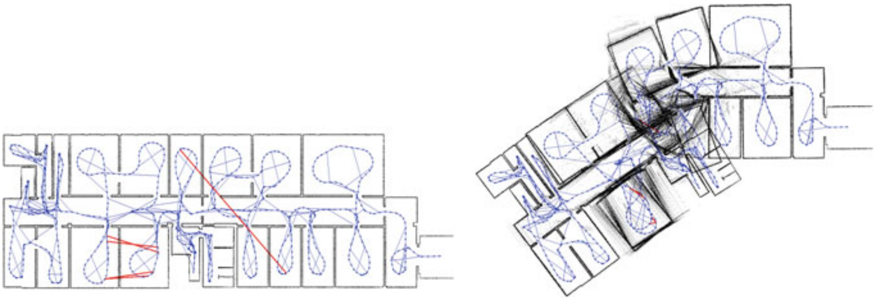
The concept of topometric localization has also been exploited by Badino et al. [3]. The authors build a topological map as a vehicle travels along its route. Each node is linked with a pose in the environment and the localization algorithm estimates which node the robot is currently in. The robot pose is then the pose stored in that node. Xu et al. [33] extended this to consider multiple roads and branchings along paths. Both approaches differ from ours, as the real estimation part is purely topological and the authors still rely on a global reference frame for the robot pose.

Ideas closely related to topometric localization are also present in teach-and-repeat frameworks. Sprunk et al. [28] proposed a data-driven approach using laser scanners and demonstrated millimeter-level accuracy without building any globally consistent map of the environment. McManus et al. [21] proposed an approach based on 3D laser reflectivity that is able to handle long-range navigation in challenging environments. The map is a chain of poses, where each pose has a submap associated to it. Localization is performed on the latter, relative to the corresponding pose, and a set of heuristics to switch submaps is presented. Krüsi et al. [16] build a similar metrical/topological map, but rely on ICP to perform localization on the submaps with a local motion planner to avoid obstacles. Our approach differs from teach and repeat paradigms since we do not localize only on a single route and we further propose a sound estimation framework for tracking the reference frame.

3 Relative Topometric Localization

In this section we describe the *relative topometric localization* paradigm and relate it to the *metric* one. In metric localization, one seeks to estimate the pose of the robot at each time step, given the map of the environment and the history of sensor measurements and controls. The main assumption is that the map is a globally consistent metric representation of the environment, consisting of the absolute position of relevant features in the environment (e.g., cells in an occupancy grid or a set of key-points for visual localization). The location of those features, as well as the pose of the robot, is expressed with respect to a single reference frame, which is the global coordinate frame of the map. Formally, this is equivalent to recursively estimating the posterior $p(\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, m)$, with robot pose $\mathbf{x}_t$, measurements $\mathbf{z}_{1:t}$, odometry readings $\mathbf{u}_{1:t}$[1] and map m.

This is often estimated by a finite set of particles, resulting in the well established field of Monte Carlo localization (MCL) [11]. MCL has been proven to be a robust approach for localization and is often deployed on robots when the map m is known a-priori. Nevertheless, estimating the pose of the robot in an absolute frame of reference implicitly assumes the map m to be globally consistent, an assumption that can easily be violated in the presence of outlier associations during the SLAM process. Such a scenario can especially occur if the map of the environment is automatically computed by the robot, without human post-processing, and the underlying SLAM system introduces wrong associations due to, say, perceptual aliasing [1]. Figure 1 showcases how the effect of a small amount of associations can impact the resulting map and consequently cripple the estimate of MCL in a large portion of it.



(a) Ground truth map

(b) Optimized map with outliers

Fig. 1 Effect of outliers in the association on the global consistency of the resulting map. Introducing just 5 outliers in the data association (marked in *red*) renders the central part of the map completely unusable for MCL

[1] Note that this is equivalent to the more traditional formulation with control inputs.

3.1 Relative Topometric Paradigm

In the relative topometric paradigm, we relax the assumption of a globally consistent metric map. As map representation, we follow an approach much akin to the one proposed by Howard et al. [2]. We consider the map of the environment to be a collection of patches that are locally homeomorphic to the Euclidean space, thus inducing a manifold structure, without the need of defining a global embedding.

More precisely, the map is a graph of poses $\mathbf{m}_i,\ i = 1, \ldots, N$ with a set of relative transformations $\mathbf{w}_{i,j} \in \mathrm{SE}(n)$ between them, and sensory data attached to them. Each node i in the graph defines a local chart of the manifold, of which $\mathbf{m}_i$ is the origin. Here, we can assume that the projection of the sensory data of the nodes in a neighborhood of i is roughly locally consistent. This, in turn, defines an open set in which the chart is valid.

While the chart is only locally valid, we can still express the poses of the remainder of the map in this reference system by chaining the transformations $\mathbf{w}_{i,j}$ along the minimum distance spanning tree of the whole graph, with farther nodes having increasingly erroneous relative position estimates.

Let $\mathbf{z}_t = \left[\mathbf{z}_t^{(1)}\, \mathbf{z}_t^{(2)} \ldots \mathbf{z}_t^{(N)}\right]$ be a vector of measurements at time t, where each $\mathbf{z}_t^{(i)}$ arises from matching the current observations of the robot with the observations stored in the node i of the graph. We wish to estimate at each time step t the vector of relative transformations between the poses $\mathbf{m}_i$ and the robot, namely $\varDelta\mathbf{x}_t = \left[\varDelta\mathbf{x}_t^{(1)}\, \varDelta\mathbf{x}_t^{(2)} \ldots \varDelta\mathbf{x}_t^{(N)}\right] \in \mathrm{SE}(n)^N$. In the ideal situation, where the manifold can be globally embedded, $\varDelta\mathbf{x}_t^{(i)} = \ominus\mathbf{m}_i \oplus \mathbf{x}_t$. Here, $\mathbf{m}_i$ and $\mathbf{x}_t$ respectively represent the pose of the node and of the robot in an absolute coordinate frame.

Formally, this is equivalent to jointly estimating the posterior over the reference frame r_t (i.e., the index of the current chart for the manifold) and the relative poses $\varDelta\mathbf{x}_t$ of the robot in that particular chart:

$$p(r_t, \varDelta\mathbf{x}_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w}) = p(\varDelta\mathbf{x}_t \mid r_t, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w})p(r_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w}). \tag{1}$$

The distribution is composed of a discrete probability mass, associated with a continuous probability density function, which we assume to be a Gaussian. The former represents the probability of the robot being within the open set in which the chart is valid, the latter represents the distribution of the robot pose in that specific chart. Following Bar-Shalom et al. [4], the estimation of (1) can be formulated within the *Dynamic Multiple Model Estimator* framework. In this framework, the goal is to estimate the state of a system that obeys one of a finite number of system models. Each model defines its own measurement and motion model and, during the estimation process, the system may change its model of operation. In our case, we can consider the choice of the reference frame as one model of our system, resulting in a finite number of models equivalent to the number of nodes in our graph. The change of model happens when the robot is not any longer in the valid set of the current chart and a new chart must be used.

Unfortunately, the full estimation problem requires tracking the whole sequence of frames, which results in Gaussian mixture distribution with an exponentially increasing number of terms. To reduce this complexity, approximated algorithms have been proposed, such as the Interacting Multiple Model [4], Markov Chain Monte Carlo [23], or Multiple Hypothesis Tracking [5].

In this paper, we focus on position tracking and we approximate the estimation of (1) by considering only the maximum likelihood frame at each time step. We believe that this is a valid approximation for position tracking problems, which is also confirmed by the experimental results presented below. We leave the full estimation problem as future work.

4 Position Tracking in the Relative Topometric Paradigm

This section describes the proposed approximation of (1) and its instantiation for position tracking. At each time step t, we approximate the distribution over the reference frames with a deterministic distribution, centered at its maximum $\hat{r}_t$, i.e.,

$$p(r_t \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w}) \approx \delta[r_t - \hat{r}_t], \tag{2}$$

where $\delta[r]$ denotes the discrete impulse function. To compute the next estimate of r_t we thus need to find the index i over the possible reference frames that maximizes

$$p(r_t = i \mid \mathbf{w}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \sum_{r_{t-1}} p(r_t = i \mid r_{t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w}) p(r_{t-1} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w}) \tag{3}$$

$$\propto p(\mathbf{z}_t \mid r_t = i, \hat{r}_{t-1}, \mathbf{u}_{1:t}, \mathbf{w}) p(r_t = i \mid \hat{r}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}, \mathbf{w}) \tag{4}$$

$$\approx p(\mathbf{z}_t \mid r_t = i, \hat{r}_{t-1}, \mathbf{u}_t, \mathbf{w}, \varDelta\hat{\mathbf{x}}_{t-1}) p(r_t = i \mid \hat{r}_{t-1}). \tag{5}$$

Here, $\varDelta\hat{\mathbf{x}}_{t-1}$, denotes the random variable associated to the posterior distribution of $\varDelta\mathbf{x}_{t-1}$ estimated at the previous time step. We assume $\varDelta\hat{\mathbf{x}}_{t-1}$ to be approximable by a Gaussian random variable with mean $\varDelta\bar{\mathbf{x}}_{t-1}$ and covariance $\boldsymbol{\Sigma}_{t-1}$. Note that from (3) to (4), we substituted approximation (2), applied Bayes' rule on $\mathbf{z}_t$, and used the independence assumption of the measurements. From (4) to (5) we followed the same approach of the *generalized pseudo-Bayesian estimator of first order* (GPB1) [4], namely, we assumed a transition model $p(r_t \mid r_{t-1})$ on the reference nodes, and used the previous displacement estimate for computing the measurement likelihood.

In this work we restrict ourselves to a uniform $p(r_t \mid r_{t-1})$ over all nodes in a local neighborhood of r_{t-1}, although this can be easily generalized to more elaborate models. As for the measurement likelihood, we computed it as follows. We first predict the relative pose of the robot in the reference frame $r_t = i$ by propagating the previous relative pose at time $t - 1$ from the reference $\hat{r}_{t-1}$ through the relative measurements $\mathbf{w}$ defined in the map. Under normality assumptions, the likelihood is then equal to $\mathcal{N}\left(\mathbf{z}_t - \hat{\mathbf{z}}_t; \mathbf{0}, \boldsymbol{\Sigma}\right)$, where $\hat{\mathbf{z}}_t$ is the predicted observation relative to the reference frame $r_t = i$ and $\boldsymbol{\Sigma}$ is the innovation covariance matrix.

In the case of range sensors, which is the implementation target of this paper, the map includes a laser scan per node, the predicted observation is the relative pose of the robot in the reference frame of r_t, while the current observation can be the result of an ICP procedure [8].

Let $\mathbf{m}$ denote the stacked $\mathbf{m}_i$ poses, in the local neighborhood of r_t, say V_t, for which the chart associated to r_t is valid, as well as the pose $\mathbf{m}_s$ with $s = \hat{r}_{t-1}$. Then, to compute the distribution over the robot pose, we express (1) as the marginal over $\mathbf{m}$ and $\Delta\mathbf{x}_{t-1}$. Given the approximation defined in (2), we compute the distribution

$$p(\Delta\mathbf{x}_t \mid \hat{r}_t, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w}) = \iint p(\Delta\mathbf{x}_t, \Delta\mathbf{x}_{t-1}, \mathbf{m} \mid \hat{r}_t, \hat{r}_{t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}, \mathbf{w})\, \mathrm{d}\Delta\mathbf{x}_{t-1}\mathrm{d}\mathbf{m} \tag{6}$$

Under the assumption that the conditional distribution of $\Delta\mathbf{x}_t$ can be approximated reasonably by a normal distribution, we can estimate the mean $\Delta\bar{\mathbf{x}}_t$ of the marginal by maximum likelihood inference over the joint space of the relative poses $\Delta\mathbf{x}_t$, the map nodes $\mathbf{m}$, and the previous relative poses $\Delta\mathbf{x}_{t-1}$. The marginalization step is performed by simply extracting the values corresponding to $\Delta\mathbf{x}_t$ from the joint mean vector. The covariance $\boldsymbol{\Sigma}_{t-1}$, on the other hand, can be computed by linear error propagation through the optimization algorithm.

Note that the choice of reference frame (i.e., the chart) introduces a *map locality* due to the domain of the homeomorphism, implicitly discarding nodes and measurements of the graph not included in the domain.

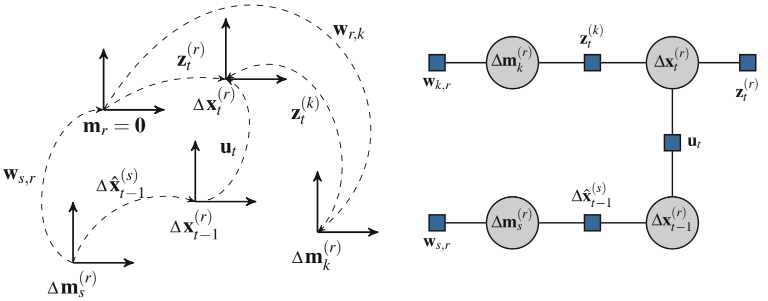
Thus, dropping $\hat{r}_t$ and $\hat{r}_{t-1}$ for simplicity, the joint likelihood is proportional to

$$p(\mathbf{z}_t|\Delta\mathbf{x}_t, \mathbf{m})p(\mathbf{w} \mid \mathbf{m})p(\mathbf{u}_t \mid \Delta\mathbf{x}_t, \Delta\mathbf{x}_{t-1})p(\Delta\hat{\mathbf{x}}_{t-1} \mid \mathbf{m}, \Delta\mathbf{x}_{t-1}), \tag{7}$$

where we assumed $\mathbf{z}_t$, $\mathbf{w}$, and $\mathbf{u}_t$ to be conditionally independent. Theoretically, we cannot assume independence between $\mathbf{w}$ and $\Delta\hat{\mathbf{x}}_{t-1}$, since $\Delta\hat{\mathbf{x}}_{t-1}$ was estimated at the previous time step by using $\mathbf{w}$ as well. To avoid overconfidence due to this correlation, the resulting covariance estimate should be corrected. The correction factor can be computed following the approach of Mourikis and Roumeliotis [22]. We, however, experimentally found its effect to be negligible.

For ease of notation we will henceforth refer to $\hat{r}_t$ as r and $\hat{r}_{t-1}$ as s. Under the chart of r, the new reference node acts as the origin of its homeomorphic Euclidean space, with respect to which its neighborhood is expressed. We can thus consider a set of relative map pose coordinates $\Delta\mathbf{m}_i^{(r)}$ $\forall i \in V_t$, which in the ideal case would be equivalent to $\ominus\mathbf{m}_r \oplus \mathbf{m}_i$.

To simplify the understanding of the conditional independence structure of the likelihood in (7), we provide in Fig. 2a the reference frames at work in the estimation problem. Here, $k \in V_t$, while the observations $\mathbf{z}_t$ are assumed to provide SE(n)


$\mathbf{w}_{r,k}$
$\mathbf{z}_t^{(r)}$
$\Delta\mathbf{x}_t^{(r)}$
$\mathbf{z}_t^{(k)}$
$\mathbf{m}_r = \mathbf{0}$
$\mathbf{u}_t$
$\mathbf{w}_{s,r}$
$\Delta\hat{\mathbf{x}}_{t-1}^{(s)}$
$\Delta\mathbf{x}_{t-1}^{(r)}$
$\Delta\mathbf{m}_k^{(r)}$
$\Delta\mathbf{m}_s^{(r)}$
$\mathbf{w}_{k,r}$
$\Delta\mathbf{m}_k^{(r)}$
$\mathbf{z}_t^{(k)}$
$\Delta\mathbf{x}_t^{(r)}$
$\mathbf{z}_t^{(r)}$
$\mathbf{u}_t$
$\Delta\hat{\mathbf{x}}_{t-1}^{(s)}$
$\Delta\mathbf{m}_s^{(r)}$
$\Delta\mathbf{x}_{t-1}^{(r)}$
$\mathbf{w}_{s,r}$

(a) Frames considered (b) Corresponding factor graph

Fig. 2 Frames and relative transformations involved in the small-scale estimation the robot's relative pose (*left*) and the factor graph associated to the same problem (*right*)

relative measurements, for example via ICP matching. In the general formulation, the latter is not required, and we commit to it only for the sake of clarity.

Under the assumption that $\mathbf{z}_t$, $\mathbf{u_t}$, and $\mathbf{w}$ are normally distributed, we can maximize the likelihood (7) by performing nonlinear least squares optimization on a reduced factor graph. Specifically, we construct a graph which has as vertices $\Delta\mathbf{m}_s^{(r)}$, $\Delta\mathbf{m}_k^{(r)}$ $\forall k \in V_t$, $\Delta\mathbf{x}_{t-1}^{(r)}$ and $\Delta\mathbf{x}_t^{(r)}$. The factors we introduce are respectively all the valid measurements $\mathbf{z}_t^{(i)}$, the odometry reading $\mathbf{u}_t$, the previous estimate $\Delta\hat{\mathbf{x}}_{t-1}^{(s)}$, and all $\mathbf{w}_{i,j}$ connecting the map poses that have been introduced in the factor graph.

Note that there needs to exist a factor correlating $\mathbf{m}_s^{(r)}$ to the remaining map vertices, as otherwise the previous estimate and the odometry reading would be neglected. If no such factor exists, it can be computed by linear error propagation.

Finally, we can compute the mean $\Delta\bar{\mathbf{x}}_t^{(r)}$ of the next estimate by setting the current reference node to zero, conditioning the factor graph on it, and optimizing with respect to the remaining vertices. The resulting factor graph associated to the example in Fig. 2a is reported in Fig. 2b. Since we are conditioning with respect to the current reference node, the factors that connected it have become priors on the other vertices. Note that in this formulation, there is no assumption on $\mathbf{z}_t$ and it does not necessarily need to be an $\mathrm{SE}(n)$ measurement, however, $\mathbf{z}_t$ should be such as to allow the overall problem to be non-singular and observable.

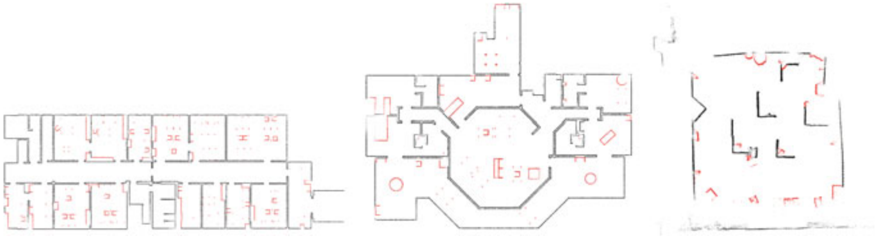
Having found the mean $\Delta\bar{\mathbf{x}}_t^{(r)}$ by nonlinear least squares optimization, the estimation of the robot's relative position to the remaining map poses is split over two methods. For the map poses $\Delta\mathbf{m}_k^{(r)}$ that appeared in the factor graph optimization, we use the resulting mean estimate $\Delta\bar{\mathbf{m}}_k^{(r)}$ and compute $\Delta\bar{\mathbf{x}}_t^{(k)} = \ominus\Delta\bar{\mathbf{m}}_k^{(r)} \oplus \Delta\bar{\mathbf{x}}_t^{(r)}$. For the remaining map poses we compute the relative estimate by chaining together $\Delta\bar{\mathbf{x}}_t^{(r)}$ and all the transformations from the current reference node to the target map pose, over the minimum distance spanning tree of the full map (in this paper we used uniform weighting). Notice that with this formulation we explicitly compute only the uncertainty of $\Delta\mathbf{x}_t^{(r)}$, and we keep all other uncertainties as implicitly defined through linear error propagation.

5 Experiments

We evaluated our relative topometric localization approach (RTL) on a 2D localization problem with range measurements. We considered virtual measurements computed by the Canonical Scan Matcher (CSM) with point-to-line metric [8] and its first order covariance approximation [7]. We further rely on g^{2}o [17] for the least squares optimization and on our MCL implementation for comparison [26].

We evaluated our approach in both simulation and real data. For a more realistic simulation, we also sourced laser data from real datasets, as performed by Olson [25]. For the simulation data, we employed the Gazebo simulator on an environment based on two real floor plans. For each environment we kept the unfurnished version, and created a further one with added obstacles and furniture. Figure 3a, b show the two environments in question, with the additional obstacles and furniture marked in red. For each of the four maps we recorded one mapping run and ten localization runs connecting distant areas in the environment. For the pseudo-simulated data, we tested against the Intel Seattle, Intel Oregon, MIT CSAIL, and ACES datasets, which are publicly available. As ground truth, we used the aligned SLAM output and computed scans by casting rays on this aligned map. As with the simulated datasets, we recorded one mapping run and ten localization runs per configuration. In the above cases, we computed laser scans with 360° of field of vision and 360 rays. Each ray was corrupted with Gaussian noise with 1 cm of standard deviation and rounded to the closest centimeter.

For the real robot experiment we created an environment in our lab, both with and without additional obstacles. We used a KUKA omnirob, equipped with two Sick S300 laser scanners with 541 beams and 270° of field of vision. The ground truth was approximated by using a motion capture system with 10 Raptor-E cameras recording at 300 Hz. Again, we recorded one mapping run and 10 localization runs per configuration. To synchronize the data from the motion capture and the robot, we periodically stopped the robot and we only considered the poses in which the robot was stopped for both evaluation and mapping.



(a) Simulated environment 1 (b) Simulated environment 2 (c) Real environment

Fig. 3 Environments considered in the experiments in addition to the more common SLAM datasets. The objects marked in *red* do not appear in the unfurnished counterparts

The motion capture may also introduce errors due to the not perfectly even floor, limited view-points of the cameras, and imperfect calibrations. Figure 3c displays the recorded laser scans aligned according to the motion capture as well as the additional obstacles that were introduced in red. Note that the map is imperfect due to the aforementioned issues, as such the quality of the evaluation for the real data is significantly lower than for the simulation datasets.

We compared our approach, RTL, with MCL as well as two additional benchmark approaches: a relative approach similar to RTL, where we assume the map is the result of an optimization process and is not subject to errors, (*LSLOC*) and MCL on the ground-truth map (*MCLGT*). Note that LSLOC, is equivalent in spirit to the approach of Konolige et al. [15] and Dayoub et al. [10]. For MCL and MCLGT we used 5000 particles, as measurement model we used likelihood fields saturated to 2 m of maximum distance, and rendered the map at 1 cm of resolution for maximum accuracy. We experimentally found that resolutions greater than 1 cm did not improve the estimate. For each dataset, we use the same unoptimized pose graph as input to RTL and for computing the globally consistent map for MCL and LSLOC.

We consider three localization scenarios: *Standard localization*, where the given maps are globally consistent; *Localization with outliers*, where the maps have different levels of inconsistency; and *Furnished versus unfurnished localization*, where maps are globally consistent and we moved furniture between the mapping and localization phase. Since we are considering a tracking problem, we initialized the starting pose of all methods with the ground truth.

For the outlier case, we considered one simulated environment and the Intel Seattle dataset with two levels of outliers, respectively 5 and 20. We generate those outliers by associating the most likely matchings returned by FLIRT [31] that were at least at 3 m of distance. A sample map with five outliers is reported in Fig. 1.

5.1 Evaluation Criteria

We evaluate the localization accuracy at each time step t by measuring the discrepancy $\boldsymbol{\varepsilon}_t^{(i)} = \ominus \breve{\mathbf{x}}_t \oplus \breve{\mathbf{m}}_i \oplus \Delta \mathbf{x}_t^{(i)}$ between the predicted relative displacements and the actual ones, for each reference frame i. Here $\breve{\mathbf{x}}_t$ and $\breve{\mathbf{m}}_i$ respectively refer to the ground-truth pose of the robot at time t and of the i-th element of the map.

We compute the error statistics according to the robot's actual distance from the reference frame, in order to quantify the accuracy of all the methods at short distances, which are of interest to the robot, but also at progressively larger distances. We compute the shortest distances from the robot to the map poses by means of Dijkstra's algorithm on the rasterized ground-truth map, with 1 cm resolution. For each time step t we bin the absolute values of the errors $\boldsymbol{\varepsilon}_t^{(i)}$ in terms of distance and use as statistics the 5th, 25th, 50th, 75th, and 95th percentile over the whole trajectory of all runs. When computing these statistics we do not take into account runs that diverged; we assume a method to diverge on any particular run if the average absolute error was greater than 1 m in translation or 60° in rotation in the last 10% of

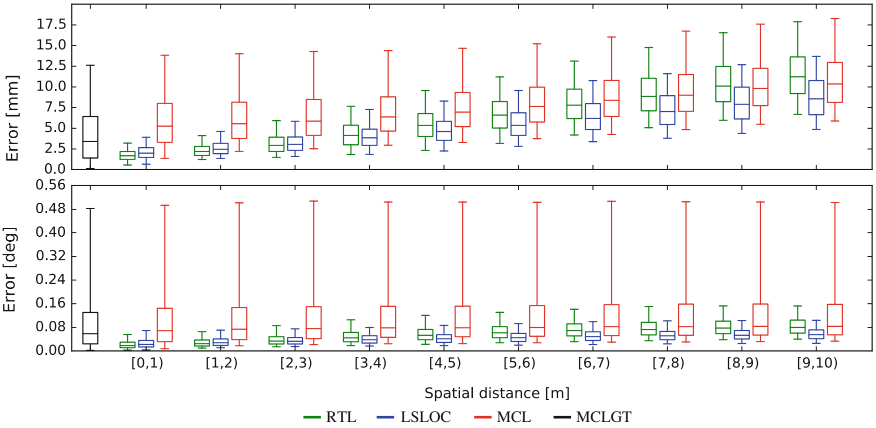


Fig. 4 Box plot of the errors in translation and rotation for the standard test scenario on the simulated datasets. The data was gathered over 40 localization runs

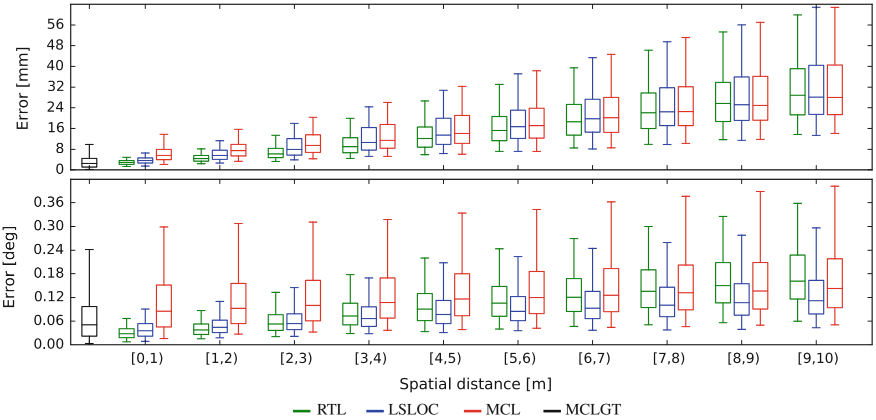


Fig. 5 Box plot of the errors in translation and rotation for the standard test scenario on the pseudo-simulated datasets. The data was gathered over 40 localization runs

the trajectory. Since MCLGT uses the ground-truth map for localization the statistics are independent of the distance, we thus report for it a single set of statistics.

5.2 Accuracy Results

We report in Figs. 4 and 5 the errors in terms of translation and rotation for the simulated and pseudo-simulated runs in the standard localization scenario. The box plots are binned at 1 m of discretization, where the $[a, b)$ values refer to all distances

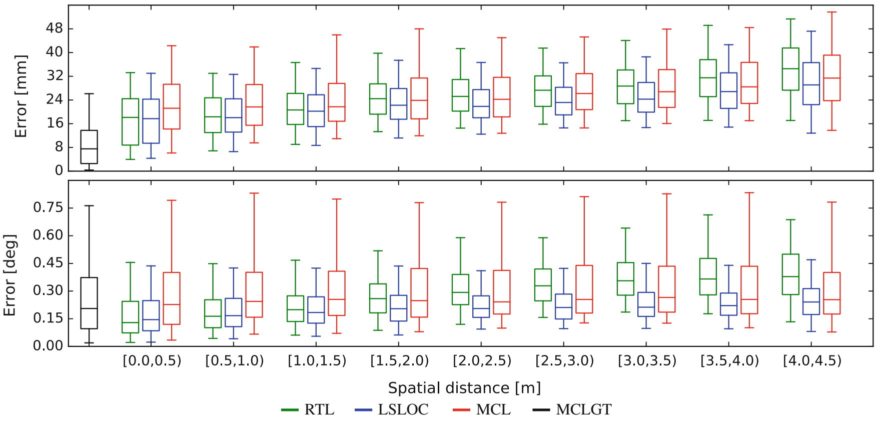


Fig. 6 Box plot of the errors in translation and rotation for the standard test scenario on the real datasets. The data was gathered over 20 localization runs

between *a* and *b*. The errors for the pseudo-simulated datasets are indeed larger than the simulated ones, which suggest an improved realism in the data sourcing.

As expected, the errors increase with respect to the distance for all methods, as even the most accurate SLAM map is subject to incremental error. At larger distances the error of RTL increases more as the local chart is increasingly inaccurate. This is to be expected and is the trade-off between local and global accuracy. RTL and LSLOC consistently achieve lower errors than MCL, particularly for orientation, and at short distances improve even on MCLGT. Nevertheless, given the scale of the graphs (millimeters and fractions of degrees) such differences are negligible.

Figure 6 reports the accuracy results for the real datasets in the standard localization scenario. The errors are larger than both the pseudo-sumulated and simulated datasets, partially due to the inaccuracy of the motion capture framework, the various unmodeled errors, and possibly due to the greater number of outliers in the laser range readings. Contrary to the previous tests, there is no clear preference to which method is most accurate. These results confirm the practical effectiveness of MCL when its assumptions are satisfied, with RTL showing competitive performance.

Figure 7 shows the accuracy results for the localization scenario with 5 outliers. LSLOC and MCL respectively diverged 18 and 17 times out of 20 localization runs, and even when converged the methods exhibit significantly large errors in both position and orientation. This result showcases the brittleness of traditional approaches which rely on the assumption of a globally consistent map. On the contrary, RTL is virtually unaffected by the outliers up to approximately 4 m of distance. At longer distances the relative position estimates may be obtained by chaining estimates through the outliers, which significantly degrades the accuracy. Similarly, Fig. 8 reports the accuracy results for the localization scenario with 20 outliers. LSLOC and MCL diverged in all runs, while RTL converged in all instances and shows similar performance. Although we have observed no such instances, in principle, RTL might be

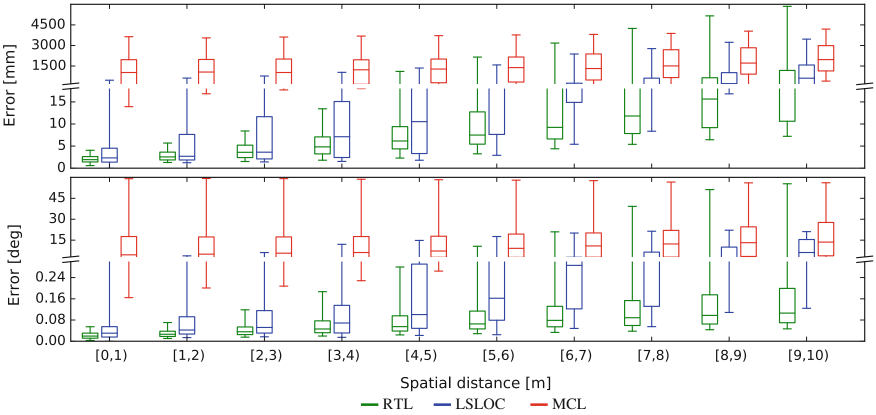


Fig. 7 Box plot of the errors in translation and rotation for the test scenario on localization with 5 outliers. The data was gathered over 20 runs, with LSLOC having diverged 18 times and MCL 17 times. The error axis is broken in two scales, in order to display both small and large errors

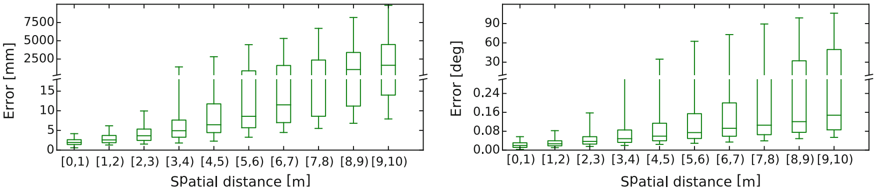


Fig. 8 Box plot of the errors in translation and rotation of RTL for the test scenario on localization with 20 outliers. The data was gathered over 20 runs. RTL converged in all instances, the other methods diverged. The error axis is broken in two scales, to display both small and large errors

also trapped in a wrong reference frame and exhibit divergence, since we are committing on a single estimate for the reference frame. This problem can be counteracted by considering a distribution over the reference frames.

Figures 9 and 10 report the errors in localization for the furnished versus unfurnished test scenario, respectively on simulated and real datasets. For the simulated data MCL diverged in 4 instances, while MCLGT in 7.

The MCL methods show significantly degraded accuracy when localizing on a map which does not reflect the structure of the perceived environment, as also shown in the literature [32]. We conjecture it is due to the fact that truncated likelihood fields and beam-based observation models are not robust enough to non-negligible discrepancies between the map and the observed environment. For future works it would be of interest to explore the conditions under which MCL results in degraded estimates and if it is possible to improve the observation model to account for them.

RTL and LSLOC, on the other hand, are virtually unaffected by the presence or absence of furniture. We believe this is due to the robustness of the ICP variant implemented in CSM and due to the fact that the localization is constrained to follow

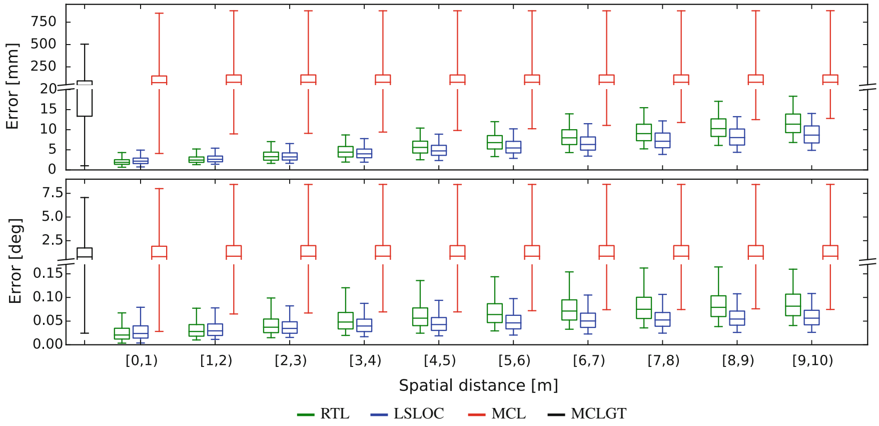


Fig. 9 Box plot of the errors in translation and rotation for the furnished versus unfurnished test scenario on simulated datasets. The data was gathered over 40 runs. MCL diverged in 4 instances, MCLGT in 7. The error axis is broken in two scales, in order to display both small and large errors

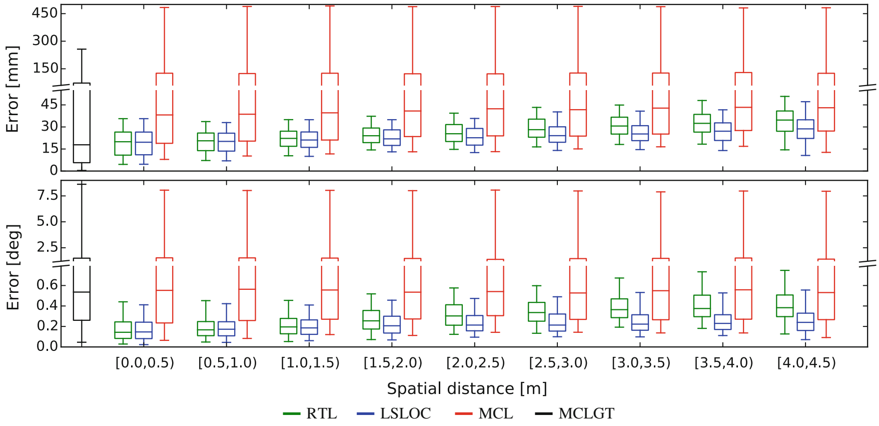


Fig. 10 Box plot of the errors in translation and rotation for the furnished versus unfurnished test scenario on real datasets. The data was gathered over 20 runs. The error axis is broken in two scales, in order to display both small and large errors

the connectivity of the graph and cannot "cross" a room wall, which can indeed happen with a standard variant of MCL.

5.3 Timing Results

In our implementation, each localization step requires roughly 220 ms of time on a multithreaded Intel® Core™ i7-3770K CPU clocked at 3.50 GHz. By comparison, MCL with 5000 particles requires approximately 90 ms. The current bottleneck is the number of ICP matchings executed per time step. This can be drastically reduced by introducing a better transition model $p(r_t \mid r_{t-1})$, which is currently merely uniform.

6 Conclusion

In this paper we introduced a novel localization approach, *relative topometric localization*, that relaxes the assumption of global consistency of the map used for localization. We represent the map as a pose graph endowed with sensory data for each node, which induces a manifold-like structure that, contrary to a globally consistent map, is not required to be fully embeddable in a Euclidean space. We reformulated the localization problem as estimating the topological node on the graph and a relative metric rigid body transformation. In doing so we obtained a method that is both robust to outliers in the map, takes into account the uncertainty in the mapping process, and that is independent of whether the graph has been optimized or not. We showed through extensive evaluation on both simulated and real data that our method not only significantly improves the state of the art in terms of localization accuracy when the global map consistency assumption fails, but also that for short relative distances it is equally accurate, or better, than MCL for the nominal cases.

Acknowledgements This work has been partially supported by the European Commission under the grant numbers ERC-AG-PE7-267686-LIFENAV, FP7-610603-EUROPA2, and H2020-645403-ROBDREAM.

References

1. Agarwal, P., Tipaldi, G., Spinello, L., Stachniss, C., Burgard, W.: Robust map optimization using dynamic covariance scaling. In: Proceedings of the IEEE International Conference on Robotics and Automation (2013)
2. Andrew Howard, G.S.S., Matarić, M.J.: Multi-robot mapping using manifold representations. Proc. IEEE Special Issue Multi-robot Syst. **94**(9), 1360–1369 (2006)
3. Badino, H., Huber, D., Kanade, T.: Real-time topometric localization. In: Proceedings of the IEEE International Conferance on Robotics and Automation (2012)
4. Bar-Shalom, Y., Li, X.R., Kirubarajan, T.: Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software. Wiley, New York (2004)
5. Blackman, S.S.: Multiple hypothesis tracking for multiple target tracking. IEEE Aerospace Electron. Syst. Mag. **19**(1), 5–18 (2004)
6. Blanco, J.L., González-Jiménez, J., Fernandez-Madrigal, J.A.: Sparser relative bundle adjustment (SRBA): constant-time maintenance and local optimization of arbitrarily large maps. In: Proceedings of the IEEE International Conference on Robotics and Automation (2013)
7. Censi, A.: An accurate closed-form estimate of ICP's covariance. In: Proceedings of the IEEE International Conference on Robotics and Automation (2007)
8. Censi, A.: An ICP variant using a point-to-line metric. In: Proceedings of the IEEE International Conference on Robotics and Automation (2008)

9. Churchill, W., Newman, P.: Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (2012)
10. Dayoub, F., Morris, T., Upcroft, B., Corke, P.: Vision-only autonomous navigation using topometric maps. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2013)
11. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte Carlo localization for mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (1999)
12. Fox, D., Burgard, W., Thrun, S.: Markov localization for mobile robots in dynamic environments. J. Artif. Intell. Res. **11**, 391–427 (1999)
13. Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based SLAM. IEEE Trans. Intell. Transp. Syst. Mag. **2**, 31–43 (2010)
14. Jensfelt, P., Austin, D., Wijk, O., Andersonn, M.: Feature based condensation for mobile robot localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (2000)
15. Konolige, K., Marder-Eppstein, E., Marthi, B.: Navigation in hybrid metric-topological maps. In: Proceedings of the IEEE International Conference on Robotics and Automation (2011)
16. Krüsi, P., Bücheler, B., Pomerleau, F., Schwesinger, U., Siegwart, R., Furgale, P.: Lighting-invariant adaptive route following using iterative closest point matching. J. Field Robot. **32**, 534–564 (2014)
17. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation (2011)
18. Latif, Y., Cadena, C., Neira, J.: Robust loop closing over time. In: Proceedings of Robotics: Science and Systems (2012)
19. Leonard, J., Durrant-Whyte, H.: Mobile robot localization by tracking geometric beacons. IEEE Trans. Robot. **7**(4), 376–382 (1991)
20. Mazuran, M., Tipaldi, G.D., Spinello, L., Burgard, W., Stachniss, C.: A statistical measure for map consistency in SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation (2014)
21. McManus, C., Furgale, P., Stenning, B., Barfoot, T.D.: Lighting-invariant visual teach and repeat using appearance-based lidar. J. Field Robot. **30**, 254–287 (2013)
22. Mourikis, A.I., Roumeliotis, S.I.: On the treatment of relative-pose measurements for mobile robot localization. In: Proceedings of the IEEE International Conference on Robotics and Automation (2006)
23. Oh, S., Russell, S., Sastry, S.: Markov chain monte carlo data association for multi-target tracking. IEEE Trans. Autom. Control **54**(3), 481–497 (2009)
24. Olson, E., Agarwal, P.: Inference on networks of mixtures for robust robot mapping. Int. J. Robot. Res. **32**, 826–840 (2013)
25. Olson, E.B.: Real-time correlative scan matching. In: Proceedings of the IEEE International Conference on Robotics and Automation (2009)
26. Röwekämper, J., Sprunk, C., Tipaldi, G.D., Stachniss, C., Pfaff, P., Burgard, W.: On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2012)
27. Sibley, G., Mei, C., Reid, I., Newman, P.: Adaptive relative bundle adjustment. In: Proceedings of Robotics: Science and Systems (2009)
28. Sprunk, C., Tipaldi, G.D., Cherubini, A., Burgard, W.: Lidar-based teach-and-repeat of mobile robot trajectories. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2013)
29. Strasdat, H., Davison, A.J., Montiel, J., Konolige, K.: Double window optimisation for constant time visual SLAM. In: IEEE International Conference on Computer Vision (2011)
30. Sünderhauf, N., Protzel, P.: Switchable constraints for robust pose graph SLAM. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2012)

31. Tipaldi, G.D., Arras, K.O.: FLIRT - interest regions for 2D range data. In: Proceedings of the IEEE International Conference on Robotics and Automation (2010)
32. Tipaldi, G.D., Meyer-Delius, D., Burgard, W.: Lifelong localization in changing environments. Int. J. Robot. Res. **32**(14), 1662–1678 (2013)
33. Xu, D., Badino, H., Huber, D.: Topometric localization on a road network. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (2014)

Fast Sample-Based Planning for Dynamic Systems by Zero-Control Linearization-Based Steering

Timothy M. Caldwell and Nikolaus Correll

1 Introduction

Planning for dynamic systems is trajectory exploration through nonconvex state spaces. These non-convexities often include configuration space obstacles and regions of inevitable collision [12]. While sample-based planning algorithms differ in many ways, they commonly conduct many short trajectory explorations to connect or nearly connect a large number of stochastically sampled states with the goal of connecting a start state to a final state or goal region. Earlier algorithms like RRT [12] guarantee under certain conditions [11] with probability one that if a connecting trajectory exists, that the algorithm will find it. Newer planners like RRT*, PRM*, SST*, and FMT* [8–10, 13, 14] guarantee asymptotic convergence to the globally optimal connecting trajectory. For planning dynamic systems, [16] directly tackles the theoretical issues associated with the differential constraints. However, planners for dynamic systems are still in need of computationally efficient methods to *compute distances*, e.g. for assessing nearest neighbors, and to *steer*, i.e. to extend to sampled states. Furthermore, each method must be able to handle numerical issues like sensitivities to initial conditions so that the steering trajectories can be of significant time horizon.

The methods proposed in this paper are complementary to [2, 3, 15, 17]: For planning dynamic systems, distance computation and steering are solutions to optimal control problems. They are often approximated through linearizing the nonlinear dynamics and solving linear quadratic regulation (LQR) or tracking (LQT) problems [2, 3, 15]. Each [2, 3, 15] linearize around a single state and so the system is time-invariant. In comparison, we linearize around the zero-control or “free” trajectory

T.M. Caldwell (✉) · N. Correll
University of Colorado, Boulder, CO 80309, USA
e-mail: timothy.caldwell@colorado.edu

N. Correll
e-mail: ncorrell@colorado.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_26

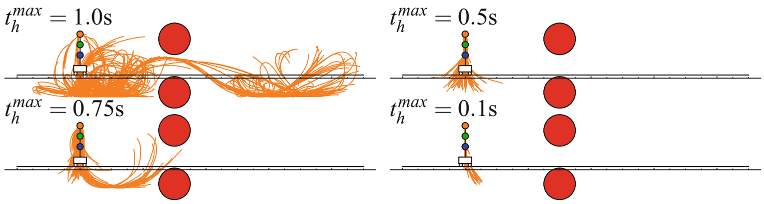


Fig. 1 Trajectories of a triple pendulum on a cart planned using an RRT-like approach with differing max time horizons for a constant number of vertices (1000). Planning with longer time horizons covers significantly more search space for the same number of vertices, at the expense of increased number of insertion failures due to collisions. In this example, computing for $t_h^{max} = 1.0\,\text{s}$ takes around three times longer than for $t_h^{max} = 0.1\,\text{s}$

so that the linearization remains a good approximation for longer time horizons. Since our approximate system is time-varying we formulate the LQR and LQT so that through precomputation and caching they can be solved efficiently for many distance and steering executions from the same state.

Both [2, 17] set up the linear quadratic optimal control problem as an optimal state transfer without a running state cost. As [5] indicates, the solution to such a problem is open-loop, which can be numerically intractable as the conditioning of the reachability Gramian becomes poor. Indeed, we show that the reachability Gramian's condition number approaches machine precision for the n-link pendulum on a cart in the examples, and as such, the methods in [2, 17] are infeasible. Instead, we propose implementing a feedback loop and performing the same state transfer procedure on the closed-loop system, which has a reachability Gramian with improved conditioning.

The state and trajectory solutions to the LQR problems are infeasible—i.e. trajectories of a linear system can not be trajectories of a nonlinear system. In order to steer, we implement the trajectory functional projection operator in [4, 6] to project approximate trajectories to feasible trajectories.

The proposed distance computation and steering methods in the paper are contributions to general planners. The methods are designed so that long time horizon exploring is viable. This is reflected in the examples where a small number of vertices of an RRT are needed to plan a 3-link pendulum on a cart through a corridor of obstacles (Fig. 1). The pendulum on a cart is under-actuated, unstable and when uncontrolled is chaotic. To the best of our knowledge, no one has planned a 3-link or greater pendulum on a cart.

2 Review of Planning for Dynamic Systems

We briefly review planning for dynamic systems. The kinodynamic planning problem is to find a state and control trajectory (x, u) that connects a start state x_{start} to a goal region while maintaining the differential constraint of the system dynamics

$$\dot{x}(t) = f(x(t), u(t)). \tag{1}$$

If a trajectory satisfies the system dynamics, then we say it is feasible. Sample-based planning algorithms find a path by generating a graph $G = (V, \mathscr{E})$ where the vertices V are explored states and the edges $\mathscr{E}$ are state and control trajectories that connect two states in V—i.e. $(x, u; t_h) \in \mathscr{E}$ implies for time horizon $t_h > 0$, $x(0) \in V$, and $x(t_h) \in V$. Every edge must be dynamically feasible in that the state and control must satisfy Eq. 1. Additionally, the state and control must remain in the set of *allowable state and control* (X, U) throughout the full trajectory where $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$. The set X is free of all obstacles.

Methods based on RRT build the graph through two functions: nearest neighbor and steering. The nearest neighbor calculation relies on the choice of distance function between a state $x_0 \in V$ and a sampled state $x_{samp} \in X$. Ideally, this distance is the cost J to transfer x_0 to x_{samp}. The nearest state is the $x_0 \in V$ with least distance to x_{samp}. Steering computes a trajectory that satisfies Eq. 1 and transfers the system from an initial state $x_0 \in V$ to a neighborhood of a desired state $x_{samp} \in X$. Such a problem can be treated as an optimal control problem which finds a trajectory that minimizes some cost J. The cost for computing distance and steering can be the same. We consider the following cost function:

$$J(x, u; t_h) := \frac{1}{2}\int_0^{t_h} u^T(\tau)R(\tau)u(\tau)\, d\tau + \frac{1}{2}(x(t_h) - x_{samp})^T P_1 (x(t_h) - x_{samp}) \tag{2}$$

where $R = R > 0$ is symmetric positive definite and $P_1 = P_1^T \geq 0$ is symmetric positive semi-definite. With a little work, the results in this paper can be extended to general costs as long as the cost can be locally approximated by a quadratic.

The problem of minimizing J constrained to the dynamics Eq. 1 is a nonlinear optimal control problem for which numerical methods can be slow. For this reason, we approximate the dynamics by linearizing about one of two candidates. The candidate used in [2, 3, 15, 17] is simply a point $x_0 \in \mathbb{R}^n$. The second candidate, which we propose, is the zero-control trajectory $x_{zero}(t)$ and is the solution to:

$$\dot{x}_{zero}(t) = f(x_{zero}(t), 0), \text{ s.t. } x_{zero}(0) = x_0. \tag{3}$$

Let $x_T(t) = x_0$ or $x_{zero}(t)$ depending on the candidate chosen. The linear terms are $A(t) = \frac{\partial}{\partial x(t)} f(x(t), u(t))|_{(x_T(t),0)}$ and $B(t) = \frac{\partial}{\partial u(t)} f(x(t), u(t))|_{(x_T(t),0)}$ with approximate state and control

$$\begin{gathered}\tilde{x}(t) = x_T(t) + z(t), \text{ and } \tilde{u}(t) = v(t),\\ \text{where } \dot{z} = A(t)z(t) + B(t)v(t), \text{ s.t. } z(0) = 0.\end{gathered} \tag{4}$$

Using the linearization, steering and distance computations are LQT problems. The next section address the LQT problem and formulates an efficient approach for many executions for the same vertex in a planning graph that is numerically tractable.

Through the linearization, the approximate solution to the non-linear optimal control problem reduces to matrix manipulation.

3 Affine LTV State Exploration

When a system is affine linear it is possible to find a subset of states in $\mathbb{R}^n$ that are reachable from any initial state for a bounded control effort.

Suppose the system with state and control $(\tilde{x}, \tilde{u})$ evolves according to

$$\tilde{x}(t) := \tilde{x}_T(t) + z(t) \text{ and } \tilde{u}(t) := v(t)$$

where $\tilde{x}_T$ is some time-varying translation from the origin and z is the solution to the linear differential equation

$$\dot{z}(t) = A(t)z(t) + B(t)v(t), \text{ s.t. } z(0) = 0 \tag{5}$$

where $A(t) \in \mathbb{R}^{n\times n}$ and $B(t) \in \mathbb{R}^{n\times m}$ are piecewise continuous. Through the state-transition matrix (STM) $\Phi(t, \tau)$ of $A(t)$, the solution to the linear differential equation is

$$z(t) = \int_0^t \Phi(t, \tau)B(\tau)v(\tau)d\tau. \tag{6}$$

The STM is the solution to the matrix differential equation

$$\frac{\partial}{\partial\tau}\Phi(t, \tau) = -\Phi(t, \tau)A(\tau), \Phi(t, t) = Id_{n\times n} \tag{7}$$

where $Id_{n\times n}$ is the $n \times n$ identity matrix. We wish to find the states $\tilde{x}' \in \mathbb{R}^n$ such that there exists a control v constrained to some set $\mathscr{V} \subset \mathscr{U}$ which transfers $\tilde{x}(0) = \tilde{x}_T(0)$ to $\tilde{x}'$ in t_h time. Define the set reachable at t_h with control constrained to $\mathscr{V}$ as

$$\tilde{X}_{\mathscr{V}} := \left\{\tilde{x}' \in \mathbb{R}^n | \exists v \in \mathscr{V} \text{ where } \tilde{x}' = \tilde{x}_T(t) + \int_0^{t_h} \Phi(t, \tau)B(\tau)v(\tau)d\tau\right\}.$$

Consider the control constrained by the control energy

$$\|v\|_R := \frac{1}{2}\int_0^{t_h} v(\tau)^T R(\tau)v(\tau)\, d\tau \tag{8}$$

where $R(\cdot)$ is a piecewise continuous matrix with $R(\tau) = R(\tau)^T > 0$ symmetric positive-definite. The control set with bounded control energy is

$$\mathscr{V}_{\delta v} := \left\{ v \in \mathscr{U} \,|\, \frac{1}{2}\int_0^{t_h} v(\tau)^T R(\tau) v(\tau)\, d\tau < \delta v \right\} \tag{9}$$

where $\delta v > 0$ and $t_h > 0$.

3.1 Open-Loop Exact Linear Shooting

Suppose v has the form

$$v(t) = -R^{-1}(t)B(t)^T \Phi(t_h, t)^T \eta \tag{10}$$

where $\eta \in \mathbb{R}^n$. The significance of this control form is that it is the *minimal-energy control* with control energy Eq. 8 (see [7] Theorem 11.4). This control is open-loop because of its independence on z.

By setting η, the control Eq. 10 transfers $\tilde{x}(0)$ to some $\tilde{x}' \in \mathbb{R}^n$ through Eq. 5. This procedure is shooting. The δv bounded control reachable set $\tilde{X}_{\mathscr{V}_{\delta v}}$ are the points $\tilde{x}' \in \mathbb{R}^n$ for which there exists an η for which both $v \in \mathscr{V}_{\delta v}$ and v transfers $\tilde{x}(0)$ to $\tilde{x}'$. This set $\tilde{X}_{\mathscr{V}_{\delta v}}$ is given in the following Lemma:

Lemma 1 *Supposing $A(\cdot)$ and $B(\cdot)$ are so that Eq. 5 is controllable on $[0, t_h]$ and that v has form Eq. 10, then*

$$\tilde{X}_{\mathscr{V}_{\delta v}} = \left\{ \tilde{x}' \in \mathbb{R}^n | \tfrac{1}{2}(\tilde{x}' - \tilde{x}_T(t_h))^T W_0(t_h)^{-1}(\tilde{x}' - \tilde{x}_T(t_h)) < \delta v \right\} \tag{11}$$

where the matrix $W_0(t)$, $t \in (0, t_h]$ is symmetric positive-definite and W_0 is the solution to:

$$\dot{W}_0(t) = A(t)W_0(t) + W_0(t)A(t)^T + B(t)R(t)^{-1}B(t)^T \; s.t. \; W_0(0) = 0.$$

Proof The symmetric positive-definiteness of $W_0(\cdot)$ follows from $(A(\cdot), B(\cdot))$ controllable and $R(\cdot)$ symmetric positive-definite. Its existence is guaranteed since it is the solution to a linear differential equation. The integral form of $W_0(t)$ is:

$$W_0(t) = \int_0^t \Phi(t,\tau)B(\tau)R(\tau)^{-1}B(\tau)^T\Phi(t,\tau)^T d\tau.$$

An η is allowable if $v \in \mathscr{V}_{\delta v}$ (see Eq. 9). That is, η is allowable if

$$\begin{aligned}\frac{1}{2}\int_0^{t_h} v^T(\tau)R(\tau)v(\tau)d\tau &= \frac{1}{2}\eta^T \int_0^{t_h} \Phi(t_h,\tau)B(\tau)R^{-1}(\tau)B(t)^T\Phi(t_h,\tau)^T d\tau\eta \\ &= \tfrac{1}{2}\eta^T W_0(t_h)\eta < \delta v.\end{aligned} \tag{12}$$

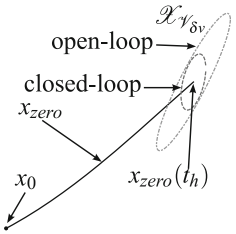


Fig. 2 Reachable set for open-loop and closed-loop double integrator for $x_0 = [0.5, 1]^T$, $\delta_v = 0.2$, $t_h = 1.0$

Through the form of v, the solution to z for some $\eta \in \mathbb{R}^n$ is

$$z(t) = -\int_0^t \Phi(t,\tau)B(\tau)R(\tau)^{-1}B(\tau)^T\Phi(t_h,\tau)^T d\tau\eta = -W_0(t)\Phi(t_h,t)^T\eta.$$

Since $W_0(t_h)$ is positive-definite, it is invertible and the η which transfers $z(0) = 0$ to a desired z' is $\eta = -W_0(t_h)^{-1}z'$. Plugging η in Eq. 12 we arrive at the set of reachable z'. The reachable $\tilde{x}'$ are $\tilde{x}' = \tilde{x}'_T(t_h) + z'$ which is the set $\tilde{X}_{\mathscr{V}_{\delta v}}$. □

The set of reachable states $\tilde{X}_{\mathscr{V}_{\delta v}}$ is an ellipsoid centered around $\tilde{x}_T(t_h)$, as seen in Fig. 2 for the double integrator system. A planning algorithm can leverage Lemma 1 to choose a reachable state and compute the state and control trajectories $(\tilde{x}, \tilde{u})$ that transfer the linear system to that desired state. This exact steering procedure for affine linear systems is shown in Algorithm 1. The procedure relies on a well conditioned $W_0(\cdot)$, especially at t_h. The time-varying matrix W_0 is the reachability Gramian weighted by R^{-1}. It is guaranteed to be invertible when the system $(A(\cdot), B(\cdot))$ is controllable, but numerical inversion requires a well conditioned matrix, which is not the case for the n-link inverted pendulum on a cart analyzed in the examples section. The condition number for time $t_h = 1.0$, $\kappa(W_0(1.0))$, for 1, 2, and 3-link pendulum on the cart at the unstable equilibrium is:

$$\begin{aligned} 1-link &: \kappa(W_0(1.0)) = 1.32 \times 10^6 \\ 2-link &: \kappa(W_0(1.0)) = 1.08 \times 10^8 \\ 3-link &: \kappa(W_0(1.0)) = 3.33 \times 10^{15}. \end{aligned} \tag{13}$$

Algorithm 1 (Open-Loop Exact Linear Steering)

```
For $\tilde{x}' \in \tilde{X}_{\mathscr{V}_{\delta v}}$:
    1. $\eta \leftarrow -W_0(t_h)^{-1}(\tilde{x}' - \tilde{x}_T(t_h))$
    2. $\tilde{x}(t) \leftarrow \tilde{x}_T(t) - W_0(t)\Phi(t_h,t)^T\eta$, and
    3. $\tilde{u}(t) \leftarrow -R(t)^{-1}B(t)^T\Phi(t_h,t)^T\eta$
```

For the 3-link pendulum, the condition number approaches machine precision on many computing devices. When the conditioning is sufficiently poor, the numerical error between $\tilde{x}'$ and the computed $\tilde{x}(t)$ through Algorithm 1 invalidates the procedure. For unstable systems, the conditioning can be improved through a closed-loop control form with stabilizing feedback. The goal is to design a feedback so that Algorithm 1 becomes a sort of stable shooting algorithm.

3.2 Closed-Loop Exact Linear Steering

In order to make Algorithm 1 numerically tractable, we consider a closed-loop system with a better conditioned reachability Gramian than the open-loop system. The closed-loop linear system is (A_K, B) where $A_K(t) := A(t) - B(t)K(t)$ with time-varying feedback gain $K(t) \in \mathbb{R}^{m\times n}$. For now, we assume K is given. It can be computed as the optimal feedback gain of a LQR problem. With a properly designed K, the closed-loop system (A_K, B) should have better numerical properties—e.g. better conditioning of relevant matrices—than the open-loop system.

Let $\Phi_K(\cdot,\cdot)$ be the state-transition matrix corresponding to A_K. The closed-loop control form is

$$v(t) = -K(t)z(t) - R^{-1}(t)B(t)^T\Phi_K(t_h, t)^T\eta. \tag{14}$$

The closed-loop linear system dynamics are

$$\dot{z}(t) = A_K(t)z(t) - B(t)R(t)^{-1}B(t)^T\Phi_K(t_h, t)^T\eta, \text{ s.t. } z(0) = 0$$

with solution

$$\begin{aligned} z(t) &= -\int_0^t \Phi_K(t,\tau)B(\tau)R(\tau)^{-1}B(\tau)^T\Phi_K(t_h,\tau)^T d\tau\eta \\ &= -W_K(t)\Phi_K(t_h,t)^T\eta \end{aligned}$$

where W_K is the reachability Gramian corresponding to A_K, weighted by R^{-1}, and has differential form:

$$\dot{W}_K(t) = A_K(t)W_K(t) + W_K(t)A_K(t)^T + B(t)R(t)^{-1}B(t)^T \text{ s.t. } W_K(0) = 0. \tag{15}$$

As with Lemma 1, we wish to find the reachable subset of $\mathbb{R}^n$ in t_h time with bounded control energy, except for the closed-loop form:

Lemma 2 *Supposing $A(\cdot)$ and $B(\cdot)$ are so that Eq. 5 is controllable on $[0, t_h]$ and that v has form Eq. 14, then*

$$\tilde{X}_{\mathcal{V}_{\delta v}} = \{\tilde{x}' \in \mathbb{R}^n | (\tilde{x}' - \tilde{x}_T(t_h))^T S_K(t_h)^{-1}(\tilde{x}' - \tilde{x}_T(t_h)) < \delta v\} \tag{16}$$

where the matrix $S_K(t)$, $t \in (0, t_h]$ is symmetric positive-definite and is the solution to: (omitting $(\cdot)$)

$$\dot{S}_K = A_K S_K + S_K A_K^T + (K W_K - R^{-1} B^T)^T R (K W_K - R^{-1} B^T) \; s.t. \; S_K(0) = 0. \tag{17}$$

The proof of Lemma 2 is similar to that of Lemma 1 and for brevity is omitted. Similar to Lemma 1, Lemma 2 finds that the reachable states with the closed-loop control form an ellipsoid centered at $\tilde{x}_T(t_h)$, except where the ellipsoid's axes are parameterized by $S_K(t_h)^{-1}$ as opposed to $W_0(t_h)^{-1}$. The analog to Algorithm 1 is

Algorithm 2 (Closed-Loop Exact Linear Steering)
For $\tilde{x}' \in \tilde{X}_{\mathcal{V}_{\delta v}}$ and feedback gain K:

```
1. $\eta \leftarrow -W_K(t_h)^{-1}(\tilde{x}' - \tilde{x}_T(tf))$
2. $\tilde{x}(t) \leftarrow \tilde{x}_T(t) - W_K(t)\Phi_K(t_h, t)^T \eta$, and
3. $\tilde{u}(t) \leftarrow -K(t)z(t) - R(t)^{-1}B(t)^T \Phi(t_h, t)^T \eta$
```

The conditioning of W_K and S_K for the n-link cart pendulum is a significant improvement over W_0 (see Eq. 13):

$$\begin{aligned}
1 - link &: \kappa(W_K(1.0)) = 5.84 & \kappa(S_K(1.0)) = 3.80 \times 10^4\\
2 - link &: \kappa(W_K(1.0)) = 5.49 \times 10^2 & \kappa(S_K(1.0)) = 2.52 \times 10^6\\
3 - link &: \kappa(W_K(1.0)) = 1.03 \times 10^5 & \kappa(S_K(1.0)) = 2.57 \times 10^6.
\end{aligned}$$

The closed loop form should be used when the open loop reachability Gramian is poorly conditioned.

3.3 Inexact Linear Steering

The reachability results provide the minimal energy control to a reachable set. When the desired state is not within the reachable set a new objective can be considered which weights the importance of tracking the desired state. Consider the cost

$$J(\tilde{x}, \tilde{u}; t_h) := \frac{1}{2}\int_0^{t_h} \tilde{u}^T(\tau) R(\tau) \tilde{u}(\tau)\, d\tau + \frac{1}{2}(\tilde{x}(t_h) - \tilde{x}_{des})^T P_1 (\tilde{x}(t_h) - \tilde{x}_{des}) \tag{18}$$

where $P_1 = P_1 \geq 0$ is symmetric positive semi-definite. Parameterized by (z, v),

$$\begin{aligned}
J(z, v; t_h) = &\frac{1}{2}\int_0^{t_h} v^T(\tau) R(\tau) v(\tau)\, d\tau\\
&+ \tfrac{1}{2}(\tilde{x}_T(t_h) + z(t_h) - \tilde{x}_{des})^T P_1 (\tilde{x}_T(t_h) + z(t_h) - \tilde{x}_{des}).
\end{aligned} \tag{19}$$

The problem is to find a state and control trajectory that satisfies the linear dynamics Eq. 5. That is, we wish to solve the problem

Problem 1 Solve

$$\begin{aligned} &\min_{z,v;t_h} J(z, v; t_h) \\ \text{s.t. } &\dot{z}(t) = A(t)z(t) + B(t)v(t), z(0) = 0. \end{aligned}$$

For a fixed t_h, this problem is a linear quadratic tracking LQT problem [1], the solution of which is well known. The optimal control is

$$v^{\star}(t) = -K^{\star}(t)z^{\star}(t) - R(t)^{-1}B(t)^T \Phi_K(t, t_h)P_1(\tilde{x}_T(t_h) - \tilde{x}_{des}) \tag{20}$$

with optimal feedback gain $K^{\star}(t)$

$$K^{\star}(t) = R(t)^{-1}B(t)^T P(t) \tag{21}$$

where $P(t)$ is the solution to the Riccati equation

$$-\dot{P}(t) = A(t)^T P(t) + P(t)A(t) - P(t)B(t)R(t)^{-1}B(t)^T P(t) \text{ s.t. } P(t_h) = P_1. \tag{22}$$

Notice that $v^{\star}$ has the closed-loop form Eq. 14 except for a specific feedback gain $K(t) = K^{\star}(t)$ and where $\eta = P_1(\tilde{x}_T(t_h) - \tilde{x}_{des})$. The gain K in Eq. 14 can be chosen as the optimal feedback gain from this LQT problem. The $K(t)$ would depend on the choice of $\tilde{x}_{des}$ as well as P_1. If $P_1 = 0$—i.e. the cost function reduces to just the control energy—then $K^{\star} \equiv 0$—i.e. there is no feedback—and the optimal control is the open-loop control Eq. 10. As such, the procedure Algorithm 1 does indeed compute the minimal control energy trajectory that transfers the system to a reachable state.

The following algorithm computes the optimal cost $\tilde{J}^{\star}(t_h) := J(\tilde{x}^{\star}, \tilde{u}^{\star}, t_h)$, and trajectory $(\tilde{x}^{\star}, \tilde{u}^{\star})$ for fixed t_h:

Algorithm 3 (Fixed t_h Inexact Linear Steering)
For $\tilde{x}_{des} \in X$ and $t_h > 0$:

```
1. $P(t) \leftarrow$ solve Riccati equation Eq. 22
2. $K^{\star}(t) \leftarrow R(t)^{-1}B(t)^T P(t)$
3. $z^{\star}(t) \leftarrow$ solve $\dot{z}^{\star}(t) = A_{K^{\star}}(t) - B(t)R(t)^{-1}B(t)^T \Phi_K(t, t_h)P_1(\tilde{x}_T(t_h) - \tilde{x}_{des})$
4. $\tilde{x}^{\star}(t) \leftarrow \tilde{x}_T(t) + z^{\star}(t)$, and
5. $\tilde{u}^{\star}(t) \leftarrow -K^{\star}(t)z^{\star}(t) - R(t)^{-1}B(t)^T \Phi_K(t, t_h)P_1(\tilde{x}_T(t_h) - \tilde{x}_{des})$
6. $\tilde{J}^{\star}(t_h) \leftarrow$ solve cost function Eq. 19
7. Return $\tilde{J}^{\star}(t_h)$ and $(\tilde{x}^{\star}, \tilde{u}^{\star})$
```

In comparison to Algorithms 1 and 2, this algorithm computes a trajectory that tracks any state in X as opposed to just the states in a reachable set. However, the trajectory does not transfer the state to $\tilde{x}_{des}$ and as such is an inexact steering.

3.4 Efficient Inexact Linear Steering

For an affine linear system—i.e. for a set x_T, A and B—Algorithm 3 must be executed anew for distinct t_h and $\tilde{x}_{des}$. As we show here, inexact linear steering gains efficiency through precomputation and caching to remove redundancies from multiple algorithm execution.

As previously noted, the closed-loop control Eq. 14 has the same form as the solution to the LQT problem with optimal control Eq. 20. The efficiency is gained through precomputing and fixing a feedback gain $K(t)$ and solving for η as was done in Sect. 3.2. In other words, the problem is minimize the cost Eq. 19 constrained to the closed-loop control Eq. 14:

Problem 2 With fixed $K(t)$, solve

$$\begin{aligned} &\min\nolimits_{\eta;t_h} J(z,v,t_h) \\ \text{s.t. } &\dot{z}(t) = A_K(t)z(t) - B(t)v(t)R^{-1}(t)B(t)^T\Phi_K(t_h,t)^T\eta,\ z(0)=0 \\ &v(t) = -K(t)z(t) - R^{-1}(t)B(t)^T\Phi_K(t_h,t)^T\eta. \end{aligned}$$

The solution to Problem 2 is not the same as the solution to Problem 1 unless the feedback gain $K(t)$ happens to be the optimal $K^\star(t)$.

The solution to Problem 2 is given in the following Lemma:

Lemma 3 *For fixed $t_h > 0$ and K and supposing $A(\cdot)$ and $B(\cdot)$ are so that Eq. 5 is controllable on $[0,t_h]$, the solution to Problem 2 is*

$$\eta^\star = P_{t_h}(\tilde{x}_T(t_h) - \tilde{x}_{des}) \tag{23}$$

where

$$P_{t_h} = (W_K(t_h)P_1W_K(t_h) + S_K(t_h))^{-1}W_K(t_h)P_1.$$

Additionally,

$$\begin{aligned} z^\star(t) &= -W_K(t)\Phi_K(t_h,t)^T\eta^\star, \\ v^\star(t) &= [K(t)W_K(t) - R(t)^{-1}B(t)^T]\Phi_K(t_h,t)^T\eta^\star \end{aligned} \tag{24}$$

and

$$\begin{aligned} &J(z^\star,v^\star;t_h) = \tfrac{1}{2}(\eta^\star)^TS_K(t_h)\eta^\star \\ &+\tfrac{1}{2}(\tilde{x}_T(t_h) - W_K(t_h)\eta^\star - \tilde{x}_{des})^TP_1(\tilde{x}_T(t_h) - W_K(t_h)\eta^\star - \tilde{x}_{des}). \end{aligned} \tag{25}$$

Proof The integral form of z is Eq. 6:

$$\begin{aligned} z(t) &= \textstyle\int_0^t \Phi_K(t,\tau)B(\tau)v(\tau)d\tau = -\int_0^t \Phi_K(t,\tau)B(\tau)R(\tau)^{-1}B(\tau)^T\Phi(t_h,\tau)^Td\tau\eta \\ &= -W_K(t)\Phi_K(t_h,t)^T\eta. \end{aligned}$$

Plugging $z(t)$ into $v(t)$,

$$v(t) = (K(t)W_K(t) - R(t)^{-1}B(t)^T)\Phi_K(t_h, t)^T\eta.$$

As such, $z(t)$ and $v(t)$ depend linearly on η. The control energy is

$$\begin{aligned}&\tfrac{1}{2}\int_0^{t_h} v^T(\tau)R(\tau)v(\tau)\,d\tau\\ &\quad= \tfrac{1}{2}\eta^T \int_0^{t_h} \Phi_K(t_h,\tau)(K(\tau)W_K(\tau) - R(\tau)^{-1}B(\tau)^T)^T\\ &\qquad\cdot R(\tau)(K(\tau)W_K(\tau) - R(\tau)^{-1}B(\tau)^T)\Phi_K(t_h,\tau)d\tau\eta\\ &\quad= \tfrac{1}{2}\eta^T S_K(t_h)\eta.\end{aligned}$$

The cost is thus

$$J = \tfrac{1}{2}(\eta)^T S_K(t_h)\eta + \tfrac{1}{2}(\tilde{x}_T(t_h) - W_K(t_h)\eta - \tilde{x}_{des})^T P_1(\tilde{x}_T(t_h) - W_K(t_h)\eta - \tilde{x}_{des})$$

with optimal $\eta^\star$ where $\frac{\partial}{\partial\eta}J|_{\eta\to\eta^\star} = 0$:

$$\frac{\partial}{\partial\eta}J|_{\eta\to\eta^\star} = [(S_K(t_h) + W_K(t_h)P_1W_K(t_h))\eta - W_K(t_h)P_1(\tilde{x}_T(t_h) - \tilde{x}_{des})]_{\eta\to\eta^\star} = 0.$$

Since $S_K(t_h)$ and $W_K(t_h)$ are positive definite through the controllability assumption, $(S_K(t_h) + W_K(t_h)P_1W_K(t_h))$ is invertible. Solving for $\eta^\star$ results in Eq. 23. □

The lemma instructs how to do efficient fixed t_h inexact steering:

Algorithm 4 (Efficient Fixed t_h Inexact Linear Steering)
For $\tilde{x}_{des} \in X$ and $t_h > 0$:

```
$P_{t_h} \leftarrow (W_K(t_h)P_1W_K(t_h) + S_K(t_h))^{-1}W_K(t_h)P_1$
$\eta^\star \leftarrow P_{t_h}(\tilde{x}_T(t_h) - \tilde{x}_{des})$
$\tilde{x}^\star(t) \leftarrow \tilde{x}_T(t) - W_K(t)\Phi_K(t_h, t)^T\eta^\star$
$\tilde{u}^\star(t) \leftarrow [K(t)W_K(t) - R(t)^{-1}B(t)^T]\Phi_K(t_h, t)^T\eta^\star$
$\tilde{J}^\star(t_h) \leftarrow$ solve Eq. 25
Return $\tilde{J}^\star(t_h)$ and $(\tilde{x}^\star, \tilde{u}^\star)$
```

The algorithm is efficient compared to Algorithm 3 because it does not rely on solving any differential equations assuming certain functions have been precomputed. The functions to be precomputed and saved in memory are $K(t)$, $W_K(t)$, $S_K(t)$ and $\Phi_K(t_h^{max}, t)$ for a specified long time horizon $t \in [0, t_h^{max}]$, $t_h^{max} > 0$. As such, Algorithm 4 relies solely on matrix manipulations to return a fixed t_h inexact linear steering trajectory assuming $t_h < t_h^{max}$. Note that $\Phi_K(t_h, t) = \Phi_K(t_h^{max}, t_h)^{-1}\Phi_K(t_h^{max}, t)$. Also, $\Phi(t_h, t_h) = Id_{n\times n}$ and so $\tilde{x}^\star(t_h)$, $\tilde{u}^\star(t_h)$ and $\tilde{J}^\star(t_h)$ do not rely on precomputing Φ.

As of yet, the feedback gain is assumed to have been chosen in some way. A reasonable choice is the optimal feedback gain to the LQT problem, Problem 1, for the max time horizon $t_h = t_h^{max}$. It is the case that the solution to Problem 2 is equivalent to the solution to Problem 1 when $P_1 = P_{t_h}$. In other words, if $P_1 = P_{t_h}$, then $K = K^\star$, which will be the case at least at t_h^{max} for this choice of K.

4 Extending to Nonlinear Systems

We extend the affine linear steering results in Sect. 3 to nonlinear dynamics $\dot{x}(t) = f(x(t), u(t))$ by projecting inexact and exact linear steering trajectories to feasible trajectories. A planner can pick which linear steering algorithm, Algorithms 1–4, to approximately transfer the system to a desired state x_{des}. The resulting approximate trajectories $(\tilde{x}, \tilde{u})$ are not feasible unless the dynamics are linear.

The trajectory functional projection operator $\mathscr{P}$ proposed in [6] maps $(\tilde{x}, \tilde{u})$ to a feasible trajectory.

$$\begin{bmatrix} x \\ u \end{bmatrix} = \mathscr{P}\left(\begin{bmatrix} \tilde{x} \\ \tilde{u} \end{bmatrix}\right) := \begin{cases} \dot{x} = f(x, u) \\ u = \tilde{u} - K(x - \tilde{x}). \end{cases} \tag{26}$$

The projection is a feedback loop with gain K, reference signal $\tilde{x}$, and feedforward term $\tilde{u}$. The feedback gain may be chosen as the optimal feedback of an LQR or LQT problem.

Inexact steering transfers a vertex $x_0 \in V$ to a state near a desired state x_{des}. Suppose $K(t)$, $W_K(t)$, $S_K(t)$ and $\Phi_K(t_h^{max}, t)$ have been computed for the vertex x_0. Then, the efficient inexact steering trajectory is computed by projecting the approximate trajectories given by efficient inexact linear steering, Algorithm 4.

```
Algorithm 5 (Efficient Inexact Steering)
For $x_0 \in X$, $x_{des} \in X$, and $t_h \in (0, t_h^{max}]$:

1. $(\tilde{x}^\star, \tilde{u}^\star, \tilde{J}^\star(t_h)) \leftarrow$ Algorithm 4 for $t_h$
2. $(x, u) \leftarrow \mathscr{P}(\tilde{x}^\star, \tilde{u}^\star)$
3. return $(x, u, \tilde{x}, \tilde{u}, \tilde{J}^\star(t_h), t_h)$
```

5 Examples and Algorithm

Using the steering and nearest neighbor methods proposed in the paper we construct an RRT to plan a trajectory that transfers a 3-link pendulum on a cart through a corridor of obstacles to a goal region. The state is composed of the pendulum angles θ_i, their angular velocities $\dot{\theta}_i$, the cart position p and the cart velocity $\dot{p}$. The control is the force applied to the cart, accelerating it forward or backward, constrained by $u \in [-20, 20]$ Newtons. The pendulums are in inverted equilibrium for the start state, with cart position at $p_{start} = 0\,\text{m}$ as seen in Fig. 1. The obstacles are circles in the workspace. The 3-link pendulum has 8 states. Each pendulum head has mass 0.1 kg and the pendulum lengths are assumed massless. All pendulums links have length $1/3\,\text{m}$ for a total length of 1 m. The cart has mass 1 kg. The matrices R and P_1 in the cost J, Eq. 18, are $R = [0.025]$ and $P_1 = Id_n$ the $n \times n$ identity matrix. We execute an RRT-like algorithm with efficient inexact steering.

5.1 Algorithm

An RRT formulation with efficient inexact steering through precomputing follows. At each iteration, a sample state $x \in X$ and a sample time $t_{h,samp} \in (0, t_h^{max}]$ over a uniform distribution is taken. The vertex that is the nearest neighbor is computed and the system is steered using the efficient inexact steering algorithm. The resulting approximate trajectory is projected to the set of feasible trajectories. An insertion failure occurs if the projected trajectory collides with a boundary of X such as an obstacle. Otherwise, the precomputations for the new vertex are made and the new vertex is appended to the graph.

Algorithm 6 (RRT with Efficient Inexact Steering) *for* $x_{start} \in X$:

```
Precompute $x_{zero}, K, W_K, S_K, \Phi_K$ for $x_{start}$ over $t \in [0, t_h^{max}]$. (Eqs. 3, 21, 15, 17, 7)
$N_{init} = \{x_{start}, x_{zero}, K, W_K, S_K, \Phi_K\}$
$V \leftarrow \{N_{init}\}$; $\mathscr{E} \leftarrow \emptyset$
while $x_{goal}$ not found:
  $x_{samp} \leftarrow sample(x)$, $t_{h,samp} \leftarrow sample((0, t_h^{max}])$
  $N_{near} \leftarrow nearestneighbor(V, x_{samp})$.
  $(x_{new}, u_{new}, \tilde{x}_{new}, \tilde{u}_{new}, \tilde{J}^\star, t_{h,samp}) \leftarrow$ Algorithm 5 from $N_{near}$
  $(x, u) \leftarrow \mathscr{P}(\tilde{x}, \tilde{u})$ (Eq. 26)
  if $(x_{new}(t), u_{new}(t)) \in (X, U)$ for all $t \in [0, t_{h,samp}]$:
    Precompute $x_{zero}, K, W_K, S_K, \Phi_K$ for $x_{new}(t_{h,samp})$
          over $t \in [0, t_h^{max}]$.
    $N_{new} \leftarrow \{x_{new}(t_{h,samp}), x_{zero}, K, W_K, S_K, \Phi_K\}$
    $V \leftarrow V \cup \{N_{new}\}$; $\mathscr{E} \leftarrow \mathscr{E} \cup \{(x_{new}, u_{new}; t_{h,samp})\}$
return $G = (V, \mathscr{E})$
```

5.2 Implementation

Source code implementing Algorithm 6 for the 3-link pendulum on a cart in C++ can be obtained at https://github.com/timocaldwell/agile_rrt/. See the README for implementation details and for animations of solution trajectories of the 3-link pendulum on a cart as it traverses past obstacles. All results were computed on the University of Colorado's JANUS Supercomputer. A single run of Algorithm 6 utilized a single 2.8Ghz Intel Westmere processor.

5.3 Results

We compare planning results past two obstacles for multiple max time horizons $\bar{t}_h = 0.05, 0.1, 0.175, 0.25, 0.5$ s and linearizing about $x_T = x_0$ or $x_T(t) = x_{zero}(t)$. The obstacles are circles, one placed at $(3, -0.85)$ and the other at $(3, 0.85)$, both with radius 0.6. The setup and results of applying Algorithm 6 for differing max time

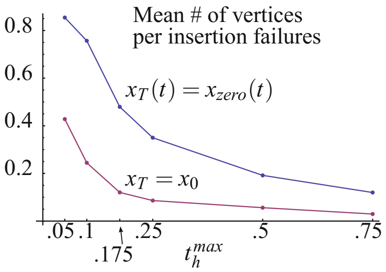


Fig. 3 The mean number of tree vertices per number of insertion failures for 100 runs of RRT at differing max time horizons for the linearization candidates

horizons are shown in Fig. 1. Qualitatively, the planner explores significantly more space with greater time horizons for the same number of tree vertices.

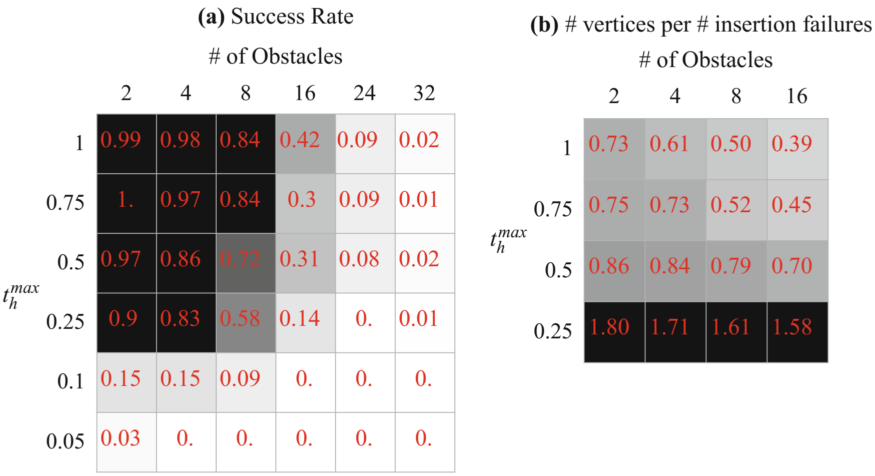
For each combination of max time horizon and linearization candidate, we execute Algorithm 6 100 times. We stop execution once a state $x' \in X$ is discovered within a goal region defined by $\|x' - x_{center}\| < \delta = 2$, where x_{center} is the equilibrium point with $p_{center} = 6\,\mathrm{m}$ and each pendulum in the inverted configuration. To compare the candidate linearizations, the mean number of vertices in the RRT per number of failed insertions is reported in Fig. 3. Due to the higher quality approximation through linearizing about x_{zero}, the linearization candidate x_{zero} results in about 4 times fewer insertion failures than linearizing about x_0.

We execute Algorithm 6 for randomly selected obstacles to illuminate how differing max time horizons perform under differing obstacle densities. Each obstacle's center is set to be within $[2.5, 5.5] \times [-1, 1]$ and the obstacles radius is within $[0, 0.1]$. The goal region is any state $x \in \mathbb{R}^n$ where the top pendulum head is located to the right of the obstacles—i.e. if for x, (X_{pend3}, Y_{pend3}) is the location of the 3rd pendulum head, then x is within the goal region when $X_{pend3} > 6\,\mathrm{m}$. This way, all obstacles are within a region designated between the start state and the goal region.

We ran the experiment 100 times for each combination of max time horizons of 0.05, 0.1, 0.25, 0.5, 0.75, 1.0s and obstacle densities of 2, 4, 8, 16, 24, and 32 obstacles. A success was attributed to runs that found the goal region within 10000 vertices. The success rate is shown in Fig. 4a. The longer time horizons were more successful. For instance, there were about 3 times more successful plans for $t_h^{max} = 1.0\,\mathrm{s}$ than $t_h^{max} = 0.25\,\mathrm{s}$ at 16 obstacles. The shortest time horizons had difficulty finding the goal region within the allotted number of vertices.

As one may expect, trajectories of longer time horizon are more likely to collide with an obstacle resulting in a failed insertion. In Fig. 4b, we show the number of vertices per number of insertion failures and find that at 16 obstacles there are about 4 times as many insertion failures per vertex for $t_h^{max} = 1.0\,\mathrm{s}$ than $t_h^{max} = 0.25\,\mathrm{s}$. However, the ratio is less for 2 obstacles where it is a little more than 2 times.

Fig. 5 shows example successful plans for 24 randomly placed obstacles.



(a) Success Rate

t_h^{max} \ # of Obstacles	2	4	8	16	24	32
1	0.99	0.98	0.84	0.42	0.09	0.02
0.75	1.	0.97	0.84	0.3	0.09	0.01
0.5	0.97	0.86	0.72	0.31	0.08	0.02
0.25	0.9	0.83	0.58	0.14	0.	0.01
0.1	0.15	0.15	0.09	0.	0.	0.
0.05	0.03	0.	0.	0.	0.	0.

(b) # vertices per # insertion failures

t_h^{max} \ # of Obstacles	2	4	8	16
1	0.73	0.61	0.50	0.39
0.75	0.75	0.73	0.52	0.45
0.5	0.86	0.84	0.79	0.70
0.25	1.80	1.71	1.61	1.58

Fig. 4 Results for planning through a differing number of randomly placed obstacles for a max of 10,000 RRT vertices. The number of obstacles are given. **a** Success rate, where a success is awarded if the planner finds a path to the goal region past the obstacles. **b** Number of vertices used to find a successful path divided by the number of failed insertions

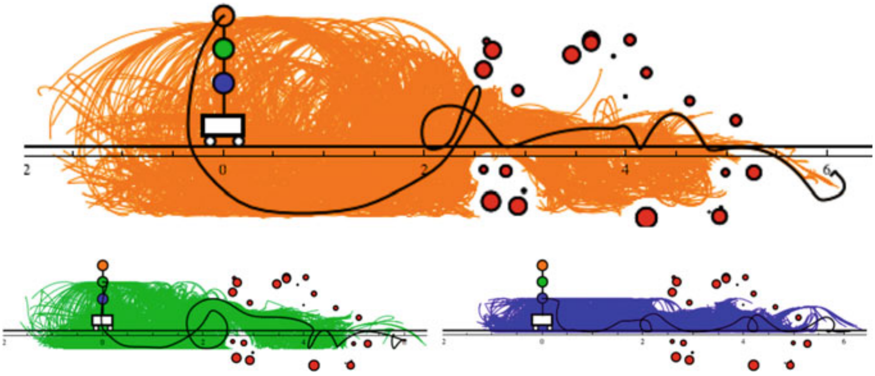


Fig. 5 An example successful plan for 24 random obstacles. Depicts the explored paths for each pendulum head with the path that takes the top head the furthest in *black*

5.4 Discussion

The results compare the capability of an RRT-based planner to search the allowable state space of a dynamic system for differing max time horizons, differing linearizations, and differing obstacle densities. Qualitatively, Fig. 1 provides evidence that planning with longer time horizons will search a space using far fewer vertices. Quantitative evidence is given in Fig. 4a where the success rate is greater for longer time horizons when the number of vertices in the tree is used as a stopping criteria. However, as seen in Fig. 4b, there are more failed insertions for denser obstacle sets. We conclude that the max time horizon should be variable on the local obstacle density, which is a topic of future work.

Additionally, when longer time horizons are needed, Fig. 3 shows that linearizing about the zero-control trajectory results in far fewer insertion failures and should be used for the 3-link pendulum on a cart. For systems with even stronger nonlinearities, like hybrid systems, the higher quality approximation may be even more necessary, and is also a future direction.

6 Conclusion

Complementing [2, 3, 15, 17], we approximate the optimal control problems that arise in sample-based planning through a linearization. However, we linearize about the zero-control trajectory, making the approximation valid for longer time horizons. Moreover, we steer a closed-loop system instead of the numerically less tractable open-loop system. Each of these decisions were made so that steering is viable over long time horizons. With longer time horizons, a planner's exploration can be coarser, thereby requiring fewer nodes in a graph, thereby decreasing a sample-based planner's computation time.

Acknowledgements This work has been supported by NASA Early Career Faculty fellowship NNX12AQ47GS02 and ARO grant W911NF1410203. This work utilized the Janus supercomputer, which is supported by the National Science Foundation (award number CNS-0821794) and the University of Colorado Boulder. The Janus supercomputer is a joint effort of the University of Colorado Boulder, the University of Colorado Denver and the National Center for Atmospheric Research. Janus is operated by the University of Colorado Boulder

References

1. Anderson, B.D.O., Moore, J.B.: Optimal Control: Linear Quadratic Methods. Dover Publications INC (1990)
2. Glassman, E., Tedrake, R.: A quadratic regulator-based heuristic for rapidly exploring state space. In: IEEE International Conference on Robotics and Automation, pp. 5021–5028 (2010)
3. Goretkin, G., Perez, A., Platt Jr. R., Konidaris, G.: Optimal sampling-based planning for linear-quadratic kinodynamic systems. In: IEEE International Conference on Robotics and Automation, pp. 2429–2436 (2013)
4. Hauser, J.: A projection operator approach to the optimization of trajectory functionals. In: IFAC World Congress (2002)
5. Hauser, J.: On the computation of optimal state transfers with application to the control of quantum spin systems. In: American Control Conference, pp. 2169 – 2174 (2003)
6. Hauser, J., Meyer, D.G.: The trajectory manifold of a nonlinear control system. In: IEEE Conference on Decision and Control, pp. 1034–1039 (1998)
7. Hespanha, J.P.: Linear Systems Theory. Princeton university press, Princeton (2009)

8. Janson, L., Pavone, M.: Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions. In: International Symposium on Robotics Research (2013)
9. Jeon, J.H., Karaman, H., Frazzoli, E.: Anytime computation of time-optimal off-road vehicle maneuvers using the rrt*. In: IEEE Conference on Decision and Control, pp. 3276–3282 (2011)
10. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. In: *The International Journal of Robotics Research*, pp. 846–894 (2011)
11. Kunz, T., Stilman, M.: Kinodynamic rrts with fixed time step and best-input extension are not probabilistically complete. In: International Workshop on the Algorithmic Foundations of Robotics (2014)
12. Lavalle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: The International Journal of Robotics Research, pp. 378–400 (2001)
13. Li, Y., Littlefield, Z., Bekris, K.E.: Sparse methods for efficient asymptotically optimal kinodynamic planning. In: Workshop on the Algorithmic Foundations of Robotics (2014)
14. Littlefield, Z., Li, Y., Bekris, K.E.: Efficient sampling-based motion planning with asymptotic near-optimality guarantees for systems with dynamics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1779–1785 (2013)
15. Perez, A., Platt, R., Konidaris, G., Kaelbling, L., Lozano-Perez, T.: LQR-RRT*: optimal sampling-based motion planning with automatically derived extension heuristics. In: IEEE International Conference on Robotics and Automation, pp. 2537–2542 (2012)
16. Schmerling, E., Janson, L., Pavone, M.: Optimal sampling-based motion planning under differential constraints: the drift case with linear affine dynamics. In: IEEE Conference on Decision and Control, pp. 2574–2581 (2015)
17. Webb, D.J., van den Berg, J.: Kinodynamic RRT*: asymptotically optimal motion planning for robots with linear dynamics. In: IEEE International Conference on Robotics and Automation, pp. 5054–5061 (2013)

Active Multi-view Object Recognition and Online Feature Selection

Christian Potthast, Andreas Breitenmoser, Fei Sha and Gaurav S. Sukhatme

1 Introduction

Recognizing objects from visual percepts is a challenging task for mobile robots. Conventional approaches for object recognition are tour de force, relying almost exclusively on complex statistical models for classification and heavy engineering of computationally-intensive features [4, 14]. The processing capability of the recognition system, however, is seldom questioned and assumed to be unconstrained. In stark contrast, mobile robots are often underpowered and need to be frugal on resources for computing and reasoning—being able to move is their first priority!

Active object recognition [2, 8] is an appealing paradigm to overcome the challenge. Here, a mobile robot configures and positions its sensor to inspect an object and recognize it as a previously learnt one. Active object recognition enables exploratory actions to account for unforeseen objects or scenes, and offers greater flexibility in actively seeking the most useful information for inference under uncertainty.

As such, adaptive action selection can potentially harvest the statistical modeling power of conventional approaches even under the severe (computing) constraints on the mobile robots. The general idea is for the sensors to be controlled such that only data with high information content is collected, *adapted dynamically* to the robot's

C. Potthast (✉) · A. Breitenmoser · F. Sha · G.S. Sukhatme
Department of Computer Science, University of Southern California (USC),
Los Angeles, CA 90089, USA
e-mail: potthast@usc.edu

A. Breitenmoser
e-mail: andreas.breitenmoser@usc.edu

F. Sha
e-mail: feisha@usc.edu

G.S. Sukhatme
e-mail: gaurav@usc.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_27

knowledge (i.e., the belief state about the identity of the object) as a result of running inference on previously acquired information.

Concretely, two common actions have been explored in previous work: to compute a subset of yet informative features and to control the number of different viewpoints. However, those works do not exploit the full potential of combining the two strategies in concert.

The main idea of our work hinges on the following intuition: *how to reduce overall costs for recognition by relying on very simple features extracted at multiple views in lieu of complex features extracted at a single view?* Specifically, in this paper, we describe an information-theoretic framework that unifies these two strategies: view planning for moving robots to the optimal viewpoints and dynamic feature selection for extracting simple yet informative features. However, unlike standard feature selection, the two strategies are symbiotic in our algorithm. The selected features at any given time depend on the past trajectory of the viewpoints while simultaneously affecting the future viewpoint(s).

We conduct extensive evaluations on a large RGB-D camera dataset of single object scenes. Our results show that dynamic feature selection reduces the computation time by five folds at runtime. Furthermore, when we combine it with viewpoint planning, we increase significantly the recognition accuracy, often as high as 8–15%. We further demonstrate the proposed approach through experiments with a quadcopter robot, equipped with a RGB-D camera that collects multiple views by flying around an object. Our results show pose estimates within ±45° of the ground truth and object recognition accuracies of over 90%, despite using relatively simple features in rather challenging object recognition experiments.

The remainder of the paper is organized as follows. Section 2 discusses related work. We formalize the main idea of active multi-view object recognition in Sect. 3, followed by a detailed description of how to plan optimal views and select features online in Sect. 4. We report empirical evaluations in Sect. 5 and conclude in Sect. 6.

2 Related Work

Works on active perception go back as far as the seminal paper [3]. “Active” can either refer to adjustments of the sensor parameters themselves, e.g., adaptively changing the focal length of a camera, as in [3, 8], or mean that the entire sensor moves together with the underlying platform. The latter is particularly common in robotics, where robots plan new observation locations and viewing directions actively [2, 11, 13]. This is known as robotic view planning [18] and has various applications, such as inspection of underwater structures, object search in indoor environments or detecting objects on a table top [2]. The common goal of these and similar approaches is to recognize objects and estimate their pose [8, 12, 13]. To deal with uncertainty in the robot’s state space, [2, 20] formulated the problem as a POMDP, however, their decision making comes with a higher computational cost. References [6, 7, 16] propose multi-view object recognition frameworks that use cameras and image databases;

alternatively, objects can be recognized by extracting features from laser scanners and RGB-D depth cameras [2, 9, 15]. Reference [15] applies the online boosting approach from [10] to point clouds in the robotic domain for feature selection. Feature selection can support the object recognition task by reducing the computation time and confusions through suboptimal feature sets [21]. A common alternative to boosting is the selection of features by mutual information [17, 21]. Reference [1] evaluates the feature selection methods from [17] when applied to the RGB-D dataset of [12]. Although many approaches for active view planning as well as online feature selection exist, to the best of our knowledge, the advantages of both approaches have never been leveraged in combination. Reference [22] proposes a unifying view on active recognition and feature selection, however, focuses on feature interdependencies and the selection of features given a concrete context, e.g., a specific target object to detect. Their method, as well as the other feature selection methods for object recognition above, do not explicitly consider viewpoint selection, whereas our object recognition framework applies both feature and viewpoint selection in concert.

3 Problem Formulation

Given a static scene and an object in question, we want to recognize the object as well as estimate its pose relative to the coordinate system of the observing sensor. We are interested in choosing a sequence of actions for the sensor which maximizes the recognition accuracy and leads to a minimum operation time and thus minimal energy consumption. In each time step t, we are provided with a set of possible actions a_t and have the choice of (1) stopping the recognition process, (2) staying at the current location and computing a next feature f_t with different feature type (i.e., applying a null action $a_t = \emptyset$), or (3) moving the sensor to a new observation location by taking the action $a_t \neq \emptyset$ and computing a first feature f_t from the new viewpoint. We assume that the type of the first feature evaluated at a new location is pre-defined by the user. Further, we restrict the motion of the sensor to a path that connects fixed discretized viewing positions. This reduces complexity in computation compared to a continuous representation of viewpoints. After an action is taken, we obtain a new sensor measurement. The sensor measurements are parametrized by features from a feature vector $\boldsymbol{f} = [f^1, \ldots, f^N]$ which includes N different feature types in total. The values of the features depend on the identity of an object. We incorporate the uncertainty of the observations by the conditional probability density function $p(f_t|o_t)$ with $o_t = \{c_t, \theta_t\}$. In order to infer about the current object class c_t and orientation angle θ_t of the observed object from a set of measurements, we use a *Bayesian framework*. The parameters are assumed to be obtained from a physical object model and to be trained offline. The general system diagram of our active recognition framework is presented in Fig. 1. It consists of an inner and an outer loop and the four modules: *object observation*, *feature selection*, *recognition state estimation*, and *viewpoint selection*.

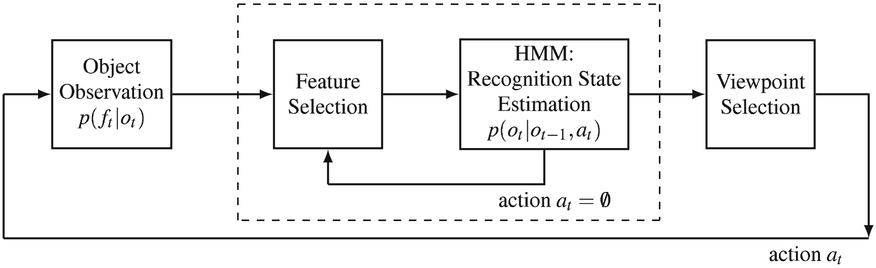

Object Observation $p(f_t|o_t)$
Feature Selection
HMM: Recognition State Estimation $p(o_t|o_{t-1}, a_t)$
Viewpoint Selection
action $a_t = \emptyset$
action a_t

Fig. 1 Overview of the active multi-view object recognition framework. After each observation, the HMM updates the recognition state, upon which a new feature (*inner loop*) or a new viewpoint (*outer loop*) is selected. The feature type that is selected first at any new location is pre-defined

At each observation location, we start off with extracting one first feature. We then have the choice of either extracting another feature (i.e., to locally exploit) or moving to a new location (i.e., to globally explore), depending on our stopping criterion. However, the premise is to extract as much information as possible at the current location since moving to a new location is inherently more costly. Also, we only want to move to a new location if the gain of information (in expectation) significantly reduces the uncertainty of the belief state, or in other words, if the recognition accuracy is expected to improve significantly. Each time we decide to add a new feature, we compute the utility of all remaining features in our arsenal and select the one with the highest utility. We only allow to add one feature at a time, and once a feature is added it cannot be removed from the set anymore. By doing this, integrating a new feature is simply an update step, without the need of re-computing the posterior of the entire feature set. We continue until either we reach our stopping criterion or all N features are observed at a given location, forcing us to move to the next location or to terminate. By this, ideally, at each location we have only extracted useful features and left out features that yield no additional information. Additionally, not extracting a feature saves time, which helps to reduce the total time needed to recognize the object.

Sequential Bayesian Recognition: Given a new feature f_t observed from the object, we want to efficiently integrate the new observation with results from prior observations and update our current belief about the object class and orientation. We formulate the recognition and pose estimation of the object as a Bayesian network in form of a HMM, where the observation sequence is defined as $S = \{o_1, \ldots, o_t, a_2, \ldots, a_t\}$ as shown in Fig. 2. The hidden state of our HMM is defined as recognition state o_t with $o_{1:T} = \{o_1, \ldots, o_t, \ldots, o_T\}$, the observations are given as observed features f_t with $f_{1:T} = \{f_1, \ldots, f_t, \ldots, f_T\}$ and the actions to move between viewpoints are given as a_t with $a_1 = \emptyset$ and $a_{2:T} = \{a_2, \ldots, a_t, \ldots, a_T\}$. The state of the hidden process satisfies the Markov property, that is, given the value of o_{t-1}, the current state o_t is independent of all the states prior to $t - 1$. We can define the joint distribution of the HMM as follows:

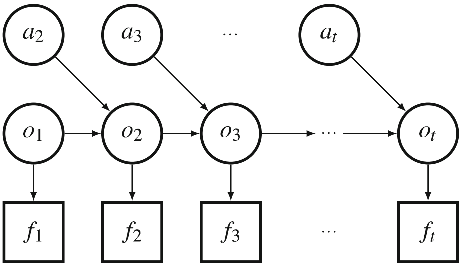


Fig. 2 HMM for sequential Bayesian recognition. Given a sequence of observed features $f_{1:t}$ and executed actions $a_{2:t}$, we reason about the hidden recognition state of an object at time t, $o_t = \{c_t, \theta_t\}$, including its object class $c_t = c^k$ and object pose $\theta_t = \theta^q$. (Null action a_1 is not displayed.)

$$\begin{aligned} P(o_{1:T}, f_{1:T}, a_{2:T}) &= P(o_{1:T})\, P(a_{2:T}|o_{1:T})\, P(f_{1:T}|o_{1:T}, a_{2:T}) \\ &= \left[P(o_1) \prod_{t=2}^{T} P(o_t|o_{t-1}, a_t)\right]\left[\prod_{t=1}^{T} P(f_t|o_t)\right]\left[\prod_{t}^{T} P(a_t)\right]. \end{aligned} \tag{1}$$

The *observation model* is given as $P(f_t|o_t)$, representing the likelihood that feature f_t is generated by recognition state o_t. The transition probability is given by $P(o_t|o_{t-1}, a_t)$, which means that the transition to the next recognition state is dependent on the previous recognition state (i.e., the previous object class and orientation) as well as the action that has been applied. The posterior marginals $P(o_t|f_{1:T}, a_{2:T})$ of all hidden state variables given a sequence of observations can be efficiently computed using the *forward-backward algorithm*. The algorithm computes a smoothed estimate given all evidence in two passes, with the first pass going forward in time while the second pass going backwards in time:

$$\begin{aligned} P(o_t|f_{1:T}, a_{2:T}) &= P(o_t|f_{1:t}, f_{t+1:T}, a_{2:t}, a_{t+1:T}) \\ &\propto P(f_{t+1:T}, a_{t+1:T}|o_t)\, P(o_t|f_{1:t}, a_{2:t}). \end{aligned} \tag{2}$$

In the first pass, we compute the forward probabilities $P(o_t|f_{1:t}, a_{2:t})$, i.e., the probabilities for all $t \in \{1, \ldots, T\}$ ending up in a particular state o_t given the first t observations in the sequence. In the second step, we compute the probabilities $P(f_{t+1:T}, a_{t+1:T}|o_t)$ of the remaining observations given any starting point t.

Observation Model: For recognition, we learn a function $g : \mathbb{R}^{|f|} \rightarrow \mathbb{N}$ that maps the extracted object features in $\boldsymbol{f}$ to a unique object identifier $i \in \{1, \ldots, K\,Q\}$, each pointing to a distinct object model $o^i = \{c^k, \theta^q\}$, with $k \in \{1, \ldots, K\}$ and $q \in \{1, \ldots, Q\}$. K denotes the total number of object classes c^k and Q defines the number of discretized object poses θ^q per object class. Learning multiple models per object class allows us to train a model for a partial view of an object. This enables inference about an object's pose and reduces the variance in the model. The function is learned in a supervised fashion, with all possible features $\boldsymbol{f} = [f^1, \ldots, f^N]$ and the unique identifier i available as the value pair $(\boldsymbol{f}, i)$ at model training phase. While we assume that all features of an object are observed at training phase, we do not make this assumption during prediction, where not all components of **f** are observed due to feature selection.

To deal with the missing data problem at prediction time, we use a Gaussian generative model of the form $p(f|o^i) = \prod_{m=1}^{M} \mathcal{N}(f|\mu_{o^i,m}, \sigma_{o^i,m})$. M is the number of independent normal distributions to model a feature f. Each feature f is represented as a feature histogram with M bins, and the mean $\mu_{o^i,m}$ and variance $\sigma_{o^i,m}$ for each bin m are computed over the training data of each object model o^i. We use M independent normal distributions instead of a multivariate normal distribution to avoid high-dimensional features, which usually perform rather poorly due to the curse of dimensionality. Defining the problem as an independent Gaussian generative classifier has the additional advantage of handling missing features by marginalizing them out at prediction time, i.e., instead of inferring about the missing features, we simply perform inference on a subset of the full feature vector $\boldsymbol{f}$.

The Gaussian representation of the features is compact and fast to train, since it only requires the computation of the Gaussian feature parameters $\mu_{o^i,m}$ and $\sigma_{o^i,m}$. Sequential learning as well as sequential updates of mean and variance are possible, which has the potential of adaptively adding new observations to our trained features. For example, we can compute the observation model for the full feature vector $\boldsymbol{f}$ as

$$p(\boldsymbol{f}|o^i) = \prod_{n=1}^{N} \psi_n \prod_{m=1}^{M} \mathcal{N}(f^n|\mu_{o^i,m}, \sigma_{o^i,m}), \tag{3}$$

with ψ_n denoting a scaling factor which scales the different feature distributions.

4 Action Selection

In the case of object recognition, the task of action selection is twofold. On the one hand, we would like to select the lowest possible number of viewpoints that make an informed decision about an object possible. On the other hand, however, certain viewing choices or combinations of features may be well explained by more than one hypothesis. This means, the measurement distributions that correspond to the competing hypotheses are similar for a particular viewpoint. With respect to optimization, we want to select the actions that yield the most useful information and to stop acquiring and incorporating new observations once we have reached a desired level of certainty about an object.

Our Bayesian inference framework leads naturally to an iterative information-theoretic approach for deciding on the next action a_{t+1}. Action selection attributes a *utility score* $s_t(a, f)$ to each possible next action a and feature f. Possible next actions are either to stay (i.e., $a = \emptyset$) and compute a next feature f, or to move and compute a new first feature from a different observation location; the observation location is selected from the discrete set of all possible next viewing positions. The next action is eventually evaluated by $f_{t+1} = \text{argmax}_f \; s_t(a, f)$, for the best feature f while fixing $a = \emptyset$, and by $a_{t+1} = \text{argmax}_a \; s_t(a, f)$, selecting the best a (without selecting f). Several methods from information theory have been shown to work well in action selection tasks. The more popular methods include *expected loss of entropy* (LE) [6, 16], *mutual information* (MI) [2, 8] and *Kullback–Leibler divergence* (KL) [13], which we adapted in our framework for the action selection task.

Convergence and Stopping Criterion: The sequential update allows to acquire new information and with that reduce the uncertainty of the belief state. Unfortunately, one cannot guarantee that the sequential decision process reduces the uncertainty in every single step. But, on average, by utilizing information-theoretic action selection, the uncertainty will be reduced. The problem with information-theoretic methods is to find an adequate *stopping criterion*. In the case of feature selection, we need a stopping criterion that defines when to stop adding new features from the current location and rather move to a new location. Similarly, for viewpoint selection we need a stopping criterion to know when we can terminate the object recognition process and rely on the observations and inference we have made so far. The results of an action selection method highly depend on the stopping criterion as well as the parameters controlling that criterion. Wrong choices can lead to sub-optimal performance, with all possible actions a being processed in the worst case.

4.1 Feature Selection

Feature selection can be performed at training time or at prediction time. At training time, the feature space is reduced in advance by selecting relevant features, such that the loss of significant information is minimal. At prediction time, features are selected and computed online. In this work, we train all features and select features only at prediction time. This keeps the complexity low during training and facilitates updates by adding new features without the need to re-train the entire training set.

We want to compute relevant features, i.e., the features should provide additional information and preferably result in a unimodal posterior distribution. This becomes especially important on a robot platform with minimal computation power and short operation time. In the following subsections, we describe two methods for feature selection, one using LE and one using MI.

4.1.1 Feature Selection by Expected Loss of Entropy

Maximum LE indicates a feature that, if extracted next, would lead to the highest reduction in entropy of the posterior distribution on average. In other words, the selection of this feature reduces the uncertainty about our hypothesis on average the most. We compute the average LE because we cannot be certain about which object we are currently observing.

We compute the utility score $s_t(a, f)$ for a null action $a = \emptyset$ and a next possible feature f, based upon the confidence in the current posterior probability $P(o^i | f_{1:t}, a_{2:t})$ obtained from (2), as

$$s_t(a, f) = \sum_{i=1}^{K\,Q} P(o^i | f_{1:t}, a_{2:t})\, E[\varDelta H(o^i, f_{1:t}, f, a_{2:t})], \tag{4}$$

with $K\,Q$ object models in the model set. In this formulation, LE is weighted by the current belief of the object class o^i being the correct hypothesis, and estimated as the expectation $E[\cdot]$ of the difference in entropy between two subsequent posterior distributions,

$$E[\varDelta H(o^i, f_{1:t}, f, a_{2:t})] = H(o_t | f_{1:t}, a_{2:t}) - \int p(f | o^i)\, H(o_t | f_{1:t}, f, a_{2:t})\, \mathrm{d}f\,. \tag{5}$$

As we do not know which feature to expect, we need to integrate over the entire feature space. The feature distribution $p(f | o^i)$ is given by the observation model. The Shannon entropy $H(\cdot)$ measures the amount of uncertainty about the object model o and can be written as $H(o_t | f_{1:t}, a_{2:t}) = -\sum_{i=1}^{K\,Q} P(o^i | f_{1:t}, a_{2:t}) \log P(o^i | f_{1:t}, a_{2:t})$ in the discrete case.

Since (5) is not possible to compute in closed form, we approximate by sampling from the feature distribution, and obtain

$$E[\varDelta H(o^i, f_{1:t}, f, a_{2:t})] \approx H(o_t | f_{1:t}, a_{2:t}) - \sum^{\varGamma} H(o_t | f_{1:t}, f, a_{2:t}), \tag{6}$$

with $\varGamma$ being the number of samples drawn from the feature distribution $P(f | o^i)$. We can further approximate (6) by sampling with zero variance, which yields the mean μ_{o^i} of our trained object model o^i, and with that, a *mean feature* vector $\boldsymbol{f}_{\mu_{o^i}}$. In the case of selecting the next feature, the mean feature vector contains only one feature $f_{\mu_{o^i}}$, which leads to

$$E[\varDelta H(o^i, f_{1:t}, f, a_{2:t})] \approx H(o | f_{1:t}, a_{2:t}) - P(f_{\mu_{o^i}} | o^i)\, H(o | f_{1:t}, f_{\mu_{o^i}}, a_{2:t})\,. \tag{7}$$

The evaluation of $s_t(a, f)$ is quite costly to compute, since it involves the sequential update of the posterior distribution given the predicted observation. Even with the mean feature approximation of (7), this still involves the computation of $K\,Q$ predictive posteriors. Restricting attention to only the most probable o^i object hypothesis can further reduce the computational cost.

Stopping Criterion: For LE, we stop integrating more features when the loss of entropy for the selected action is small, $max(s_t(a, f)) < \tau_1$.

4.1.2 Feature Selection by Mutual Information

In contrast to LE, which uses the predictive posterior distribution, MI uses the conditional entropy. Compared to the predictive posterior distribution, the conditional entropy is much cheaper to evaluate, making the next action fast to compute in comparison. The utility score is computed as $s_t(a, f) = I(o_t; f|a)$, by letting $a =$, from the averaged MI,

$$I(o_t; f) = \sum_{i=1}^{K\,Q} P(o^i|f_{1:t}, a_{2:t}) \int p(f|o^i) \log \frac{p(f|o^i)}{\sum_{j=1}^{K\,Q} p(f|o^j)} \, \mathrm{d}f \,. \tag{8}$$

Since this quantity again is not possible to compute in closed form, we also approximate (8) with the mean feature $f_{\mu_{o^i}}$ by

$$I(o_t; f) \approx \sum_{i=1}^{K\,Q} P(o^i|f_{1:t}, a_{2:t}) \; P(f_{\mu_{o^i}}|o^i) \log \frac{P(f_{\mu_{o^i}}|o^i)}{\sum_{j=1}^{K\,Q} P(f_{\mu_{o^i}}|o^j)} \,. \tag{9}$$

Here, the mean feature once again is one particular feature, namely the feature we want to test for its information content.

Stopping Criterion: We stop to select a new feature from the current location when $s_t(a, f)$ decreases compared to the previous value [21].

4.2 View Planning

The goal in view planning is to evaluate and select new viewpoints that yield us novel and useful information about the object. Initially, we have not observed anything. So every observation has equal usefulness under the assumption that we do not have any prior knowledge and a uniform belief state. However, every subsequent observation from a different viewpoint bears additional information that may help us to identify the object correctly. The problem consists of not only finding new observation locations that provide previously unseen information, but also viewing positions that help us to improve our current belief of the object in question. In the following subsections, we formulate three methods for viewpoint selection based on three different information-theoretic measures, namely LE, MI and the KL-based *Jeffrey's divergence*.

Stopping Criterion: The stopping criterion for whether to select a new observation location or to terminate depends on the entropy. We will return the MAP estimate of the posterior distribution whenever $H(P(o_t|f_{1:t}, a_{2:t}) < \tau_2$.

4.2.1 Viewpoint Selection by Expected Loss of Entropy

Similar as with LE for feature selection, we compute the next best viewpoint by using the approximation via the mean feature vector $\boldsymbol{f}_{\mu_{o^i}}$. This time, the mean feature vector contains all features and not just a subset. Since we sample the mean feature from our model, there is no overhead in feature computation, as there would be for making an actual observation. Given the set of actions to move to possible viewing positions, we now compute the utility score $s_t(a, f)$ for an action $a \neq \emptyset$ and a feature f as

$$s_t(a, f) = \sum_{i=1}^{K\,Q} P(o^i | f_{1:t}, a_{1:t})\; E[\Delta H(o^i, f_{1:t}, f, a_{2:t}, a)]\,. \tag{10}$$

We can see that we again have to compute a complete update of the posterior distribution—this time, however, also including the transition of the sensor from the currently believed state to a new state by performing action a. $E[\Delta H(o^i, f_{1:t}, f, a_{1:t}, a)]$ in (10) is then obtained by a “mean feature” approximation,

$$E[\Delta H(o^i, f_{1:t}, f, a_{2:t}, a)] \approx H(o_t | f_{1:t}, a_{2:t}) - P(\boldsymbol{f}_{\mu_{o^i}} | o^i, a) H(o_t | f_{1:t}, \boldsymbol{f}_{\mu_{o^i}}, a_{2:t}, a)\,. \tag{11}$$

4.2.2 Viewpoint Selection by Mutual Information

As seen for feature selection before, the difference between MI and LE is that the second term in (11) is the conditional entropy instead of the entropy of the posterior. The utility score again is directly obtained from MI, $s_t(a, f) = I(o_t; f | a)$, where MI for viewpoint selection is now defined as follows

$$I(o_t; f | a) \approx \sum_{i=1}^{K\,Q} P(o^i | f_{1:t}, a_{2:t})\; P(\boldsymbol{f}_{\mu_{o^i}} | o^i, a)\; \log \frac{P(\boldsymbol{f}_{\mu_{o^i}} | o^i, a)}{\sum_{j=1}^{K\,Q} P(\boldsymbol{f}_{\mu_{o^i}} | o^j, a)}\,. \tag{12}$$

4.2.3 Viewpoint Selection by Jeffrey’s Divergence

Another way to choose new viewing positions is by selecting viewpoints such that, regardless of the true object and pose under observation, the most likely distributions of the resulting measurements are predictably dissimilar. Therefore, a new viewpoint must be selected such that competing hypotheses will appear as different as possible, and that the distinction measured by the features is increased. Finally, the best action is evaluated by computing

$$s_t(a, f) = \sum_{i=1}^{K\,Q} \sum_{j=1}^{K\,Q} P(o^i | f_{1:t}, a_{2:t})\, P(o^j | f_{1:t}, a_{2:t})\, J(P(\boldsymbol{f}_{\mu_{o^i}} | o^i, a) \parallel P(\boldsymbol{f}_{\mu_{o^i}} | o^j, a))\,, \tag{13}$$

where $J(P(\boldsymbol{f}_{\mu_{o^i}} | o^i, a) \parallel P(\boldsymbol{f}_{\mu_{o^i}} | o^j, a))$ is the Jeffrey's divergence, defined as $J(P_1 \parallel P_2) = KL(P_1 \parallel P_2) + KL(P_2 \parallel P_1)$, and $KL(P_1 \parallel P_2) = \int_{-\infty}^{\infty} p_1(x) \log \frac{p_1(x)}{p_2(x)}\, \mathrm{d}x$ denotes the relative entropy or KL-divergence for distributions P_1 and P_2. The KL-divergence presents the difference in information if samples from the distribution P_1 are assumed to be samples from P_2 .

5 Experiments

To test the framework we have described in the previous sections, we ran extensive evaluations on real world data. In this section, we report on the performance of the individual modules and then demonstrate the performance of the complete system.

5.1 RGB-D Object Dataset

We evaluated our object recognition and action selection framework on the publicly available RGB-D object dataset introduced in [12]. The dataset contains 41 877 data points, each comprising a color image, a 3D point cloud and the orientation of the object, for a total of 300 everyday objects. For each object, the data is recorded from viewing directions emanating from the circumference all around the object, including the three different viewing angles (angles of inclination) of 30°, 45° and 60° per direction. The overall dataset is challenging, since the objects in the dataset can look very similar, especially when they belong to the same object category. Furthermore, the objects in the dataset exhibit large changes in illumination due to the different viewing angles from which the objects were captured.

We compute a feature vector $\boldsymbol{f}$ that contains $N = 4$ independent features: object bounding box, color histogram, SIFT [14] and viewpoint feature histogram (VFH) [19]. Each feature represents a different object property: size, color, scale-invariant image description and geometry. In accordance with (3), each single feature or component of a feature histogram, respectively, is expressed as independent normal distribution. As we will see, the color feature is the fastest and most powerful single feature among the four features, and thus the first feature we evaluate at each location.

In all our experiments, we use the same evaluation procedure as proposed in [12] and also used in [5]. We test our framework on the task of recognizing the instance of an object. We train our object models from $\boldsymbol{f}$ using data captured from viewpoints at viewing angles of 30° and 60° in each direction, and test them against features computed from data that is recorded from viewpoints at viewing angles of 45°. Based on the computed features, we generate $L = 8$ different object models per object instance, which clusters the viewing directions along the circumference horizontally into bins spanning 45°. This subdivision trades computation cost off against performance, and defines the accuracy of the objects' pose estimates. All the reported accuracies of our multi-view object recognition results are expressed as *recall* and are average values over 10 trials. Regarding the stopping criteria, we terminate the action selection procedure whenever either the entropy of the posterior distribution has been reduced to $\tau_2 = 0.1$ or the MAP estimate of the posterior distribution exceeds 60% in accuracy. After termination, we use the MAP estimate of the posterior distribution after (2) to determine the most likely object. For the LE formulation in feature selection we set $\tau_1 = 0.1$ and after termination take as well the MAP estimate as the most likely object.

5.2 *Online Feature Selection*

The first experiment confirms the usefulness of online feature selection in a single-view recognition task. Selecting the right subset of features can reduce the computation cost while keeping comparable recognition performance.

Each feature has a cost associated for computation; the average computation times of our features are as follows: 0.01 s for the bounding box, 0.0005 s for the color histogram, 0.025 s for the SIFT features, and 2.0 s for the VFH. Figure 3 shows the average total cost of computation per object and the recognition accuracy plotted against individual feature combinations. The color feature seems to have the best single recognition performance and is the cheapest to compute. The VFH on the other hand does not have a lot of recognition power as such and is rather expensive. From

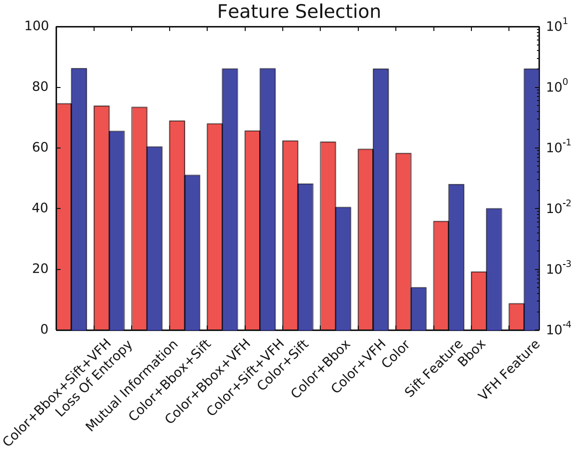


Fig. 3 Performance and cost of different feature combinations in single-view object recognition. The recognition accuracies are displayed as *red* bars (*left side* scale, in %). The average cost of computing the feature combinations per object is shown in log scale as *blue* bars (in seconds)

Table 1 Individual performance for single-view object recognition. "Loss Entropy" and "Mutual Information" denote the feature selection method, with "F Cost" being the average cost per object for computing the features and "FS Cost" being the average cost per object for feature selection

Method	Accuracy (%)	F cost (s)	FS cost (s)	Total (s)
C+B+S	68.91	0.04	0.00	0.04
C+B+S+V	74.59	2.04	0.00	2.04
Loss entropy	73.82	0.19	1.05	1.24
Mutual information	73.21	0.1	0.2	0.3
Reference [12] (kSVM)	74.80	–	–	–
Reference [4] (Kernel Descriptor)	91.2	3.2	–	3.2

a cost versus performance standpoint, the feature combination {Color, Bbox, SIFT} performs best with an accuracy of 68.91% and a low average feature computation cost of 0.04 s. However, note that the VFH also has its value since, if included, it increases the recognition accuracy to 74.59%, as detailed in Table 1. Additionally, we can see that using feature selection (MI, LE) reduces the cost drastically while keeping recognition accuracy similar to using all features.

Table 1 further shows that the average total cost of feature computation can be considerably reduced by making use of feature selection (see the third and fourth row of column "F Cost" in Table 1). Here, we utilize LE and MI according to Sect. 4.1 in the feature selection methods. Among the different methods, the LE measure seems to select more features, and more importantly, is much more expensive to compute than MI because we need to update the posterior distribution and compute a full update step.

We compared all results with the results of the original dataset in [5, 12]. The first paper has very comparable results to our HMM framework and uses similar features to ours, no timing has been provided but it is probably slightly higher than our features. The latter paper reports much higher accuracies; this is mainly due to a more descriptive feature which seems to generalize very well. The cost for such a feature, however, is very high computation time. Moreover, the features used in the two papers are learned in batch, which makes them less flexible compared to our sequential feature learning approach.

5.3 *Multi-view Object Recognition*

Although the computation time can be substantially reduced through feature selection, the achievable recognition accuracies are limited around 75% for single-view object recognition. The reasons are the relative simplicity of our four feature types and their limitation in generalizing well for the given RGB-D dataset. However, even with fairly simple features we can achieve improved recognition accuracies for both object class and pose by utilizing an information-theoretic multi-view object recognition approach. In the following tests, we use the MI measure for feature selection, since it is on par with the LE measure regarding recognition performance, and in addition, has lower computation cost (see Table 1).

Table 2 shows the performance and Table 3 summarizes the cost of multi-view object recognition. The first row shows that using all features (i.e., no feature selection) has a very high cost but random viewpoint selection already increases the recognition accuracy by about 10%. In the second row, due to feature selection using mutual information and random viewpoint selection, we can see a gain in reduced computation time and an increase in accuracy. On average, random viewpoint selection terminates after a little more than 3 observations. By using the information-theoretic

Table 2 Performance of our multi-view object recognition framework. All but the first row uses the MI measure in the feature selection method. We show the average recognition accuracy, the average number of observations and the average bin error of estimated viewing directions. Each bin represents a range of observation angles of 45°

Method	Accuracy	Observations	Avg. bin error
C+B+S+V (no FS)	84.48% ± 1.4	3.4 ± 0.2	2.41 ± 0.07 (bins) (±67.5°)
Random	86.76 % ± 0.4	3.2 ± 0.2	2.19 ± 0.02 (bins) (±67.5°)
Jeffrey’s divergence	88.34 % ± 0.6	2.7 ± 0.1	1.62 ± 0.01 (bins) (±45.0°)
Loss entropy	89.82 % ± 1.3	2.4 ± 0.2	1.54 ± 0.01 (bins) (±45.0°)
Mutual information	91.87 % ± 0.7	2.5 ± 0.1	1.48 ± 0.01 (bins) (±45.0°)

Table 3 Average computation cost per object for our multi-view object recognition framework. The columns show the average cost for computing the feature (“F Cost”), average cost for feature selection (“FS Cost”) and average cost for viewpoint selection (“VS Cost”)

Method	F cost (s)	FS cost (s)	VS cost (s)	Total cost (s)
C+B+S+V (no FS)	7.02 ± 0.42	0.00	0.0	7.02 ± 0.42
Random	0.40 ± 0.04	0.83 ± 0.05	0.0	1.23 ± 0.09
Jeffrey’s divergence	0.30 ± 0.03	0.58 ± 0.02	3.E–4 ± 2.E–4	0.88 ± 0.05
Loss entropy	0.23 ± 0.04	0.50 ± 0.04	5.4 ± 0.6	6.13 ± 0.68
Mutual information	0.24 ± 0.03	0.51 ± 0.03	1.1 ± 0.2	1.85 ± 0.26

measures for viewpoint selection, we find that LE, MI and Jeffrey's divergence all perform similarly well—with MI being the best in terms of recognition accuracy. The lowest computation time is achieved by using the Jeffrey's divergence because it can compute new viewing positions in almost no time. The MI formulation is superior to the LE formulation in speed, since it does not need to compute the predictive posterior.

We further evaluate our multi-view approach for estimating an object's pose. As previously mentioned, we train each object with a number of different object models, in our case $L = 8$ models per object. When observing a feature, we can infer from which model it has most likely been generated, and by this, further infer about the orientation of the object. The performance of the object's pose estimation is presented in Table 2. The table shows the average bin error of the estimated viewing directions, which is computed from the differences between the inferred and the true viewing angles. For the first row, all the features were computed (i.e., no feature selection); the feature selection methods with different information-theoretic measures were used for the remaining rows. We are on average >2 bins off in our estimation without view planning (i.e., random selection of views), which corresponds to an average pose error of about 67.5°. If we apply one of the three more sophisticated viewpoint selection methods, we yield an average bin error of <2 bins. This results in an average pose error of 45°. For comparison, the experiments in [5] report an average pose error of 44.8°, which is on par with our results.

5.4 Quadcopter Experiments

Finally, to show the overall performance of our active multi-view object recognition framework, we have conducted experiments on our custom-built quadcopter platform. As shown in Fig. 4, the quadcopter is equipped with an Xtion RGB-D camera for data acquisition and can fly autonomously to a new observation location.

Given an object, the task is to fly the quadcopter around the object and take as many observations as necessary to correctly recognize the object. Compared to the simulated experiments above, where data is captured with a RGB-D camera mounted on a tripod, experiments on a robotic platform are generally more challenging. Specific challenges regarding object recognition on a fast moving platform include motion blur and constant changes in illumination.

As we do not have any of the objects used in the RGB-D dataset at hand, we captured 10 new objects and added them to the dataset, which now consists of a total of 310 objects. We captured the data in the same way as described in [12] and chose objects similar to the ones already contained in the dataset. We trained our models with all 310 objects and observations taken at the two viewing angles of 30° and 60°. In the actual experiment, the quadcopter then flies around the object at a distance and height that allows for a viewing angle of 45° roughly.

In each run of the experiments, the robot starts at a random position with respect to the object, from where it acquires the first observation of the object. From there on,

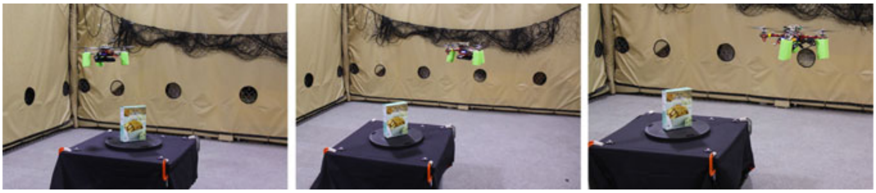


Fig. 4 Quadcopter experiment and test setup. The quadcopter robot, equipped with a RGB-D camera, collects multiple views of a target object (e.g., a cereal box) in an active object recognition task

Table 4 Single-view and multi-view object recognition based on data from the quadcopter robot. In both cases, results with and without feature selection are presented. Advantages of online feature selection as well as multi-view object recognition are revealed by the experiments

Method	Accuracy	Obsv.	Avg. bin error	Avg. Cost (s)
Single-view				
C+B+S+V (no FS)	82.16 %	1	2.1(bins)(±67.5°)	2.14
Feature Selection (MI)	87.32 %	1	1.9(bins)(±45°)	0.19
Multi-view				
Mutual information (no FS)	96.5 % ± 0.5	2.0 ± 0.1	1.6± 0.1(bins)(±45°)	5.63 ± 0.3
Mutual information	99.8 % ± 0.1	1.6 ± 0.01	1.1± 0.1(bins)(±45°)	1.48 ± 0.2

it selects the following actions as described in the previous section—either inferring about another feature from f or flying to a new viewpoint. We terminate the process once our stopping criteria are reached. Figure 4 shows an example of the quadcopter taking observations of a cereal box object during one of the experiments.

Table 4 summarizes the experimental results from the quadcopter experiments, averaged over eight runs with different starting locations. We can see that for all experiments, whether conducted with single-view or multi-view object recognition, online feature selection not only helps to decrease the computation time but also increases the recognition accuracy. The average pose error is similar to the error in the simulation experiments, however, the average bin error is smaller, which results in more correctly classified bins in total.

Multi-view recognition on average only required 1.6 observations, which means that often we are able to make a confident decision after only one scan. In some instances, however, by acquiring additional observations, the recognition accuracy could be increased by more than 10–99.8%. This of course comes with a price of actually moving the robot to a new location which furthermore increases the cost of recognizing the object. In practice, whether the cost of moving is justified, is probably a tradeoff between accuracy and energy saving. However, we can clearly see that adaptive feature selection has huge benefits by decreasing computation time, which is very beneficial for vehicles with low energy budget like quadcopters.

6 Conclusion and Future Work

We have presented an information-theoretic action selection framework for object recognition that is capable of sequential training and classification, view planning as well as online feature selection. Online feature selection is especially useful since it is adaptive, can reduce computation time drastically and is more flexible, compared to offline feature selection. Our framework was shown to lead to increased recognition accuracies by using four relatively simple features in combination with the multi-view object recognition approach. Furthermore, we demonstrated that information-theoretic formulations perform better than random action selection strategies in that they require fewer observations and less computation time.

Compared to state-of-the-art single-view object recognition methods, our multi-view object recognition approach achieves comparable results. Our approach is not primarily meant to replace single-view methods but rather to complement them. Put in other words, if a confident decision can be made about an object after only one observation, there will be no need for taking a second view. However, there will always be objects with inherent ambiguity, and it is in those cases in which the multi-view approach is brought to bear. Feature selection is useful regardless of the number of views and the object in question, since it reduces the dimensionality of the feature space and decreases the overall computation cost.

The kernel features used in [5] seem to be far superior to our simple features. Thus, the question arises how our approach would perform when using these more complex features instead. We are currently in the process of experimenting with kernel features and will test them with our framework.

Acknowledgements The authors would like to thank Jörg Müller for his help with the development of the quadcopter robot platform.

References

1. Ali, H., Marton, Z.C.: Evaluation of feature selection and model training strategies for object category recognition. In: Intelligent Robots and Systems, pp. 5036–5042. IEEE/RSJ (2014)
2. Atanasov, N., Sankaran, B., Ny, J.L., Pappas, G.J., Daniilidis, K.: Nonmyopic view planning for active object classification and pose estimation. IEEE Trans. Robot. **30**(5), 1078–1090 (2014)
3. Bajcsy, R.: Active perception. Proc. IEEE **76**(8), 966–1005 (1988)
4. Bo, L., Ren, X., Fox, D.: Depth kernel descriptors for object recognition. In: International Conference on Intelligent Robots and Systems, pp. 821–826. IEEE (2011)

5. Bo, L., Ren, X., Fox, D.: Unsupervised feature learning for RGB-D based object recognition. In: Experimental Robotics - The 13th International Symposium on Experimental Robotics, pp. 387–402 (2012)
6. Borotschnig, H., Paletta, L., Prantl, M., Pinz, A.: Active object recognition in parametric eigenspace. In: Proceedings of the BMVC, pp. 63.1–63.10 (1998)
7. Brown, M., Lowe, D.G.: Unsupervised 3D object recognition and reconstruction in unordered datasets. In: 5th International Conference on Digital Imaging and Modeling, pp. 56–63 (2005)
8. Denzler, J., Brown, C.M.: Information theoretic sensor data selection for active object recognition and state estimation. Trans. pattern Anal. Mach. Intell. **24**(2), 145–157 (2002)
9. Finman, R., Whelan, T., Kaess, M., Leonard, J.J.: Toward lifelong object segmentation from change detection in dense RGB-D maps. In: European Conference on Mobile Robots (2013)
10. Grabner, H., Bischof, H.: On-line boosting and vision. In: Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 260–267 (2006)
11. Hollinger, G.A., Mitra, U., Sukhatme, G.S.: Active classification: theory and application to underwater inspection. In: International Symposium on Robotics Research (ISRR). Flagstaff, AZ (2011)
12. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view RGB-D object dataset. In: International Conference on Robotics and Automation, pp. 1817–1824. IEEE (2011)
13. Laporte, C., Arbel, T.: Efficient discriminant viewpoint selection for active Bayesian recognition. Int. J. Comput. Vis. **68**(3), 267–287 (2006)
14. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
15. Luber, M., Spinello, L., Arras, K.O.: People tracking in RGB-D data with on-line boosted target models. In: International Conference on Intelligent Robots and Systems, pp. 3844–3849. IEEE/RSJ (2011)
16. Paletta, L., Pinz, A.: Active object recognition by view integration and reinforcement learning. Robot. Auton. Syst. **31**(1), 71–86 (2000)
17. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)
18. Potthast, C., Sukhatme, G.S.: A probabilistic framework for next best view estimation in a cluttered environment. J. Vis. Commun. Image Represent. **25**(1), 148–164 (2014)
19. Rusu, R.B., Bradski, G., Thibaux, R., Hsu, J.: Fast 3D recognition and pose using the viewpoint feature histogram. In: International Conference on Intelligent Robots and Systems. IEEE/RSJ (2010)
20. Salvagnini, P., Pernici, F., Cristani, M., Lisanti, G., Del Bimbo, A., Murino, V.: Non-myopic information theoretic sensor management of a single pan-tilt-zoom camera for multiple object detection and tracking. Comput. Vis. Image Underst. **134**, 74–88 (2015)
21. Verleysen, M., Rossi, F.: Advances in Feature Selection with Mutual Information (CoRR). Springer, Berlin (2009)
22. Zhou, X.S., Comaniciu, D., Krishnan, A.: Conditional feature sensitivity: a unifying view on active recognition and feature selection. In: 9th International Conference on Computer Vision (2003)

An Iterative Kalman Smoother for Robust 3D Localization and Mapping

Dimitrios G. Kottas and Stergios I. Roumeliotis

1 Introduction and Related Work

Over the past decade, localization systems fusing inertial data, from an inertial measurement unit (IMU), with visual observations, from a camera [i.e., vision-aided inertial navigation systems (VINS)] have become a popular choice for GPS-denied navigation (e.g., indoors [1], or in space [2]). The dramatic upswing in manufacturing of low-cost, miniature IMUs and cameras, combined with the increasing capabilities of embedded computers, have made mobile devices (e.g., cell phones) excellent platforms for VINS, but raised also new challenges, when designing algorithms with improved robustness and efficiency characteristics.

The Maximum a Posteriori (MAP) estimator for VINS, corresponds to a batch least-squares (BLS) problem, over the platform's trajectory, and the map of the environment. Unfortunately, BLS approaches cannot serve time-critical navigation applications (e.g., augmented reality), due to their unbounded processing and memory requirements, as the problem size increases with time [3].

Filtering approaches, on the other hand, maintain bounded processing time, by optimizing over a sliding window of the most recent visual and inertial measurements, while absorbing past measurements into a prior cost term. Depending on their representation of the prior information, filtering methods can be classified into extended Kalman filters (EKFs) and inverse filters (INVFs) [4].

EKF-based algorithms exhibit excellent numerical stability, and have been demonstrated in real-time VINS implementations [5, 6]. Despite their efficiency, however, they do not allow re-processing the visual and/or inertial measurements within

D.G. Kottas (✉) · S.I. Roumeliotis
Department of Computer Science and Engineering, University of Minnesota,
Minneapolis, MN, USA
e-mail: dkottas@cs.umn.edu

S.I. Roumeliotis
e-mail: stergios@cs.umn.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_28

the optimization window, which may severely affect their performance under challenging conditions. Specifically, consider visual observations, that arrive as feature tracks, spanning a set of images within the estimator's sliding window. EKF-based approaches (e.g., the MSC-KF [2]), postpone their processing till they have reached their maximum length, which causes a delay in the state correction equal to the time between when a feature track is available for processing (i.e., when it spans at least two images), and when it is actually processed. Although small (typically less than 2 s for personal localization), such a delay can affect the accuracy of time-critical navigation systems. Furthermore, under scenarios with increased linearization errors [e.g., (re)-initialization (after failure) of the system's operation, when the available estimates of critical quantities, such as the IMU's velocity and biases, are of low accuracy], re-processing visual and inertial measurements, can significantly improve accuracy and robustness, by increasing their rate of convergence.

Such appealing capabilities of filtering algorithms, are supported by the INVF (e.g., [7–10]), which allows re-processing all inertial and visual measurements within the optimization window considered. Unfortunately, due to the high condition number of the Hessian, INVFs typically require 64-bit precision, reducing their efficiency, especially when considering that most current mobile devices feature ARM processors and NEON co-processors that provide a 4-fold processing speedup when using 32-bit precision. Note also that, with the exception of [11], existing INVFs for VINS do not classify visual observations based on their track length, not allowing their efficient processing (e.g., as in the MSC-KF) when possible.

To overcome the limitations of the EKF and INVF when applied to visual-inertial odometry (VIO), in [12], we introduced an iterative Kalman smoother (IKS), that shares the advantages of both approaches. In this paper, we extend the methodology of [12], to the most general case for VINS, which allows (re)-processing visual data either using a VIO approach (as in the MSC-KF), or as SLAM landmarks when their track length exceeds the estimator's sliding window. In particular, we introduce a sliding window IKS for VINS with the following key advantages:

- The IKS iteratively re-linearizes both inertial and camera measurements within the estimator's window, and re-processes visual data over multiple overlapping sliding-window epochs, thus improving robustness and increasing accuracy.
- The IKS employs a covariance matrix, as well as a set of linearized constraints (instead of a Hessian) for representing prior information, thus inheriting the superior numerical properties of the EKF and leading to efficient implementations.

Besides the detailed analysis and description of the proposed IKS algorithm, we demonstrate its robustness and assess its accuracy in simulations and experiments over challenging indoor VINS scenarios, including, filter initialization and scarcity of feature tracks due to sudden turns or camera occlusions. Finally, we provide a timing comparison between the IKS and the EKF, using a mobile processor.

2 Vision-Aided Inertial Navigation System (VINS)

The system state[1] at time t_k is given by $\mathbf{x}_k = \left[\mathbf{x}_{I_k}^T \; {}^{I_\ell}\boldsymbol{\ell}_k^T\right]^T$ where $\mathbf{x}_{I_k}$ contains all kinematic quantities, describing the motion of the IMU's frame of reference $\{I_k\}$, while ${}^{I_\ell}\boldsymbol{\ell}_k$ comprises landmarks of the environment. In particular, $\mathbf{x}_{I_k} = \left[{}^{I_k}\mathbf{q}_G^T \; {}^G\mathbf{p}_{I_k}^T \; {}^G\mathbf{v}_{I_k}^T \; \mathbf{b}_{a_k}^T \; \mathbf{b}_{g_k}^T\right]^T$, where ${}^{I_k}\mathbf{q}_G$ is the quaternion representation of the orientation of the global frame $\{G\}$ in $\{I_k\}$, ${}^G\mathbf{v}_{I_k}$ and ${}^G\mathbf{p}_{I_k}$ are the velocity and position of $\{I_k\}$ in $\{G\}$ respectively, while $\mathbf{b}_{a_k}$ and $\mathbf{b}_{g_k}$ correspond to the gyroscope and accelerometer biases. Finally, ${}^{I_\ell}\boldsymbol{\ell}_k$ comprises N_k landmarks, i.e., ${}^{I_\ell}\boldsymbol{\ell}_k = \left[{}^{I_\ell}\mathbf{f}_1^T \dots {}^{I_\ell}\mathbf{f}_{N_k}^T\right]^T$ where ${}^{I_\ell}\mathbf{f}_j$ denotes the inverse-depth parameterization of feature $\mathbf{f}_j$ in $\{I_\ell\}$.

2.1 Inertial Measurements

The IMU provides measurements of the platform's rotational velocity and linear acceleration, contaminated by white Gaussian noise and time-varying biases. Let $\mathbf{u}_{k,k+1}$ denote the inertial measurements within the time interval $\left[t_k, \; t_{k+1}\right]$, which through integration, define a constraint (discrete-time process model) of the form:

$$\mathbf{x}_{I_{k+1}} = \mathbf{f}(\mathbf{x}_{I_k}, \; \mathbf{u}_{k,k+1} - \mathbf{w}_{k,k+1}) \tag{1}$$

where $\mathbf{w}_{k,k+1}$ is a discrete-time zero-mean white Gaussian noise process with covariance $\mathbf{Q}_k$.[2] Linearizing (1), at the state estimates corresponding to the two consecutive states, $\mathbf{x}_{I_k}^\star$, $\mathbf{x}_{I_{k+1}}^\star$, results in the following IMU measurement model, relating the error states $\widetilde{\mathbf{x}}_{I_k}^\star$ and $\widetilde{\mathbf{x}}_{I_{k+1}}^\star$:

$$\widetilde{\mathbf{x}}_{I_{k+1}}^\star = \mathbf{r}_{u_{k,k+1}}^\star + \boldsymbol{\Phi}_{k+1,k}^\star \widetilde{\mathbf{x}}_{I_k}^\star + \mathbf{G}_{k+1,k}^\star \mathbf{w}_{k,k+1} \tag{2}$$

where $\mathbf{r}_{u_{k,k+1}}^\star := \mathbf{f}(\mathbf{x}_{I_k}^\star, \mathbf{u}_{k,k+1}) - \mathbf{x}_{I_{k+1}}^\star$, and we have defined the error state $\widetilde{\mathbf{x}}_{I_k}^\star$ as the difference between the true state $\mathbf{x}_{I_k}$ and the linearization point $\mathbf{x}_{I_k}^\star$ (i.e., $\widetilde{\mathbf{x}}_{I_k}^\star = \mathbf{x}_{I_k} - \mathbf{x}_{I_k}^\star$).[3] The Jacobians $\boldsymbol{\Phi}_{k+1,k}^\star$ and $\boldsymbol{G}_{k+1,k}^\star$ are evaluated at $\mathbf{x}_{I_k}^\star$, $\mathbf{x}_{I_{k+1}}^\star$, and are available in numerical or analytical form [1]. Although the corresponding cost term,

$$c_{u_{k,k+1}}(\tilde{\mathbf{x}}_{k:k+1}^\star) = ||\mathbf{r}_{u_{k,k+1}}^\star - \left[-\boldsymbol{\Phi}_{k+1,k}^\star \; \mathbf{I}\right] \begin{bmatrix} \tilde{\mathbf{x}}_{I_k}^\star \\ \tilde{\mathbf{x}}_{I_{k+1}}^\star \end{bmatrix} ||_{\mathbf{Q}_k^\star}^2$$

[1] Without loss of generality, we assume that the IMU-camera extrinsic calibration is the identity transformation and the clocks of the two sensors are perfectly synchronized. In practice, both are included in the system's state following the methodologies described in [13] and [6], respectively.

[2] The interested reader is referred to [1, 5, 14], and references there-in, for details on IMU integration.

[3] For the quaternion $\mathbf{q}$ we employ a multiplicative error model $\widetilde{\mathbf{q}} = \mathbf{q} \otimes \mathbf{q}^{\star -1} \simeq \left[\frac{1}{2}\delta\boldsymbol{\theta}_{3\times 1}^T \; 1\right]^T$.

where $\tilde{\mathbf{x}}^{\star}_{I_{k:k+1}} := \left[\tilde{\mathbf{x}}^{\star T}_{I_k}\ \tilde{\mathbf{x}}^{\star T}_{I_{k+1}}\right]^T$ and $\mathbf{Q}^{\star}_k = \mathbf{G}^{\star}_{k+1,k}\mathbf{Q}_k\mathbf{G}^{\star T}_{k+1,k}$, can be re-linearized multiple times within the INVF framework, current EKF-based approaches linearize (1) only *once*, during state and covariance propagation (i.e., every time a new inertial measurement becomes available). This limitation may negatively affect performance when the linearization errors are large (e.g., during system initialization).

2.2 Visual Observations

In order for a camera to provide kinematic information, a feature-tracking pipeline is required. A common choice comprises a feature-extraction method (e.g., the Harris corner [15]) along with a tracking algorithm (e.g., the Kanade-Lucas-Tomasi (KLT) [16]). Once a new image arrives, point features from the previous image, are tracked to the new one, while new features are extracted from areas that just entered the camera's field of view [1]. An example of the feature tracks generated from such an image-processing pipeline is shown in Fig. 1.

Consider a point feature $\mathbf{f}_j$, observed in camera poses $\mathbf{x}_{I_{k+1:k+N_j}}$, where $N_j \leq M$ and M is the window's length. We represent $\mathbf{f}_j$, using its homogeneous coordinates and inverse distance in $\{I_{k+1}\}$, hereafter denoted by ${}^{I_{k+1}}\mathbf{f}_j$, or using its Cartesian coordinates, ${}^{I_{k+1}}\mathbf{p}_{f_j}$. For the m-th measurement, $m \in \left[1, \ldots, N_j\right]$, the observation $\mathbf{z}_{k+m,j}$ acquired by a calibrated camera is:

$$\mathbf{z}_{k+m,j} = \pi(C({}^{I_{k+m}}\mathbf{q}_{I_{k+1}})\,{}^{I_{k+1}}\mathbf{p}_{f_j} + {}^{I_{k+m}}\mathbf{p}_{I_{k+1}}) + \mathbf{n}_{k+m,j} \qquad (3)$$

where, $\pi(\left[x\ y\ z\right]^T) = \left[\frac{x}{z}\ \frac{y}{z}\right]^T$, while $\mathbf{n}_{k+m,j} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_2)$, and $\mathbf{I}_2$ is the 2×2 identity matrix. Linearizing (3), yields:

$$\tilde{\mathbf{z}}^{\star}_{k+m,j} = \mathbf{H}^{\star}_{R,k+1,j}\tilde{\mathbf{x}}^{\star}_{I_{k+1}} + \mathbf{H}^{\star}_{R,k+m,j}\tilde{\mathbf{x}}^{\star}_{I_{k+m}} + \mathbf{F}^{\star}_{k+m,j}\,{}^{I_{k+1}}\tilde{\mathbf{f}}^{\star}_j + \mathbf{n}_{k+m,j}$$

Collecting all N_j observations of feature $\mathbf{f}_j$ in one vector, yields:

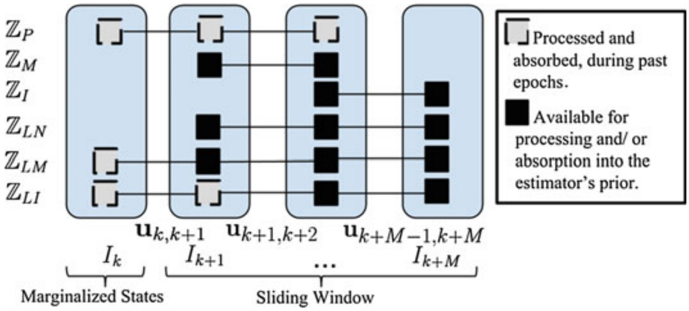


Fig. 1 At t_{k+M}, there exist 6 categories of feature tracks described in Sect. 2.3

$$\tilde{\mathbf{z}}_j^\star = \mathbf{H}_{R,j}^\star \tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star + \mathbf{F}_j^{\star\, I_{k+1}} \tilde{\mathbf{f}}_j^\star + \mathbf{n}_j \tag{4}$$

which corresponds to the cost term:

$$c_{z_{f_j}}(\tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star, {}^{I_{k+1}}\tilde{\mathbf{f}}_j) = ||\tilde{\mathbf{z}}_j^\star - \mathbf{H}_{R,j}^\star \tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star - \mathbf{F}_j^{\star\, I_{k+1}} \tilde{\mathbf{f}}_j||_{\sigma^2 \mathbf{I}}^2 \tag{5}$$

Consider an orthonormal matrix $\boldsymbol{\Theta}_j$, partitioned as $\boldsymbol{\Theta}_j = \begin{bmatrix}\mathbf{S}_j & \mathbf{U}_j\end{bmatrix}$, where the 3 columns of $\mathbf{S}_j$ span the column space of $\mathbf{F}_j^\star$, while the $2N_j - 3$ columns of $\mathbf{U}_j$, its left null space. Projecting (4) onto $\boldsymbol{\Theta}_j$, partitions $c_{z_{f_j}}$ into two parts:

$$\begin{aligned} c_{z_{f_j}}(\tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star, {}^{I_{k+1}}\tilde{\mathbf{f}}_j) &= ||\boldsymbol{\Theta}_j^T \left(\tilde{\mathbf{z}}_j^\star - \mathbf{H}_{R,j}^\star \tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star - \mathbf{F}_j^{\star\, I_{k+1}} \tilde{\mathbf{f}}_j\right) ||_{\sigma^2 \mathbf{I}_{2N_j}}^2 \\ &= ||\mathbf{r}_j^{K\star} - \mathbf{H}_j^{K\star} \tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star ||_{\sigma^2 \mathbf{I}_{2N_j - 3}}^2 + ||\mathbf{r}_j^{M\star} - \mathbf{H}_j^{M\star} \tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star - \mathbf{R}_j^{M\star\, I_{k+1}} \tilde{\mathbf{f}}_j||_{\sigma^2 \mathbf{I}_3}^2 \\ &= c_{z_{f_j}}^K(\tilde{\mathbf{x}}_{k+1:k+N_j}^\star) + c_{z_{f_j}}^M(\tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star, {}^{I_{k+1}}\tilde{\mathbf{f}}_j) \end{aligned} \tag{6}$$

with $\mathbf{r}_j^{K\star} = \mathbf{U}_j^T \tilde{\mathbf{z}}_j^\star$, $\mathbf{H}_j^{K\star} = \mathbf{U}_j^T \mathbf{H}_{R,j}^\star$, and $\mathbf{r}_j^{M\star} = \mathbf{S}_j^T \tilde{\mathbf{z}}_j^\star$, $\mathbf{H}_j^{M\star} = \mathbf{S}_j^T \mathbf{H}_{R,j}^\star$, $\mathbf{R}_j^{M\star} = \mathbf{S}_j^T \mathbf{F}_{R,j}^\star$. The second term, $c_{z_{f_j}}^M$ contains *all* information regarding feature f_j, while $c_{z_{f_j}}^K$ defines a multi-state constraint *only* among the poses $\mathbf{x}_{I_{k+1:k+N_j}}$. For feature tracks that do not exceed the estimator's window, as in the MSC-KF [2], we consider *only* the cost term $c_{z_{f_j}}^K$. Specifically, since $\mathbf{R}_j^{M\star}$ is an invertible square matrix and *for any* $\tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star$ *there exists* a ${}^{I_{k+1}}\tilde{\mathbf{f}}_j$, such that $c_{z_{f_j}}^M$ is exactly zero, minimizing (6) is equivalent to minimizing $c_{z_{f_j}}^K(\tilde{\mathbf{x}}_{I_{k+1:k+N_j}}^\star)$. As we describe later on, for feature tracks whose span exceeds the sliding window (i.e., SLAM landmarks), $c_{z_{f_j}}^M$ allows initializing them into the estimator's map, and subsequently optimizing their measurements across *non-overlapping* epochs of the sliding window.

2.3 Visual Observations in Sliding-Window Estimators

We classify visual observations, in sliding-window estimators, based on their:
(i) **Track length** which distinguishes SLAM landmarks from MSC-KF features, since for the latter, their track length does not exceed the estimator's window. Thus, optimizing over the MSC-KF feature's coordinates is not required for minimizing their corresponding cost terms [see (6)].
(ii) **Earliest observation** which if it involves the sliding window's "tail" (oldest) state, it does not allow postponing their absorption into the estimator's prior.
In particular, we distinguish the following categories of visual observations:

- **Past features** ($\mathbb{Z}_P$) corresponding to visual observations that were absorbed in a past epoch of the sliding window and cannot be re-processed by any filter.

- **MSC-KF features** whose tracking length does not exceed the estimator's window, i.e., $N_j \leq M$, hence they are not mapped but rather used only for providing multi-state constraints involving the camera poses observing them. Based on their earliest observation, we further classify MSC-KF features into 2 sets:
  - **Mature features** ($\mathbb{Z}_M$): These are MSC-KF features that have reached their maximum length. Both the EKF and the INVF linearize, process, and absorb them in a single step. Note also that the INVF, as well as the Iterative EKF (I-EKF) [4] can re-linearize these observations.
  - **Immature features** ($\mathbb{Z}_I$): This set represents feature tracks that have already entered the image-processing pipeline, but have not reached their maximum length, yet. Although the INVF can use (and re-linearize) these measurements multiple times, across overlapping epochs of the sliding window (from the second time they are observed till the track exits the optimization window), the EKF is not able to do so. This limitation introduces a delay, between the "birth" of a feature track and its impact on the filter's state estimates; a potential drawback of EKF-based approaches for time-critical applications.

- **SLAM landmarks** corresponding to features, whose track length exceeds the estimator's optimization window, i.e., $N_j > M$, and hence their optimal processing requires including them into the estimator's map of the environment, ${}^{I_{k+1}}\boldsymbol{\ell}_{k+M}$. Within a single epoch of the sliding window, observations to SLAM landmarks can be partitioned into 3 categories:
  - **Mature landmarks** ($\mathbb{Z}_{LM}$): These are observations to previously initialized SLAM landmarks, which include the estimator's "tail" pose, hence their absorption into the estimator's prior, cannot be postponed for future epochs of the sliding window.
  - **Immature landmarks** ($\mathbb{Z}_{LI}$): These correspond to measurements of previously initialized SLAM landmarks, which do not involve the estimator's "tail" pose, thus their absorption can be postponed to later epochs, till one of them includes the estimator's "tail" pose, i.e., they become mature landmarks.
  - **New SLAM landmarks** ($\mathbb{Z}_{LN}$): These are feature tracks that have not yet reached their maximum length, while their present observations span all cameras within the estimator's optimization window. Hence, they will be processed, absorbed, and initialized as new landmarks in the estimator's map.

3 Estimation Algorithm Description

When designing the proposed IKS our objective is two-fold: (i) Process all inertial and visual observations within the current epoch of the sliding window $\mathbf{x}_{I_{k+1:k+M}}$ (i.e., inertial measurements $\{\mathbf{u}_{\ell,\ell+1}\}$, for $k+1 \leq \ell \leq k+M-1$, and feature tracks $\mathbb{Z}_{LM}, \mathbb{Z}_{LI}, \mathbb{Z}_{LN}, \mathbb{Z}_M$, and $\mathbb{Z}_I$), and (ii) Allow future epochs to re-process all measurements that are independent of the sliding window's "tail" state $\mathbf{x}_{I_{k+1}}$ (i.e., the inertial measurements $\{\mathbf{u}_{\ell,\ell+1}\}$, for $k+2 \leq \ell \leq k+M-1$, and visual observations to immature MSC-KF features and SLAM landmarks (i.e., $\mathbb{Z}_I$ and $\mathbb{Z}_{LI}$, respectively).

3.1 IKS Algorithm: Input

Before image $k+M$ arrives, the proposed IKS maintains:
(1) **A set of linearization points**, over the estimator's sliding-window of camera poses $\mathbf{x}^{\star}_{I_{k+1}:k+M-1}$, and landmarks ${}^{I_{k+1}}\boldsymbol{\ell}^{\star}_{k+M-1}$ that represent the estimator's best estimates given all measurements up to t_{k+M-1}.
(2) **A prior** comprising:
(a) The pdf of the oldest state, $\mathbf{x}_{I_{k+1}}$, within the sliding window approximated as a Gaussian $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1}}, \mathbf{P}^{\ominus}_{I_{k+1}})$.
(b) A set of N_L linearized constraints relating the oldest state, $\mathbf{x}^{\star}_{I_{k+1}}$, with the rest of the poses within the sliding window, expressed as:

$$\mathbf{r}^{\star\ominus}_L = \mathbf{H}^{\star\ominus}_L \tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M-1}} + \mathbf{n}_L, \ \mathbf{n}_L \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_{N_L}). \tag{7}$$

(c) A set of $3N_{k+M-1}$ linearized constraints relating the oldest state, $\mathbf{x}^{\star}_{I_{k+1}}$, with the rest of the poses within the sliding window and the SLAM landmarks ${}^{I_{k+1}}\boldsymbol{\ell}_{k+M-1}$ expressed as:

$$\mathbf{r}^{\star\ominus}_M = \mathbf{H}^{\star\ominus}_M \tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M-1}} + \mathbf{F}^{\star\ominus}_M {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M-1} + \mathbf{n}_M, \ \mathbf{n}_M \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_{3N_{k+M}}). \tag{8}$$

The ensemble of the pdf $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1}}, \mathbf{P}^{\ominus}_{I_{k+1}})$ and the linearized constraints $\{\mathbf{r}^{\star\ominus}_L, \ \mathbf{H}^{\star\ominus}_L\}$ and $\{\mathbf{r}^{\star\ominus}_M, \ \mathbf{H}^{\star\ominus}_M, \ \mathbf{F}^{\star\ominus}_M\}$ in (7) and (8) represent all information for the poses $\mathbf{x}_{I_{k+1:k+M-1}}$ and the landmarks ${}^{I_{k+1}}\boldsymbol{\ell}_{k+M-1}$, accumulated through absorption of past visual observations (i.e., $\mathbb{Z}_P$ in Fig. 1) and inertial measurements (i.e., $\{\mathbf{u}_{\ell,\ell+1}, \ \ell \leq k\}$).

3.2 IKS Algorithm: Overview

A single recursion of the IKS, involves the following steps:

1. Propagation: The prior pdf, $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1}}, \mathbf{P}^{\ominus}_{I_{k+1}})$, of $\mathbf{x}_{I_{k+1}}$ and the inertial measurements $\{\mathbf{u}_{\ell,\ell+1}\}$, for $k+1 \leq \ell \leq k+M-1$, are used for creating a prior $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+M}}, \mathbf{P}^{\ominus}_{I_{k+1:k+M}})$ for *all* the poses within the sliding window.

2. State Update: All available feature tracks, either from SLAM landmarks, i.e., $\mathbb{Z}_{LM}$, $\mathbb{Z}_{LN}$ and $\mathbb{Z}_{LI}$, and MSC-KF features, i.e., $\mathbb{Z}_M$ and $\mathbb{Z}_I$, as well as the prior constraints $\{\mathbf{r}^{\star\ominus}_L, \ \mathbf{H}^{\star\ominus}_L\}$ and $\{\mathbf{r}^{\star\ominus}_M, \ \mathbf{H}^{\star\ominus}_M, \ \mathbf{F}^{\star\ominus}_M\}$ are processed for updating the current state estimates $\mathbf{x}^{\star}_{I_{k+1:k+M}}$. This state-optimization can be performed iteratively.

3. Landmark Propagation: All landmarks are propagated to the next "tail" of the sliding window, $\mathbf{x}_{I_{k+2}}$, i.e., ${}^{I_{k+2}}\boldsymbol{\ell}_{k+M} = \left[{}^{I_{k+2}}\mathbf{f}_1^T \ldots {}^{I_{k+2}}\mathbf{f}_{N_{k+M}}^T\right]^T$.

4. Covariance Update: The measurements, $\mathbb{Z}_{LM}$, $\mathbb{Z}_{LN}$, $\mathbb{Z}_M$, and $\mathbf{u}_{k+1,k+2}$, which are about to be absorbed, are used to compute the posterior covariance $\mathbf{P}^{\oplus}_{I_{k+2}}$ of $\mathbf{x}_{I_{k+2}}$, which will become the new "tail" of the sliding window.

5. Construction of the Next Epoch's Prior Constraints: The prior constraints $\{\mathbf{r}^{\star\ominus}_{L},\ \mathbf{H}^{\star\ominus}_{L}\}$ and $\{\mathbf{r}^{\star\ominus}_{M},\ \mathbf{H}^{\star\ominus}_{M},\ \mathbf{F}^{\star\ominus}_{M}\}$ are updated so that they become independent of the state to be marginalized, $\mathbf{x}_{I_{k+1}}$, and instead reflect the new constraints between $\mathbf{x}_{I_{k+2}}$, $\mathbf{x}_{I_{k+3:k+M}}$, and ${}^{I_{k+2}}\boldsymbol{\ell}_{k+M}$.

3.3 IKS Algorithm: Detailed Description

In order to allow for a direct comparison with the INVF and the EKF, we follow a two-level presentation of the IKS: We first describe the effect that each step has on the cost function being minimized and then present the corresponding IKS equations. We start by stating that the IKS (iteratively) minimizes the cost function:

$$c_{k+M}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) = c_{P^{\ominus}_{I_{k+1}}} + c_u + c_L + c^{K}_{z_M} + c^{K}_{z_I} + c^{K}_{z_{LM}} + c^{K}_{z_{LI}} + c^{K}_{z_{LN}} \quad (9)$$
$$+ c_M + c^{M}_{z_{LM}} + c^{M}_{z_{LI}} + c^{M}_{z_{LN}}$$

where $c_{P^{\ominus}_{k+1}}$ corresponds to the prior pdf of the oldest state within the sliding window, $\mathbf{x}_{k+1}$, $c_u = \sum_{\ell=k+1}^{k+M-1} c_{u_{\ell,\ell+1}}$ to the inertial measurements $\mathbf{u}_{k+1:k+M}$ [see (2)], c_L to prior information about the poses $\mathbf{x}_{k+1:k+M-1}$ [see (7)], $c^{K}_{z_M}$, $c^{K}_{z_I}$, $c^{K}_{z_{LM}}$, $c^{K}_{z_{LI}}$, and $c^{K}_{z_{LN}}$ to geometric constraints between the poses from all available visual observations [see (6)], c_M to prior information about the SLAM landmarks ${}^{I_{k+1}}\boldsymbol{\ell}_{k+M-1}$, $c^{M}_{z_{LM}}$ [see (8)] and $c^{M}_{z_{LI}}$ to feature constraints between the poses *and* the SLAM landmarks [see (6)], and finally $c^{M}_{z_{LN}}$ to feature constraints for the *new* SLAM landmarks, ${}^{I_{k+1}}\boldsymbol{\ell}_{\mathcal{N}}$ [see (6)].

Hereafter, we employ the cost terms in (9) to describe the four main steps of the proposed IKS (see Sect. 3.2).

3.3.1 Prior Propagation

The prior pdf $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1}}, \mathbf{P}^{\ominus}_{I_{k+1}})$ and the inertial measurements $\mathbf{u}_{k+1:k+M}$ are used to generate a prior pdf $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+M}}, \mathbf{P}^{\ominus}_{I_{k+1:k+M}})$ over *all* the states, $\mathbf{x}_{I_{k+1:k+M}}$, within the sliding window. Through this process, the cost function in (9) takes the form:

$$c_{k+M}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) = c_{P^{\ominus}_{I_{k+1:k+M}}} + c_L + c^{K}_{z_M} + c^{K}_{z_I} + c^{K}_{z_{LM}} + c^{K}_{z_{LI}} + c^{K}_{z_{LN}} \quad (10)$$
$$+ c_M + c^{M}_{z_{LM}} + c^{M}_{z_{LI}} + c^{M}_{z_{LN}}$$

where, $c_{P^{\ominus}_{I_{k+1:k+M}}} = c_{P^{\ominus}_{I_{k+1}}} + \sum_{\ell=k+1}^{k+M-1} c_{u_{\ell,\ell+1}}$ corresponds to the prior $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+M}}$, $\mathbf{P}^{\ominus}_{I_{k+1:k+M}})$.
The mean $\hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+M}}$ is computed as:

$$\hat{\mathbf{x}}^{\ominus}_{I_{k+i}} = \begin{cases} \hat{\mathbf{x}}^{\ominus}_{I_{k+1}} & , i = 1 \\ \mathbf{f}(\mathbf{x}^{\star}_{I_{k+i-1}}, \mathbf{u}_{k+i-1,k+i}) + \boldsymbol{\Phi}^{\star}_{k+i,k+i-1}\delta\mathbf{x}^{\ominus}_{k+i-1} & , 2 \leq i \leq M \end{cases} \tag{11}$$

where $\delta\mathbf{x}^{\ominus}_{I_{k+i-1}} = \hat{\mathbf{x}}^{\ominus}_{I_{k+i-1}} - \mathbf{x}^{\star}_{I_{k+i-1}}$. Note that for the EKF inertial measurements are linearized and processed once, soon as they become available; hence, $\delta\mathbf{x}^{\ominus}_{I_{k+i-1}} = \mathbf{0}$ and the mean of the prior pdf, $\hat{\mathbf{x}}^{\ominus}_{I_{k+i}}$, coincides with the linearization point $\mathbf{x}^{\star}_{I_{k+i}}$. In contrast, the IKS re-processes inertial measurements by re-computing the prior over the sliding window $\mathbf{x}_{I_{k+1:k+M}}$ through the process described in (11).

The block elements of the covariance $\mathbf{P}^{\ominus}_{I_{k+1:k+M}}$ are computed through the EKF covariance propagation recursion, using, however, the most recent state estimates:

$$\begin{aligned} \mathbf{P}^{\ominus}_{I_{k+i}} &= \boldsymbol{\Phi}^{\star}_{k+i,k+i-1}\mathbf{P}^{\ominus}_{I_{k+i-1}}\boldsymbol{\Phi}^{\star T}_{k+i,k+i-1} + \mathbf{Q}^{\star}_{k},\ i = 2\ldots, M \\ \mathbf{P}^{\ominus}_{I_{k+i,k+j}} &= \boldsymbol{\Phi}^{\star}_{k+i,k+j}\mathbf{P}^{\ominus}_{I_{k+j}},\ i = 2, \ldots, M,\ j = 1, \ldots, i-1 \end{aligned} \tag{12}$$

3.3.2 State Update

All cost terms, which provide multi-state (i.e., c_L, $c^K_{z_M}$, $c^K_{z_I}$, $c^K_{z_{LM}}$, $c^K_{z_{LI}}$, $c^K_{z_{LN}}$), as well as mapping (i.e., c_M, $c^M_{z_{LM}}$, $c^M_{z_{LI}}$, $c^M_{z_{LN}}$) constraints, are used for updating the state estimates for $\mathbf{x}_{I_{k+1:k+M}}$ and ${}^{I_{k+1}}\boldsymbol{\ell}_{k+M}$. Although each of these terms could have been used independently, in successive updates, we choose to first merge them into two cost terms, c^K_S and c^M_S, comprising multi-state geometric and mapping constraints, respectively, and process them in a batch form.

Measurement Compression: Combining all mapping terms, $c_M, c^M_{z_{LM}}, c^M_{z_{LI}}\ c^M_{z_{LN}}$, into a single cost term, $c^{M'}_S$, yields:

$$\begin{aligned} c^{M'}_S(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+1}}\tilde{\boldsymbol{\ell}}_{k+M}) &= \left|\left| \begin{bmatrix} \mathbf{r}^{\star\ominus}_M \\ \mathbf{r}^{M\star}_{LI} \\ \mathbf{r}^{M\star}_{LM} \\ \mathbf{r}^{M\star}_{LN} \end{bmatrix} - \begin{bmatrix} \mathbf{H}^{\star\ominus}_M & \mathbf{F}^{\star\ominus}_M & \\ \mathbf{H}^{M\star}_{LI} & \mathbf{F}^{M\star}_{LI} & \\ & \mathbf{F}^{M\star}_{LM} & \\ \mathbf{H}^{M\star}_{LN} & & \mathbf{F}^{M\star}_{LN} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}} \\ {}^{I_{k+1}}\tilde{\boldsymbol{\ell}}^{\star}_{k+M-1} \\ {}^{I_{k+1}}\tilde{\boldsymbol{\ell}}_{\mathcal{N}} \end{bmatrix} \right|\right|^2_{\sigma^2\mathbf{I}} \\ &= ||\mathbf{r}^{M'}_S - [\mathbf{H}^{M'\star}_S\ \mathbf{F}^{M'\star}_S] \begin{bmatrix} \tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}} \\ {}^{I_{k+1}}\tilde{\boldsymbol{\ell}}^{\star}_{k+M} \end{bmatrix} ||^2_{\sigma^2\mathbf{I}} \end{aligned} \tag{13}$$

As in Sect. 2.2, we project the linearized constraints of (13) onto the column space and left nullspace of $\mathbf{F}^{M'\star}_S$, which partitions $c^{M'}_S$ into, $3N_{k+M}$ constraints, denoted by c^M_S, providing information *only* over the landmarks $\boldsymbol{\ell}_{k+M}$ and a cost term $c^{K'}_S$, providing geometric constraints, *only* over the poses $\mathbf{x}_{I_{k+1:k+M}}$, i.e.,

$$\begin{aligned} c_S^{M'}(\tilde{\mathbf{x}}^{\star}_{I_k+1:k+M}, {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) &= ||\mathbf{r}_S^{M\star} - \left[\mathbf{H}_S^{M\star}\ \mathbf{F}_S^{M\star}\right]\begin{bmatrix}\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}} \\ {}^{I_{k+1}}\tilde{\boldsymbol{\ell}}^{\star}_{k+M}\end{bmatrix}||^2_{\sigma^2\mathbf{I}_{3N_{k+M}}} \\ &+ ||\mathbf{r}_S^{K'\star} - \mathbf{H}_S^{K'\star}\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}||^2_{\sigma^2\mathbf{I}} \\ &= c_S^M(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) + c_S^{K'}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}) \end{aligned}$$

which allows c_{k+M} in (10) to take the form:

$$c_{k+M}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) = c_{P^{\ominus}_{I_{k+1:k+M}}} + c_S^M + c_S^K \tag{14}$$

where,[4] c_S^K comprises all geometric constraints, i.e., c_L, $c_S^{K'}$, $c^K_{z_M}$, $c^K_{z_I}$, $c^K_{z_{LM}}$, $c^K_{z_{LI}}$, and $c^K_{z_{LN}}$:

$$c_S^K(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}) = ||\mathbf{r}_s^{K\star} - \mathbf{H}_s^{K\star}\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}||^2_{\sigma^2\mathbf{I}}. \tag{15}$$

Note that, since $\mathbf{F}_S^{M\star}$ is a square full-rank matrix and for every $\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}$ there exists an ${}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}$ that minimizes c_S^M to zero, for minimizing (14), it suffices to first minimize $c^{\ominus}_{P_{I_{k+1:k+M}}} + c_S^K$, over $\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}$, and then solve for ${}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}$, using c_S^M. Specifically, through an I-EKF update step [4] we first update the poses $\mathbf{x}_{I_{k+1:k+M}}$:

$$\mathbf{x}^{\star\oplus}_{I_{k+1:k+M}} = \mathbf{x}^{\star}_{I_{k+1:k+M}} + \tilde{\mathbf{x}}^{o}_{I_{k+1:k+M}}, \quad \tilde{\mathbf{x}}^{o}_{I_{k+1:k+M}} = \delta\mathbf{x}^{\ominus}_{I_{k+1:k+M}} + \mathbf{P}^{\ominus}_{I_{k+1:k+M}}\mathbf{H}_S^{K\star T}\mathbf{d}_S \tag{16}$$

where $\mathbf{d}_S$ is the solution to the linear system $\mathbf{S}^{\star}\mathbf{d}_S = \mathbf{r}_S^{K\star} - \mathbf{H}_S^{K\star}\delta\mathbf{x}^{\ominus}_{I_{k+1:k+M}}$, with $\mathbf{S}^{\star} = \mathbf{H}_S^{K\star}\mathbf{P}^{\ominus}_{I_{k+1:k+M}}\mathbf{H}_S^{K\star T} + \sigma^2\mathbf{I}$, and $\delta\mathbf{x}^{\ominus}_{I_{k+1:k+M}} = \hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+M}} - \mathbf{x}^{\star}_{I_{k+1:k+M}}$.

Second, we substitute $\tilde{\mathbf{x}}^{o}_{I_{k+1:k+M}}$ into c_S^M, and solve[5] the linear system

$$\mathbf{F}_S^{M\star\, I_{k+1}}\widetilde{\boldsymbol{\ell}}^{o}_{k+M} = \mathbf{r}_S^{M\star} - \mathbf{H}_S^{M\star}\tilde{\mathbf{x}}^{o}_{I_{k+1:k+M}} \tag{17}$$

for updating the SLAM landmarks: ${}^{I_{k+1}}\boldsymbol{\ell}^{\star\oplus} = {}^{I_{k+1}}\boldsymbol{\ell}^{\star} + {}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{o}_{k+M}$.

It is important to note that processing immature observations (i.e., $\mathbb{Z}_I$ and $\mathbb{Z}_{LI}$) is *optional* for the IKS, allowing us to adjust its computational cost, based on the availability of computational resources.

3.3.3 Landmark Propagation

Before proceeding with computing the next epoch's prior pdf $\mathcal{N}(\hat{\mathbf{x}}^{\oplus}_{k+2}, \mathbf{P}^{\oplus}_{k+2})$ and linearized constraints, $\{\mathbf{r}_L^{\star\oplus}, \mathbf{H}_L^{\star\oplus}\}$ and $\{\mathbf{r}_M^{\star\oplus}, \mathbf{H}_M^{\star\oplus}, \mathbf{F}_M^{\star\oplus}\}$, we express all landmarks ${}^{I_{k+1}}\boldsymbol{\ell}_{k+M}$ w.r.t. the new "tail" pose of the sliding window, $\mathbf{x}_{I_{k+2}}$, in (9). As we describe in detail in [17], the landmark parameters in the two frames I_{k+1} and I_{k+2}, as

[4] For reducing the computational cost (linear in the number of MSC-KF features within the sliding window), the residual $\mathbf{r}_s^{K\star}$ and Jacobian matrix $\mathbf{H}_s^{K\star}$ are compressed using QR factorization [2].

[5] Not surprisingly, the computational cost of this step is cubic in the number of SLAM landmarks, which are observed at the present epoch, as in the corresponding EKF and INVF state-update steps.

well as the poses $\mathbf{x}_{I_{k+1:k+2}}$, are related through a non-linear *deterministic* constraint, $\mathbf{g}(\mathbf{x}_{I_{k+1:k+2}}, {}^{I_{k+1}}\boldsymbol{\ell}_{k+M}, {}^{I_{k+2}}\boldsymbol{\ell}_{k+M}) = \mathbf{0}$, which after linearization, becomes:

$$\begin{aligned} &\mathbf{G}^{\star}_{I_{k+1:k+2}}\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+2}} + \mathbf{G}^{\star}_{I_{k+1},\ell}\,{}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}_{k+M} + \mathbf{G}^{\star}_{I_{k+2},\ell}\,{}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}_{k+M} = \mathbf{0} \\ \Leftrightarrow\ &{}^{I_{k+1}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M} = -\mathbf{G}^{\star-1}_{I_{k+1},\ell}(\mathbf{G}^{\star}_{I_{k+1:k+2}}\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+2}} + \mathbf{G}^{\star}_{I_{k+2},\ell}\,{}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) \end{aligned} \tag{18}$$

Substituting in (9), transforms c_{k+M} to a function of $\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}$ and ${}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}$.

3.3.4 Covariance Update

Our objective is to compute the posterior $\mathcal{N}(\hat{\mathbf{x}}^{\oplus}_{I_{k+2}}, \mathbf{P}^{\oplus}_{I_{k+2}})$, which will be used as the prior pdf during the next epoch. To do so, we will operate on those terms of the cost function in (9) that contain the state $\mathbf{x}_{I_{k+1}}$, which is about to be marginalized; that is the cost function:

$$c^{M}_{k+M} = c_{P^{\ominus}_{I_{k+1}}} + c_{u_{k+1:k+2}} + c_L + c^{K}_{z_M} + c^{K}_{z_{LM}} + c^{K}_{z_{LN}} + c_M + c^{M}_{z_{LM}} + c^{M}_{z_{LN}} \tag{19}$$

In particular, we follow a 4-step process:
Prior propagation and measurement compression: Following the same process, as in Sects. 3.3.1 and 3.3.2, we use the prior $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1}}, \mathbf{P}^{\ominus}_{I_{k+1}})$ and the inertial measurements $\mathbf{u}_{k+1:k+2}$ to compute the prior $\mathcal{N}(\hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+2}}, \mathbf{P}^{\ominus}_{I_{k+1:k+2}})$, and merge the linearized constraints into two terms, c^{K}_{C} and c^{M}_{C}, comprising multi-state and mapping constraints, respectively. Thus, (19) becomes:

$$c^{M}_{k+M} = c_{P^{\ominus}_{I_{k+1:k+2}}} + c^{K}_{C} + c^{M}_{C} \tag{20}$$

Partitioning of the linearized constraints: Following the same process as in (6), we partition c^{K}_{C} into $c^{K}_{C_1}$ and $c^{K}_{C_2}$, where the first term depends only on $\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+2}}$, i.e., $c^{K}_{C}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}) = c^{K}_{C_1}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+2}}) + c^{K}_{C_2}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}})$, which after substitution in (20), yields:

$$c^{M}_{k+M} = c_{P^{\ominus}_{I_{k+1:k+2}}} + c^{K}_{C_1}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+2}}) + c^{K}_{C_2}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}) + c^{M}_{C}. \tag{21}$$

Covariance Update: At this point, we combine the first two terms in (21), thus updating the *prior pdf* $\mathcal{N}(\mathbf{x}^{\ominus}_{I_{k+1:k+2}}, \mathbf{P}^{\ominus}_{I_{k+1:k+2}})$:

$$c^{M}_{k+M} = c_{P^{\oplus}_{I_{k+1:k+2}}} + c^{K}_{C_2} + c^{M}_{C}. \tag{22}$$

$\mathcal{N}(\mathbf{x}^{\oplus}_{I_{k+1:k+2}}, \mathbf{P}^{\oplus}_{I_{k+1:k+2}})$ has mean $\hat{\mathbf{x}}^{\oplus}_{I_{k+1:k+2}} = \hat{\mathbf{x}}^{\ominus}_{I_{k+1:k+2}} + \mathbf{P}^{\ominus}_{I_{k+1:k+2}}\mathbf{H}^{K\star T}_{C_1}\mathbf{d}_C$, where $\mathbf{d}_C$ is the solution to the linear system $\mathbf{S}_C\mathbf{d}_C = \mathbf{r}^{K\star}_{C_1} - \mathbf{H}^{K\star T}_{C_1}\delta\mathbf{x}^{\ominus}_{k+1:k+2}$, with $\mathbf{S}_C = \mathbf{H}^{K\star}_{C_1}\mathbf{P}^{\ominus}_{I_{k+1:k+2}}$ $\mathbf{H}^{K\star T}_{C_1} + \sigma^2\mathbf{I}_{N_1}$, and covariance $\mathbf{P}^{\oplus}_{I_{k+1:k+2}} = \mathbf{P}^{\ominus}_{I_{k+1:k+2}}\left(\mathbf{I} - \mathbf{H}^{K\star T}_{C_1}\mathbf{S}^{-1}_{C}\mathbf{H}^{K\star}_{C_1}\mathbf{P}^{\ominus}_{I_{k+1:k+2}}\right)$.

3.3.5 Construction of Next Epoch's Prior

During this last step of the IKS, we will bring c^M_{k+M} into a form whose minimization is independent of $\mathbf{x}_{I_{k+1}}$. To achieve this, we follow a 2-step process.

Partitioning of $c_{P^{\oplus}_{I_{k+1:k+2}}}$: By employing the Schur complement, the prior term $c_{P^{\oplus}_{I_{k+1:k+2}}}$ in (22) is partitioned into a prior over $\mathbf{x}_{I_{k+2}}$, $c_{P^{\oplus}_{I_{k+2}}}$, and a conditional term, $c_{I_{k+1|k+2}}$, representing linearized constraints between $\mathbf{x}_{I_{k+1}}$ and $\mathbf{x}_{I_{k+2}}$, i.e.,

$$\begin{aligned} c_{P^{\oplus}_{I_{k+1:k+2}}}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+2}}) =& ||\tilde{\mathbf{x}}^{\star}_{I_{k+2}} - \delta\mathbf{x}^{\oplus}_{I_{k+2}}||_{\mathbf{P}^{\oplus}_{I_{k+2}}} + ||\delta\mathbf{x}^{\oplus}_{I_{k+1|k+2}} - \begin{bmatrix} \mathbf{I} & -\mathbf{P}^{\oplus}_{I_{k+1,k+2}}\mathbf{P}^{\oplus -1}_{I_{k+2}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}^{\star}_{I_{k+1}} \\ \tilde{\mathbf{x}}^{\star}_{I_{k+2}} \end{bmatrix} ||^2_{\mathbf{P}^{\oplus}_{I_{k+1|k+2}}} \\ =& c_{P^{\oplus}_{I_{k+2}}}(\tilde{\mathbf{x}}_{I_{k+2}}) + c_{I_{k+1|k+2}}(\tilde{\mathbf{x}}_{I_{k+1:k+2}}) \end{aligned} \tag{23}$$

where $\mathbf{P}^{\oplus}_{I_{k+1|k+2}} = \mathbf{P}^{\oplus}_{I_{k+1}} - \mathbf{P}^{\oplus}_{I_{k+1,k+2}}\mathbf{P}^{\oplus -1}_{I_{k+2}}\mathbf{P}^{\oplus}_{I_{k+2,k+1}}$. Substituting in (22), yields:

$$c^M_{k+M} = c_{P^{\oplus}_{I_{k+2}}} + c_{I_{k+1|k+2}} + c^K_{C_2} + c^M_C. \tag{24}$$

Marginalization of $\mathbf{x}_{I_{k+1}}$: Firstly, we combine all terms involving $\mathbf{x}_{I_{k+1}}$, i.e., $c_{I_{k+1|k+2}}$, $c^K_{C_2}$, and c^M_C into a single quadratic cost, corresponding to $15 + N^K_{C_2} + 3N_{k+M}$ linearized constraints:

$$c_J(\tilde{\mathbf{x}}_{I_{k+1:k+M}}, {}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) = ||\mathbf{b} - \mathbf{J}_1\tilde{\mathbf{x}}^{\star}_{I_{k+1}} - \mathbf{J}_2 \begin{bmatrix} \tilde{\mathbf{x}}^{\star}_{I_{k+2:k+M}} \\ {}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M} \end{bmatrix} ||^2_{\mathbf{I}_{15+N^K_{C_2}+3N_{k+M}}} \tag{25}$$

Following the same process as in (6), we partition c_J into $c_{I_{k+1|k+2:k+M}}$, that contains all information regarding $\mathbf{x}_{I_{k+1}}$, and $c_{L^{\oplus}}$ and $c_{M^{\oplus}}$, which are independent of $\mathbf{x}_{I_{k+1}}$:

$$c_J(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) = c_{I_{k+1|k+2:k+M}} + c_{L^{\oplus}} + c_{M^{\oplus}}. \tag{26}$$

where the detailed analytical expressions for $c_{L^{\oplus}}$ and $c_{M^{\oplus}}$ are given in [17].

Substituting (26) in (24), yields:

$$\begin{aligned} c^M_{k+M}(\tilde{\mathbf{x}}^{\star}_{I_{k+1:k+M}}, {}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) = & c_{P^{\oplus}_{I_{k+2}}}(\tilde{\mathbf{x}}^{\star}_{I_{k+2}}) + c_{L^{\oplus}}(\tilde{\mathbf{x}}^{\star}_{I_{k+2:k+M}}) + c_{M^{\oplus}}(\tilde{\mathbf{x}}^{\star}_{I_{k+2:k+M}}, {}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}) \\ & + c_{I_{k+1|k+2:k+M}}(\tilde{\mathbf{x}}^{\star}_{I_{k+2:k+M}}, {}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}). \end{aligned} \tag{27}$$

The last term, $c_{I_{k+1|k+2:k+M}}$ in (27), is irrelevant for the minimization of c^M_{k+M} over $\tilde{\mathbf{x}}^{\star}_{I_{k+2:k+M}}$ and ${}^{I_{k+2}}\widetilde{\boldsymbol{\ell}}^{\star}_{k+M}$ since, for any of their values, there exists a $\tilde{\mathbf{x}}^{o}_{I_{k+1}}$ that minimizes $c_{I_{k+1|k+2:k+M}}$ to *exactly* zero. Hence, all prior information from the current to the next IKS recursion, is represented completely through the terms $c_{P^{\oplus}_{k+2}}$, $c_{L^{\oplus}}$, and $c_{M^{\oplus}}$ all of which do *not* involve $\tilde{\mathbf{x}}^{\star}_{I_{k+1}}$.

4 Simulations

Our simulations involved a MEMS-quality commercial grade IMU, similar to those present on current mobile devices, running at 100 Hz, and a wide (175° degrees) field of view camera with resolution 640 × 480. Visual observations were contaminated by zero-mean white Gaussian noise with $\sigma = 1.0$ pixel. We compared the Root Mean Square Error (RMSE) for the real-time estimates[6] of the MSC-KF VINS (denoted as EKF), with that of the proposed iterative Kalman smoother (denoted as IKS). Both estimators maintained a sliding window of 10 camera poses. Feature tracks that spanned beyond the sliding window were initialized as SLAM landmarks (and marginalized when lost from the camera's field of view), while shorter feature tracks were processed as MSC-KF features.

- VINS under nominal conditions: In Simulation I (see Fig. 2), the platform's trajectory and dynamics resembled those of a person traversing 120 m of an indoor environment, while new camera poses were generated every 25 cm, and the rate that new features enter the camera's field of view followed that of real-world experimental trials. As seen in Fig. 2a, the performance difference between the EKF-based VINS and the proposed IKS is rather small, since in the presence of many visual measurements, both estimators are able to accurately track the system's state. Note however, that even under these nominal conditions the IKS always maintained a more accurate estimate of the platform's speed (see Fig. 2b), while for certain parts of the trajectory its estimate improved by ∼20%, over the EKF, due to the inability of the latter to process feature tracks immediately as they become available.

- Camera Occlusions: In Simulation II (see Fig. 3), we simulated the motion of a handheld device, "hovering" over the same scene, for 40 s, emulating a common scenario for augmented-reality applications. We then introduced 3 periods, of approximately 5 s each, during which the camera was occluded and no feature tracks were available. As evident from Fig. 3a, b, by re-processing visual and inertial measurements, the IKS, converges faster to the correct position and velocity estimates, right after re-gaining access to camera measurements.

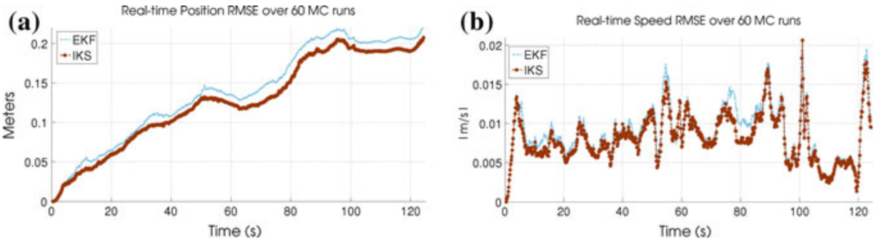


Fig. 2 Monte carlo simulation I: Comparison of the proposed IKS versus the EKF under nominal conditions: **a** Position RMSE; **b** Speed RMSE

[6] By real-time, we refer to the estimate for $\mathbf{x}_{I_k}$, right before, processing image k.

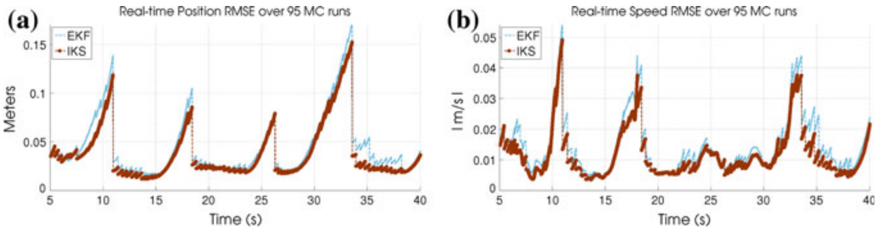


Fig. 3 Monte carlo simulation II: Comparison of the proposed IKS versus the EKF during camera occlusions: **a** Position RMSE; **b** Speed RMSE

5 Experiments

We further validated the proposed IKS on real-world data, using a Project Tango developer tablet, and as ground truth the result of a batch least-squares (BLS) over the entire set of visual and inertial measurements. As in Sect. 4, we compared the real-time estimates of the proposed IKS, to those of the MSC-KF VINS, both of which processed measurements to SLAM landmarks, as well as MSC-KF feature tracks, and maintained a sliding window of length 14.

Initialization: In Experiment I (Fig. 4), we compared the initialization phase, of both estimators, when they start from inaccurate estimates of their initial velocity and IMU biases, as often happens in practice. As it is seen in Fig. 4b, the IKS converged faster to its correct velocity estimates, which lead to a reduced position error, as compared to the EKF, for the rest of their trajectories (Fig. 4a). Note that a BLS over a small set of initial poses, could have been used for system initialization, making both estimators equally accurate. Such a choice, however, would inevitably introduce a significant time delay, a key drawback for real-time applications, while the complexity of the corresponding implementation would significantly increase. For the IKS, however, an iterative optimization during its initial phase, seamlessly takes place, without the need to transition from BLS to filtering.

Fast camera turns: In Experiment II (Fig. 5), we collected a "stress" dataset, during which the camera performed quick turns, inside an office area, causing abrupt reductions in the number, quality, and length of feature tracks for short periods of time (Fig. 5a). As it is evident from Fig. 5d, the inability of the EKF to re-process visual

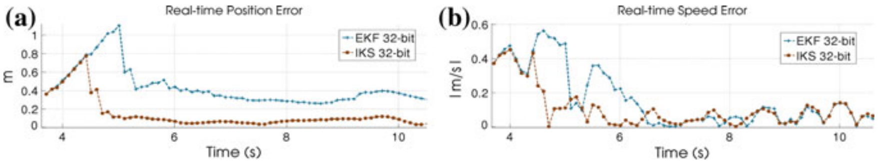


Fig. 4 Experiment I: Comparison of the proposed IKS versus the EKF during filter initialization: **a** Real-time position error; **b** Real-time speed error

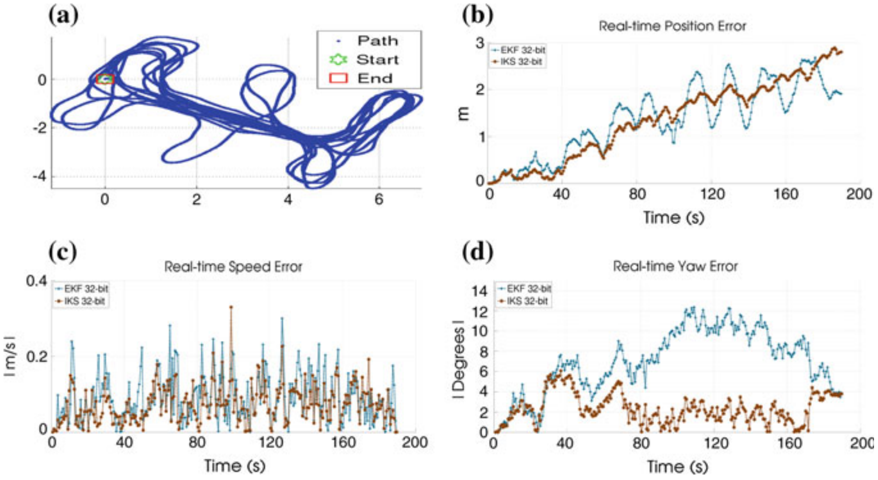


Fig. 5 Experiment II: Comparison of the proposed IKS versus the EKF during sudden turns: **a** 3D Trajectory; **b** Position error; **c** Speed error; **d** Yaw error

observations, caused sudden accumulations of yaw error, e.g., at 40 s. Furthermore, on par with our simulation results, the IKS maintained an improved real-time estimate, of the platform's velocity, throughout the experiment, while at certain points, its velocity estimate was even 2 times better than the EKF's (Fig. 5c).

Timing analysis: We used the Samsung S4 cell phone, as a testbed for comparing the processing time of the proposed IKS, *with and without processing of immature visual observations*, denoted by IKS w/, and IKS w/o, respectively, as well as, a reference EKF implementation. Albeit the 32-bit arithmetic precision, of the NEON co-processor, present on the 1.6 GHz Cortex-A15 ARM CPU of the Samsung S4, no numerical inaccuracies were introduced, when compared to 64-bit arithmetic precision. All estimators maintained a sliding window of 14 poses and on average 5 SLAM landmarks in their state vector. As it is evident from Table 1, the proposed IKS achieves real-time performance, even under re-linearization, while it is able to bring its cost down to levels comparable to the EKF by temporary disabling the re-processing of visual observations.

6 Conclusion

In this paper, we have presented an iterative Kalman smoother (IKS) for vision-aided inertial navigation that incorporates advantages of competing approaches. Through smoothing, the proposed IKS iteratively re-linearizes both inertial and visual measurements over a single, or multiple overlapping, sliding windows, thus improving robustness. At the same time, the IKS inherits the excellent numerical properties

Table 1 Timing analysis on the samsung S4: numbers denote average time in ms

Step\Algorithm	IKS w/	IKS w/o	EKF
Propagation	14	14	1
Jacobians calculation	101	12	9
Measurement compressions	14	6	2
State update	12	12	12
Covariance update	0.5	0.5	18
Prior constraints update	1.5	1.5	N/A
Total	166	46	42

of the Kalman filter, making it amenable to very efficient implementations (4-fold speed up on ARM NEON co-processor) using single-precision (32 bit) arithmetic. As part of our validation process, we demonstrated the resilience and efficiency of the proposed approach, under adverse navigation conditions.

Acknowledgements This work was supported by the National Science Foundation (IIS-1328722) and UMN's Doctoral Dissertation Fellowship.

References

1. Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I.: Consistency analysis and improvement of vision-aided inertial navigation. IEEE Trans. Robot. **30**, 158–176 (2014)
2. Mourikis, A.I., Trawny, N., Roumeliotis, S.I., Johson, A.E., Ansar, A., Matthies, L.: Vision-aided inertial navigation for spacecraft entry, descent, and landing. IEEE Trans. Robotics **25**, 264–280 (2009)
3. Nerurkar, E.D., Wu, K.J., Roumeliotis, S.I.: C-KLAM: constrained keyframe localization and mapping for long-term navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3638–3643. Hong Kong, China, May 31–June 6 (2013)
4. Jazwinski, A.: Mathematics in science and engineering. Stochastic Processes and Filtering Theory, vol. 64, Academic Press, New York (1970)
5. Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I.: Towards consistent vision-aided inertial navigation. In: Proceedings of the 10th International Workshop on the Algorithmic Foundations of Robotics (WAFR'12), pp. 559–574. Cambridge, Massachusetts, 13–15 June 2012
6. Guo, C., Kottas, D., DuToit, R., Ahmed, A., Li, R., Roumeliotis, S.: Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps. In: Proceedings of Robotics: Science and Systems, Berkeley, USA (2014)
7. Sibley, G., Matthies, L., Sukhatme, G.: Sliding window filter with application to planetary landing. J. F. Robot. **27**(5), 587–608 (2010)
8. Huang, G.P., Mourikis, A.I., Roumeliotis, S.I.: An observability-constrained sliding window filter for SLAM. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 65–72. San Francisco, CA, 25–30 Sept 2011

9. Chiu, H.-P., Williams, S., Dellaert, F., Samarasekera, S., Kumar, R.: Robust vision-aided navigation using sliding-window factor graphs. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 46–53. Karlsruhe, Germany, 6–10 May 2013
10. Leutenegger, S., Furgale, P.T., Rabaud, V., Chli, M., Konolige, K., Siegwart, R.: Keyframe-based visual-inertial SLAM using nonlinear optimization. In: Proceedings of Robotics: Science and Systems, Berlin, Germany, June 2013
11. Wu, K.J., Medhat, A., Georgiou, G., Roumeliotis, S.I.: A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In: Proceedings of the Robotics: Science and Systems, Rome, Italy, 13–17 July 2015
12. Kottas, D.G., Roumeliotis, S.I.: An iterative kalman smoother for robust 3D localization on mobile and wearable devices. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 6336–6343. Seattle, WA, 26–30 May 2015
13. Mirzaei, F.M., Roumeliotis, S.I.: A kalman filter-based algorithm for imu-camera calibration: observability analysis and performance evaluation. IEEE Trans. Robot. **24**, 1143–1156 (2008)
14. Trawny, N., Roumeliotis, S.I.: Indirect kalman filter for 3D attitude estimation. Technical Report, University of Minnesota, Deptartment of Computer Science and Engineering, March 2005
15. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the Alvey Vision Conference, pp. 147–151. Manchester, UK, Aug 31–Sept 2 1988
16. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificaial Intelligence, pp. 674–679. Vancouver, British Columbia, 24–28 Aug 1981
17. Kottas, D.G., Roumeliotis, S.I.: An iterative kalman smoother for robust 3D localization and mapping, http://www-users.cs.umn.edu/~dkottas/pdfs/iks_slam_2014_tr.pdf. Technical Report, University of Minnesota, Department of Computer Science and Engineering, MARS Lab, Apr 2015

Deterministic Sampling-Based Motion Planning: Optimality, Complexity, and Performance

Lucas Janson, Brian Ichter and Marco Pavone

1 Introduction

Probabilistic sampling-based algorithms represent a particularly successful approach to robotic motion planning problems [14, 26]. The key idea behind probabilistic sampling-based algorithms is to avoid the explicit construction of the configuration space (which can be prohibitive in complex planning problems) and instead conduct a search that probabilistically probes the configuration space with independent and identically distributed (i.i.d.) random samples. Explicitly, each i.i.d. point is generated in the same way (e.g., from a uniform distribution over the configuration space) and without using any information about any of the other sampled points. This probing is enabled by a collision detection module, which the motion planning algorithm considers as a "black box" [14]. Examples, roughly in chronological order, include the probabilistic roadmap algorithm (PRM) [13], expansive space trees (EST) [9, 20], Lazy-PRM [4], the rapidly exploring random trees algorithm (RRT) [15], sampling-based roadmap of trees (SRT) [22], rapidly-exploring roadmap [1], PRM*and RRT*[12], RRT$^\#$[2], and the fast marching tree algorithm (FMT*) [11]. A

This work was supported by NASA under the Space Technology Research Grants Program, Grant NNX12AQ43G. Lucas Janson was partially supported by NIH training grant T32GM096982. Brian Ichter was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

L. Janson (✉)
Department of Statistics, Stanford University, Stanford, CA 94305, USA
e-mail: ljanson@stanford.edu

B. Ichter · M. Pavone
Department of Aeronautics and Astronautics, Stanford University,
Stanford, CA 94305, USA
e-mail: ichter@stanford.edu

M. Pavone
e-mail: pavone@stanford.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_29

central result is that these algorithms provide *probabilistic completeness* guarantees in the sense that the probability that the planner fails to return a solution, if one exists, decays to zero as the number of samples approaches infinity [3]. Recently, it has been proven that RRT*, PRM*, RRT#, and FMT*are asymptotically optimal, i.e., the cost of the returned solution converges almost surely to the optimum [2, 11, 12].

It is natural to wonder whether the theoretical guarantees and practical performance of sampling-based algorithms would hold if these algorithms were to be de-randomized, i.e., run on a *deterministic*, as opposed to random sampling sequence. This is an important question, as de-randomized planners would significantly simplify the certification process (as needed for safety-critical applications), enable the use of offline computation (particularly important for planning under differential constraints or in high dimensional spaces—exactly the regime for which sampling-based planners are designed), and, in the case of lattice sequences, drastically simplify a number of operations (e.g., locating nearby samples). This question has received relatively little attention in the literature. Specifically, previous research [5, 10, 16] has focused on the performance of de-randomized versions of sampling-based planners in terms of convergence to feasible paths. A number of deterministic variants of the PRM algorithm were shown to be resolution complete (i.e., provably converging to a feasible solution as $n \to \infty$) and, perhaps surprisingly, offer superior performance on an extensive set of numerical experiments [5, 16]. Prompted by these results, a number of deterministic low-dispersion, incremental sequences have been designed specifically tailored to motion planning problems [17, 27, 28].

The results in [5, 10, 16] are restricted to convergence to feasible, as opposed to *optimal* paths. Several questions are still open. Are there advantages of i.i.d. sampling in terms of convergence to an optimal path? Can convergence rate guarantees for the case of deterministic sampling be provided, similar to what is done for probabilistic planners in [7, 11]? For a given number of samples, are there advantages in terms of computational and space complexity? The objective of this paper is to rigorously address these questions. Our focus is on the PRM algorithm. However, our results extend to most of the existing batch (i.e., not anytime) algorithms, including Lazy-PRM and FMT*. We summarize the paper's contributions below.

Deterministic asymptotic optimality of sampling-based planning: We show that the PRM algorithm is asymptotically optimal when run on *deterministic* sampling sequences whose ℓ_2-dispersion is upper-bounded by $\gamma\, n^{-1/d}$, for some $\gamma \in \mathbb{R}_{>0}$ (we refer to such sequences as deterministic low-dispersion sequences), and with a connection radius $r_n \in \omega(n^{-1/d})$.[1] In other words, the cost of the solution computed over n samples converges deterministically to the optimum as $n \to \infty$. As a comparison, the analogous result for the case of i.i.d. random sampling holds almost surely [11, 12] (as opposed to deterministically) and requires a connection radius $\Omega\left((\log(n)/n)^{1/d}\right)$, i.e., bigger.

[1] For $f, g : \mathbb{N} \to \mathbb{R}$, we say $f \in O(g)$ if there exists $n_0 \in \mathbb{N}$ and $k \in \mathbb{R}_{>0}$ such that $|f(n)| \leq k\,|g(n)|$ for all $n \geq n_0$. We say $f \in \Omega(g)$ if there exists $n_0 \in \mathbb{N}$ and $k \in \mathbb{R}_{>0}$ such that $|f(n)| \geq k\,|g(n)|$ for all $n \geq n_0$. Finally, we say $f \in \omega(g)$ if $\lim_{n\to\infty} f(n)/g(n) = \infty$.

Convergence rate: We show that, in the absence of obstacles, the factor of suboptimality of PRM is upper-bounded by $2D_n/(r_n - 2D_n)$, where D_n is the ℓ_2-dispersion of the sampling sequence. A slightly more sophisticated result holds for the obstacle-cluttered case. As a comparison, the analogous result for the case of i.i.d. sampling only holds in probability and is much more involved (and less interpretable) [11]. Such results could be instrumental to the certification (i.e., approval by regulatory agencies) of sampling-based planners.

Computational and space complexity: We prove that PRM, when run on a low-dispersion lattice, has computational and space complexity $O(n^2 r_n^d)$. As asymptotic optimality can be obtained using $r_n \in \omega(n^{-1/d})$, there exists an asymptotically optimal version of PRM with computational and space complexity $\omega(n)$, where $O(n)$ represents the theoretical lower bound (as, at the very least, n operations need to be carried out to load samples into memory). As a comparison, the analogous complexity results for the case of i.i.d. sampling are of order $O(n \log(n))$ [12]. We also extend our complexity analysis to non-lattice deterministic sampling sequences.

Experimental performance: Finally, we compare performance (in terms of path cost) of deterministic low-dispersion sampling versus i.i.d. sampling on a variety of test cases ranging from two to eight dimensions and from simple sets of rectangular obstacles to rather complicated mazes. In all our examples, for a given number of samples, deterministic low-dispersion sampling performs no worse and sometimes substantially better than i.i.d. sampling.

The key insight behind our theoretical results (e.g., smaller required connection radius, better complexity, etc.) is the factor difference in dispersion between deterministic low-dispersion sequences versus i.i.d. sequences, namely $O(n^{-1/d})$ versus $O((\log n)^{1/d}\, n^{-1/d})$ [6, 18]. Interestingly, the same $O(n^{-1/d})$ dispersion can be achieved with *non-i.i.d. random* sequences, e.g., randomly rotated and offset lattices. As we will show, these sequences enjoy the same *deterministic* performance guarantees of deterministic low-dispersion sequences and retain many of the benefits of deterministic sampling (e.g., fast nearest-neighbor indexing). Additionally, their "controlled" randomness may allow them to address some potential issues with deterministic sequences (in particular lattices), e.g., avoiding axis-alignment issues in which entire rows of samples may become infeasible due to alignment along an obstacle boundary. In this perspective, achieving deterministic guarantees is really a matter of i.i.d. sampling versus non-i.i.d., low-dispersion sampling (with deterministic sampling as a prominent case), as opposed to random versus deterministic. Collectively, our results, complementing and corroborating those in [5, 16], strongly suggest that both the study and application of sampling-based algorithms should adopt non-i.i.d. low-dispersion sampling. From a different viewpoint, our results provide a theoretical bridge between sampling-based algorithms with i.i.d. sampling and non-sampling-based algorithms on regular grids (e.g., D* [23] and related kinodynamic variants [21]).

Organization: This paper is structured as follows. In Sect. 2 we provide a review of known concepts from low-dispersion sampling, with a focus on ℓ_2-dispersion. In

Sect. 3 we formally define the optimal path planning problem. In Sect. 4 we present our three main theoretical results for planning with low-dispersion sequences: asymptotic optimality, convergence rate, and computational and space complexity. In Sect. 5 we present results from numerical experiments supporting our statements. Finally, in Sect. 6, we draw some conclusions and discuss directions for future work.

2 Background

A key characteristic of any set of points on a finite domain is its ℓ_2-dispersion. This concept will be particularly useful in elucidating the advantages of deterministic sampling over i.i.d. sampling. As such, in this section we review some relevant properties and results on the ℓ_2-dispersion.

Definition 1 (*ℓ_2-dispersion*) For a finite, nonempty set S of points contained in a d-dimensional compact Euclidean subspace $\mathscr{X}$ with positive Lebesgue measure, its ℓ_2-dispersion $D(S)$ is defined as

$$D(S) := \sup_{x \in \mathscr{X}} \min_{s \in S} \|s - x\|_2 = \sup\{r > 0 : \exists x \in \mathscr{X} \text{ with } B(x, r) \cap S = \emptyset\}, \quad (1)$$

where $B(x, r)$ is the *open* ball of radius r centered at x.

Intuitively, the ℓ_2-dispersion quantifies how well a space is covered by a set of points S in terms of the largest open Euclidean ball that touches none of the points. The quantity $D(S)$ is important in the analysis of path optimality as an optimal path may pass through an empty ball of radius $D(S)$. Hence, $D(S)$ bounds how closely any path tracing through points in S can possibly approximate that optimal path.

The ℓ_2-dispersion of a set of deterministic or random points is often hard to compute, but luckily it can be bounded by the more-analytically-tractable ℓ_∞-dispersion. The ℓ_∞-dispersion is defined by simply replacing the ℓ_2-norm in Eq. (1) by the ℓ_∞-norm, or max-norm. The ℓ_∞-dispersion of a set S, which we will denote by $D_\infty(S)$, is related to the ℓ_2-dispersion in d dimensions by [18],

$$D_\infty(S) \leq D(S) \leq \sqrt{d} D_\infty(S),$$

which allows us to bound $D(S)$ when $D_\infty(S)$ is easier to compute. In particular, an important result due to [6] is that the ℓ_∞-dispersion of n independent uniformly sampled points on $[0, 1]^d$ is $O((\log(n)/n)^{1/d})$ with probability 1. Corollary to this is that the ℓ_2-dispersion is also $O((\log(n)/n)^{1/d})$ with probability 1.

Remarkably, there are deterministic sequences with ℓ_2-dispersions of order $O(n^{-1/d})$, an improvement by a factor $\log(n)^{1/d}$. For instance, the Sukharev sequence [25], whereby $[0, 1]^d$ is gridded into $n = k^d$ hypercubes and their centers are taken as the sampled points, can easily be shown to have ℓ_2-dispersion of $(\sqrt{d}/2)\, n^{-1/d}$ for $n = k^d$ points. As we will see in Sect. 4, the use of sample sequences with lower

ℓ_2-dispersions confers on PRM a number of beneficial properties, thus justifying the use of certain deterministic sequences instead of i.i.d. ones. In the remainder of the paper, we will refer to sequences with ℓ_2-dispersion of order $O(n^{-1/d})$ as *low-dispersion* sequences. A natural question to ask is whether we can use a sequence that *minimizes* the ℓ_2-dispersion. Unfortunately, such an optimal sequence is only known for $d = 2$, in which case it is represented by the centers of the equilateral triangle tiling [14]. In this paper, we will focus on the Sukharev [25] and Halton sequences [8], except in two dimensions when we will consider the triangular lattice as well, but there are many other deterministic sequences with ℓ_2-dispersion of order $O(n^{-1/d})$; see [17, 27, 28] for other examples.

3 Problem Statement

The problem formulation follows very closely the problem formulation in [11]. Let $\mathscr{X} = [0, 1]^d$ be the configuration space, where $d \in \mathbb{N}$. Let $\mathscr{X}_{\text{obs}}$ be a closed set representing the obstacles, and let $\mathscr{X}_{\text{free}} = \text{cl}(\mathscr{X} \setminus \mathscr{X}_{\text{obs}})$ be the obstacle-free space, where $\text{cl}(\cdot)$ denotes the closure of a set. The initial condition is $x_{\text{init}} \in \mathscr{X}_{\text{free}}$, and the goal region is $\mathscr{X}_{\text{goal}} \subset \mathscr{X}_{\text{free}}$. A specific path planning problem is characterized by a triplet $(\mathscr{X}_{\text{free}}, x_{\text{init}}, \mathscr{X}_{\text{goal}})$. A function $\sigma : [0, 1] \to \mathbb{R}^d$ is a *path* if it is continuous and has bounded variation. If $\sigma(\tau) \in \mathscr{X}_{\text{free}}$ for all $\tau \in [0, 1]$, σ is said to be *collision-free*. Finally, if σ is collision-free, $\sigma(0) = x_{\text{init}}$, and $\sigma(1) \in \text{cl}(\mathscr{X}_{\text{goal}})$, then σ is said to be a *feasible path* for the planning problem $(\mathscr{X}_{\text{free}}, x_{\text{init}}, \mathscr{X}_{\text{goal}})$.

The goal region $\mathscr{X}_{\text{goal}}$ is said to be *regular* if there exists $\xi > 0$ such that $\forall y \in \partial\mathscr{X}_{\text{goal}}$, there exists $z \in \mathscr{X}_{\text{goal}}$ with $B(z; \xi) \subseteq \mathscr{X}_{\text{goal}}$ and $y \in \partial B(z; \xi)$ (the notation $\partial\mathscr{X}$ denotes the boundary of set $\mathscr{X}$). Intuitively, a regular goal region is a smooth set with a boundary that has bounded curvature. Furthermore, we will say $\mathscr{X}_{\text{goal}}$ is ξ-regular if $\mathscr{X}_{\text{goal}}$ is regular for the parameter ξ. Such regularity is required to quantify the probability of sampling a point in $\mathscr{X}_{\text{goal}}$ near its boundary, and is a technical condition as most common non-smooth sets can be arbitrarily approximated by smooth ones. Denote the set of all paths by Σ. A cost function for the planning problem $(\mathscr{X}_{\text{free}}, x_{\text{init}}, \mathscr{X}_{\text{goal}})$ is a function $c : \Sigma \to \mathbb{R}_{\geq 0}$; in this paper we will focus on the *arc length* function. The optimal path planning problem is then defined as follows:

> **Optimal path planning problem**: Given a path planning problem $(\mathscr{X}_{\text{free}}, x_{\text{init}}, \mathscr{X}_{\text{goal}})$ with a regular goal region and the arc length function $c : \Sigma \to \mathbb{R}_{\geq 0}$, find a feasible path σ^* such that $c(\sigma^*) = \min\{c(\sigma) : \sigma \text{ is feasible}\}$. If no such path exists, report failure.

A path planning problem can be arbitrarily difficult if the solution traces through a narrow corridor, which motivates the standard notion of path clearance [12]. For a given $\delta > 0$, define the δ-interior of $\mathscr{X}_{\text{free}}$ as the set of all configurations that are at least a distance δ from $\mathscr{X}_{\text{obs}}$. Then a path is said to have strong δ-clearance if it lies entirely inside the δ-interior of $\mathscr{X}_{\text{free}}$. Further, a path planning problem with optimal path cost c^* is called δ-robustly feasible if there exists a strictly positive sequence $\delta_n \to 0$, and a sequence $\{\sigma_n\}_{i=1}^{n}$ of feasible paths such that $\lim_{n\to\infty} c(\sigma_n) = c^*$ and for all $n \in \mathbb{N}$, σ_n has strong δ_n-clearance, $\sigma_n(1) \in \partial \mathcal{X}_{\text{goal}}$, and $\sigma_n(\tau) \notin \mathcal{X}_{\text{goal}}$ for all $\tau \in (0, 1)$.

Lastly, in this paper we will be considering a generic form of the PRM algorithm. That is, denote by gPRM (for generic PRM) the algorithm given by Algorithm 1. The function `SampleFree`(n) is a function that returns a set of $n \in \mathbb{N}$ points in $\mathcal{X}_{\text{free}}$. Given a set of samples V, a sample $v \in V$, and a positive number r, `Near`(V, v, r) is a function that returns the set of samples $\{u \in V : \|u - v\|_2 < r\}$. Given two samples $u, v \in V$, `CollisionFree`(u, v) denotes the boolean function which is true if and only if the line joining u and v does not intersect an obstacle. Given a graph (V, E), where the node set V contains x_{init} and E is the edge set, `ShortestPath`(x_{init}, V, E) is a function returning a shortest path from x_{init} to $\mathcal{X}_{\text{goal}}$ in the graph (V, E) (if one exists, otherwise it reports failure). Deliberately, we do not specify the definition of `SampleFree` and have left r_n unspecified, thus allowing for any sequence of points—deterministic or random—to be used, with any connection radius. These "tuning" choices will be studied in Sect. 4. We want to clarify that we are in no way proposing a new algorithm, but just defining an umbrella term for the PRM class of algorithms which includes, for instance, sPRM and PRM*as defined in [12].

Algorithm 1 gPRM Algorithm

```
$V \leftarrow \{x_{\text{init}}\} \cup$ SampleFree$(n)$; $E \leftarrow \emptyset$
for all $v \in V$ do
  $X_{\text{near}} \leftarrow$ Near$(V \backslash \{v\}, v, r_n)$
  for $x \in X_{\text{near}}$ do
    if CollisionFree$(v, x)$ then
      $E \leftarrow E \cup \{(v, x)\} \cup \{(x, v)\}$
    end if
  end for
end for
return ShortestPath$(x_{\text{init}}, V, E)$
```

4 Theoretical Results

In this section we present our main theoretical results. We begin by proving that gPRM on low-dispersion sequences is asymptotically optimal, in the deterministic sense, for connection radius $r_n \in \omega(n^{-1/d})$. Previous work has required r_n to be at least $\Omega((\log(n)/n)^{1/d})$ for asymptotic optimality.

Theorem 1 (Asymptotic optimality with deterministic sampling) *Let $(\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$ be a δ-robustly feasible path planning problem in d dimensions, with $\delta > 0$ and $\mathcal{X}_{goal}$ ξ-regular. Let c^* denote the arc length of an optimal path σ^*, and let c_n denote the arc length of the path returned by gPRM (or ∞ if gPRM returns failure) with n samples whose ℓ_2-dispersion is $D(V)$ using a radius r_n. Then if $D(V) \leq \gamma n^{-1/d}$ for some $\gamma \in \mathbb{R}$ and*

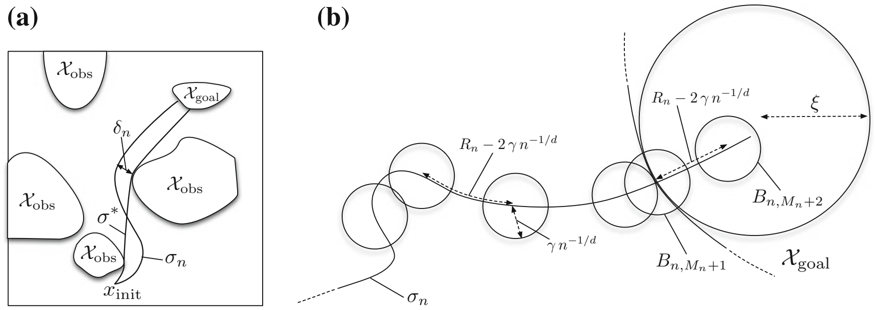


Fig. 1 **a** Illustration in 2D of σ_n as the shortest strongly δ_n-robust feasible path, as compared to the optimal path σ^*, as used in the proof of Theorem 1. **b** Illustration in 2D of the construction of $B_1, \ldots, B_{M_n+2}$ in the proof of Theorem 1

$$n^{1/d} r_n \to \infty, \tag{2}$$

then $\lim_{n\to\infty} c_n = c^*$.

Proof Fix $\varepsilon > 0$. By the δ-robust feasibility of the problem, there exists a σ_ε such that $c(\sigma_\varepsilon) \leq (1 + \varepsilon/3)c^*$ and σ_ε has strong δ_ε-clearance for some $\delta_\varepsilon > 0$, see Fig. 1a. Let R_n be a sequence such that $R_n \leq r_n$, $n^{1/d} R_n \to \infty$, and $R_n \to 0$, guaranteeing that there exists a $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$(4 + 6/\varepsilon)\gamma n^{-1/d} \leq R_n \leq \min\{\delta_\varepsilon, \xi, c^*\varepsilon/6\}. \tag{3}$$

For any $n \geq n_0$, construct the closed balls $B_{n,m}$ such that $B_{n,i}$ has radius $\gamma n^{-1/d}$ and has center given by tracing a distance $(R_n - 2\gamma n^{-1/d})i$ from x_0 along σ_ε (this distance is positive by Eq. (3)) until $(R_n - 2\gamma n^{-1/d})i > c(\sigma_\varepsilon)$. This will generate $M_n = \lfloor c(\sigma_\varepsilon)/(R_n - 2\gamma n^{-1/d})\rfloor$ balls. Define B_{n,M_n+1} to also have radius $\gamma n^{-1/d}$ but center given by the point where σ_ε meets $\mathscr{X}_{\text{goal}}$. Finally, define B_{n,M_n+2} to have radius $\gamma n^{-1/d}$ and center defined by extending the center of B_{n,M_n+1} into $\mathscr{X}_{\text{goal}}$ by a distance $R_n - 2\gamma n^{-1/d}$ in the direction perpendicular to $\partial\mathscr{X}_{\text{goal}}$. Note that by Eq. (3), $B_{n,M_n+2} \subset \mathscr{X}_{\text{goal}}$. See Fig. 1b for an illustration.

Since the dispersion matches the radii of all the $B_{n,m}$, each $B_{n,m}$ has at least one sampled point within it. Label these points $x_1, \ldots, x_{M_n+2}$, with the subscripts matching their respective balls of containment. For notational convenience, define $x_0 := x_{\text{init}}$. Note that by construction of the balls, for $i \in \{0, \ldots, M_n + 1\}$, each pair of consecutively indexed points (x_i, x_{i+1}) is separated by no more than $R_n \leq r_n$. Furthermore, since $R_n \leq \delta_\varepsilon$ by Eq. (3) above, there cannot be an obstacle between any such pair, and thus each pair constitutes an edge in the gPRM graph. Thus, we can upper-bound the cost c_n of the gPRM solution by the sum of the lengths of the edges $(x_0, x_1), \ldots, (x_{M_n+1}, x_{M_n+2})$:

$$
\begin{aligned}
c_n &\leq \sum_{i=0}^{M_n+1} \|x_{i+1} - x_i\| \leq (M_n + 2)R_n \leq \frac{c(\sigma_\varepsilon)}{R_n - 2\gamma n^{-1/d}} R_n + 2R_n \\
&\leq c(\sigma_\varepsilon) + \frac{2\gamma n^{-1/d}}{R_n - 2\gamma n^{-1/d}} c(\sigma_\varepsilon) + 2R_n = c(\sigma_\varepsilon) + \frac{1}{\frac{R_n}{2\gamma n^{-1/d}} - 1} c(\sigma_\varepsilon) + 2R_n \\
&\leq (1 + \frac{\varepsilon}{3})c^* + \frac{1}{\frac{3}{\varepsilon} + 1}(1 + \frac{\varepsilon}{3})c^* + \frac{\varepsilon}{3}c^* = (1 + \varepsilon)c^*.
\end{aligned}
$$

The second inequality follows from the fact that the distance between x_i and x_{i+1} is upper-bounded by the distance between the centers of $B_{n,i}$ and $B_{n,i+1}$ (which is at most $R_n - 2\gamma n^{-1/d}$) plus the sum of their radii (which is $2\gamma n^{-1/d}$). The last inequality follows from the facts that $c(\sigma_\varepsilon) \leq (1 + \varepsilon/3)c^*$ and Eq. (3). □

Note that if gPRM using $r_n > 2D(V)$ reports failure, then there are two possibilities: (i) a solution does not exist, or (ii) all solution paths go through corridors whose widths are smaller than 2 $D(V)$. Such a result can be quite useful in practice, as solutions going through narrow corridors could be undesirable anyways (see [16, Sect. 5] for the same conclusion).

Next, we relate the solution cost returned by gPRM to the best cost of a path with strong δ-clearance in terms of the ℓ_2-dispersion of the samples used. This is a generalization of previous convergence rates, e.g. [11], which only apply to obstacle-free spaces. Previous work also defined convergence rate as, for a fixed level of suboptimality ε, the rate (in n) that the probability of returning a greater-than-ε-suboptimal solution goes to zero. In contrast, we compute the rate (in n) that solution suboptimality approaches zero. Lastly, previous work focused on asymptotic rates in big-O notation, while here we provide exact upper-bounds for finite samples.

Theorem 2 (Convergence rate in terms of dispersion) *Consider the simplified problem of finding the shortest feasible path between two points x_0 and x_f in $\mathcal{X}_{free}$, assuming that both the initial point and final point have already been sampled. Define*

$$\delta_{max} = \sup\{\delta > 0 : \exists \textit{ a feasible } \sigma \in \Sigma \textit{ with strong } \delta\textit{-clearance}\},$$

and assume $\delta_{max} > 0$. For all $\delta < \delta_{max}$, let $c^{(\delta)}$ be the cost of the shortest path with strong δ-clearance. Let c_n be the length of the path returned by running gPRM on n points whose ℓ_2-dispersion is $D(V)$ (which we will abbreviate to D_n for convenience) and using a connection radius r_n. Then for all n such that $r_n > 2D_n$ and $r_n < \delta$,

$$c_n \leq \left(1 + \frac{2D_n}{r_n - 2D_n}\right) c^{(\delta)}. \tag{4}$$

Proof Let σ_δ be a feasible path of length $c^{(\delta)}$ with strong δ-clearance. Construct the balls $B_1, \ldots, B_{M_n}$ with centers along σ_δ as in the proof of Theorem 1 (note we are not constructing B_{M_n+1} or B_{M_n+2}), except with radii D_n and centers separated by a segment of σ_δ of arc-length $r_n - 2D_n$. Note that $M_n = \lfloor c^{(\delta)}/(r_n - 2D_n)\rfloor$. Then by definition each B_i contains at least one point x_i. Furthermore, each x_i is connected to x_{i-1} in the gPRM graph (because x_i is contained in the ball of radius $r_n - D_n$ centered at x_{i-1}, and that ball is collision-free), and x_f is connected to x_{M_n} as well. Thus c_n is upper-bounded by the path tracing through $x_0, x_1, \ldots, x_{M_n}, x_f$:

$$\begin{aligned} c_n &\le \|x_1 - x_0\| + \sum_{i=2}^{M_n} \|x_i - x_{i-1}\| + \|x_f - x_{M_n}\| \le r_n - D_n + \sum_{i=2}^{M_n} r_n + \|x_f - x_{M_n}\| \\ &\le (M_n r_n - D_n) + \left(\left(\frac{c^{(\delta)}}{r_n - 2D_n} - \left\lfloor \frac{c^{(\delta)}}{r_n - 2D_n} \right\rfloor\right)(r_n - 2D_n) + D_n\right) \\ &= c^{(\delta)} + 2D_n M_n \le \left(1 + \frac{2D_n}{r_n - 2D_n}\right) c^{(\delta)}, \end{aligned}$$

where the second and third inequalities follow by considering the farthest possible distance between neighboring points, given their inclusion in their respective balls. □

Remark 1 (*Goal region simplification*) We note that the simplification of the goal region streamlines Eq. (4) and the associated analysis. A similar result can be derived for the more general case—this would require accounting for the curvature of the goal region and constructing B_{M_n+1} and B_{M_n+2}, whose radii and separation would add a term to the convergence rate.

Remark 2 (*Convergence rate in obstacle-free environments*) Note that when there are no obstacles, $\delta_{\max} = \infty$ and $c^{(\delta)} = \|x_f - x_0\|$ for all $\delta > 0$. Therefore, an immediate corollary of Theorem 2 is that the convergence rate in free space of gPRM to the *optimal* solution is upper-bounded by $2D_n/(r_n - 2D_n)$ for $r_n > 2D_n$.

Remark 3 (*Practical use of convergence rate*) Theorem 2 provides a convergence rate result to a shortest path with strong δ-clearance. This result is useful for two main reasons. First, in practice, the objective of path planning is often to find a high-quality path with some "buffer distance" from the obstacles, which is precisely captured by the notion of δ-clearance. Second, the convergence rate in Eq. (4) could be used to certify the performance of gPRM (and related batch planners) by placing some assumptions on the obstacle set (e.g., minimum separation distance among obstacles and/or curvature of their boundaries)—this is an interesting avenue for future research.

Both the asymptotic optimality and convergence rate results can be extended to other batch planners such as Lazy-PRM or FMT*. This is however omitted due to space limitations.

Lastly, we show that using a low-ℓ_2-dispersion lattice sample set, an asymptotically-optimal (AO) version of gPRM can be run that has lower-order computational complexity than any existing AO algorithm, namely $\omega(n)$ instead of $O(n\log(n))$.

Theorem 3 (Computational complexity with deterministic sampling) *gPRM run on n samples arranged in a d-dimensional cubic lattice with connection radius r_n satisfying Eq. (2) has computational complexity and space complexity*

$$O(n^2 r_n^d). \tag{5}$$

Proof The algorithm gPRM has three steps: (1) For each sampled point x, it needs to compute which other sampled points are within a distance r_n of x. (2) For each pair of sampled points within r_n of one another, their connecting edge needs to be checked for collision, and if collision-free, its edge-length needs to be computed. (3) The shortest path through the graph produced in steps (1) and (2) from the initial point to the goal region needs to be computed.

The lattice structure makes it trivially easy to bound a point's r_n-neighborhood by a bounding hypercube with side-length $2r_n$, ensuring only $O(nr_n^d)$ nearby points need to be checked for each of the n samples, so this step takes $O(n^2 r_n^d)$ time.

In step (2), one collision-check and at most one cost computation needs to be performed for each pair of points found in step (1) to be within r_n of one another. The number of such pairs can be bounded above by the number of sampled points times the size of each one's neighborhood, leading to a bound of the form $O(n \cdot nr_n^d)$. Thus step (2) takes $O(n^2 r_n^d)$ time.

After steps (1) and (2), a weighted (weights correspond to edge lengths) graph has been constructed on n vertices with a number of edges asymptotically upper-bounded by $n^2 r_n^d$. One more property of this graph, because it is on the cubic lattice, is that the number of distinct edge lengths is asymptotically upper-bounded by nr_n^d. An implementation of Dijkstra's algorithm for the single source shortest path problem is presented in [19] with running time linear in both the number of edges and the number of vertices times the number of distinct edge lengths. Since both are $O(n^2 r_n^d)$, that is the time complexity of step (3). □

Since Theorem 1 allows $r_n \in \omega(n^{-1/d})$ while maintaining AO, Theorem 3 implies that cubic-lattice sampling allows for an AO algorithm with computational and space complexity $\omega(n)$. All other AO algorithms in the literature have computational and space complexity at least $O(n \log(n))$. While the use of an $r_n \in \omega(n^{-1/d})$ makes the graph construction phase (steps (1) and (2)) $\omega(n)$, step (3) would in general take longer, as shortest-path algorithms on a general graph with n vertices requires $\Omega(n \log(n))$. Thus the lattice structure must be leveraged to improve the complexity of step (3). We note, however, that in practice the shortest-path algorithm is typically a trivial fraction of path planning runtime, as it requires none of the much-more-complex edge-cost and collision-free computations. If we ignore this component of the runtime, the result of Theorem 3 can be extended to other PRM-like algorithms such as FMT*, non-lattice sampling schemes such as the Halton/Hammersley sequence, and a k-nearest-neighbor implementation on a lattice (where k_n is taken so that each point would be connected to its r_n-neighborhood if there were no obstacles).

5 Numerical Experiments

In this section we numerically investigate the benefits of planning with low-dispersion sampling instead of i.i.d. sampling. Section 5.1 overviews the simulation environment used for this investigation. Section 5.2 details the deterministic low-dispersion sequences used, namely, lattices and the Halton sequence. Several simulations are then introduced and results compared to i.i.d. sampling in Sects. 5.3 and 5.4. We then briefly discuss non-i.i.d., random, low-dispersion sampling schemes in Sect. 5.5. Finally, in Sect. 5.6 we discuss the results of different connection radius scalings, specifically $(\log(\log(n))/n)^{1/d}$, to numerically validate the theoretical results in Sect. 4.

5.1 *Simulation Environment*

Simulations were written in C++ and MATLAB, and run using a Unix operating system with a 2.3 GHz processor and 8 GB of RAM. The C++ simulations were run through the Open Motion Planning Library (OMPL) [24]. The planning problems simulated in OMPL were rigid body problems whereas the simulations in MATLAB involved point robots and kinematic chains. For each planning problem, the entire implementation of gPRM was held fixed (including the sequence of r_n) except for the sampling strategy. Specifically, for all simulations (except the chain and those in Sect. 5.6), we used as connection radius $r_n = \gamma_{\mathrm{PRM}} \, (\log(n)/n)^{1/d}$, where $\gamma_{\mathrm{PRM}} = 2.2\left(1 + 1/d\right)^{1/d}(1/\zeta_d)^{1/d}$ and ζ_d is the volume of the unit ball in the d-dimensional Euclidean space. This choice ensures asymptotic optimality both for i.i.d. sample sequences [11, 12] and deterministic low-dispersion sequences (Theorem 1). To provide an exact "apples-to-apples" comparison, we do not present runtime results, but only results in terms of sample count. Accordingly, our results are independent of the specific implementation of gPRM. (Note that drawing samples represents a trivial fraction of the total algorithm runtime. Furthermore, as mentioned in the introduction, deterministic sampling even allows for possible speed-ups in computation.)

5.2 *Sampling Sequences*

We consider two deterministic low-dispersion sequences, namely the Halton sequence [8] and lattices. Halton sampling is based on a generalization of the van der Corput sequence and uses prime numbers as its base to form a low-dispersion sequence of points [8, 16]. Lattices in this work were implemented as a triangular lattice in two dimensions and a Sukharev grid in higher dimensions [25].

Along with the benefits of lattices described throughout the paper, they also present some challenges [16]. First, for basic cubic lattices with the same number of lattice points k per side, the total number of samples n is fixed to k^d, which limits the potential number of samples. For example, in 10 dimensions, the first four available sample counts are 1, 1024, 59,049, and 1,048,576. There are some strategies to allow for incremental sampling [17, 27, 28], but in this paper we overcome this difficulty by simply "weighting" dimensions. Explicitly, we allow each side to have a different number of lattice points. Independently incrementing each side's number of points by 1 increases the allowed resolution of n by a factor of d, as it allows all numbers of the form $n = (k-1)^m(k)^{d-m}$.

Second, lattices are sensitive to axis-alignment effects, whereby an axis-aligned obstacle can invalidate an entire axis-aligned row of samples. A simple solution to this problem is to rotate the entire lattice by a fixed amount (note this changes nothing about the ℓ_2-dispersion or nearest-neighbor sets). We chose to rotate each dimension by 10π degrees as an arbitrary angle far from zero (in practice, problems often show a "preferential" direction, e.g., vertical, so there may be an *a priori* natural choice of rotation angle).

Finally, in a Sukharev lattice in SE(2) or SE(3), each translational lattice point represents many rotational orientations. In SE(3), this means there are only k^3 translational points when $n = k^6$. In many rigid body planning problems, the translational dimension is more cluttered and the dominating term in the cost, so the effective reduction in translational samples severely hampers planning. A solution is to "spread" the points, which entails de-collapsing all the rotational orientations at each translational lattice point by spreading them deterministically in the translational dimensions around the original point.

5.3 Simulation Test Cases

Within the MATLAB environment, results were simulated for a point robot within a Euclidean unit hypercube with a variety of geometric obstacles over a range of dimensions. First, rectangular obstacles in 2D and 3D were generated with a fixed configuration that allowed for several homotopy classes of solutions. A 2D maze with rectangular obstacles was also created (Fig. 2a). These sets of rectangular obstacles are of particular interest as they represent a possible "worst-case" for lattice-based sampling because of the potential for axis alignment between samples and obstacles. The results, shown for the 2D maze in Fig. 2 and for all experiments in Table 1, show that Halton and lattice sampling outperform i.i.d. sampling in both success rate and solution cost.

To illustrate rectangular planning in higher dimensional space, we constructed a recursive maze obstacle environment. Each instance of the maze consists of two copies of the previous dimension's maze, separated by an obstacle with an opening through the new dimension, as detailed in [11]. Figure 3a shows the maze in 2D and Fig. 3b shows the maze in 3D with the two copies of the 2D maze in black and the opening in red. Halton and lattice sampling conferred similar benefits in the recursive mazes in 2D, 3D, 4D, 5D, 6D, and 8D as they did in other simulations (see Table 1).

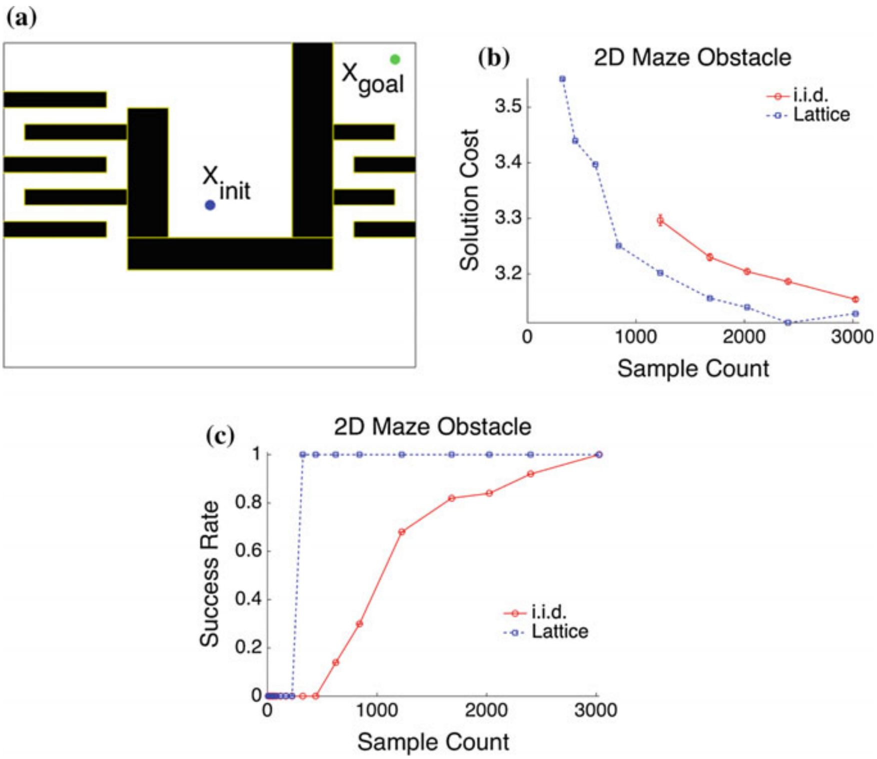


Fig. 2 **a** The planning setup for a point robot with rectangular obstacles in a 2D maze. **b**, **c** The results for solution cost and success rate versus sample count (averaged over 50 runs). For clarity we only report data for i.i.d. sampling and lattice sequences, results including Halton sampling are reported in Table 1. Only points with greater than 50% success are shown in **b**

In addition to the rectangular obstacles, hyperspherical obstacles within a Euclidean unit hypercube were generated to compare performance on a smooth obstacle set with no possibility of axis alignment between samples and obstacles. The setups for 2D and 3D (Fig. 3c, d) were fixed, while in 4D, obstacles were randomly generated to match a specified spatial coverage. Again, Halton and lattice sampling consistently outperformed i.i.d. sampling, as shown in Table 1.

As an additional high-dimensional example, an 8D kinematic chain planning problem with rotational joints, eight links, and a fixed base was created (Fig. 3e). The solution required the chain to be extracted from one opening and inserted into the other, as inspired by [17]. The chain cost function was set as the sum of the absolute values of the angle differences over all links and the connection radius was thus scaled by $\sqrt{d}\pi$. With this high dimension and new cost function the Halton and lattice perform as well as or better than i.i.d. sampling (see Table 1).

Within the OMPL environment, rigid body planning problems from the OMPL test banks were posed for SE(2) and SE(3) configuration spaces. In the SE(2) case, one

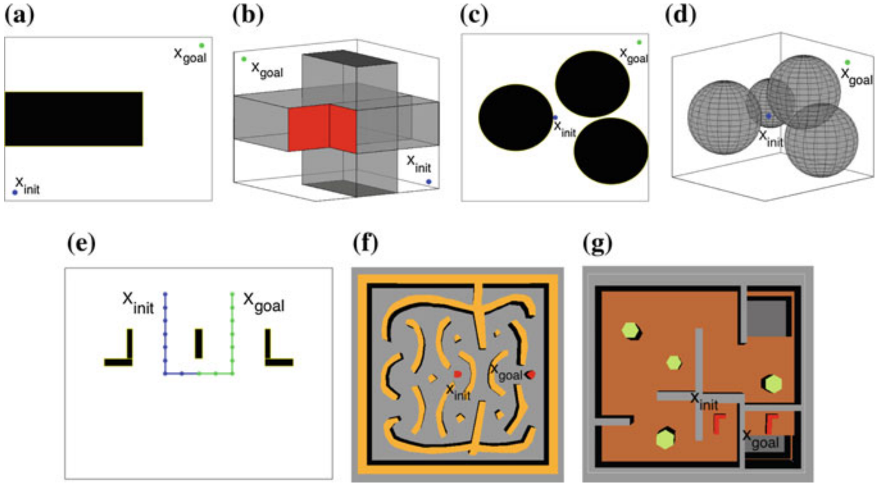


Fig. 3 Images of the recursive maze planning problem in 2D (**a**) and 3D (**b**) and the spherical obstacle sets in 2D (**c**) and 3D (**d**). Also shown are an 8D kinematic chain planning problem in **e** and the OMPL rigid body planning problems for SE(2) and SE(3) in **f** and **g** respectively. A summary of results can be found in Table 1

rotational and two translational degrees of freedom are available, resulting in a three dimensional space, shown in Fig. 3f. The SE(3) problem consists of an "L-shaped" robot moving with three translational and three rotational degrees of freedom, resulting in six total problem dimensions, shown in Fig. 3g. The SE(2) and SE(3) lattices use the spreading method described in Sect. 5.2.

5.4 *Summary of Results*

Table 1 shows a summary of the results from simulations detailed in Sect. 5.3. Results are shown normalized by the i.i.d. sampling results. In each case the sample count at which a success rate greater than 90% is achieved and sustained is reported. Additionally, the solution costs at a medium and high sampling count are shown. For nearly all cases the lattice sampling finds a solution with fewer or an equal number of samples and of lower or equal cost than that found by i.i.d. sampling. The Halton sampling also always finds a solution at lower sample counts than i.i.d. sampling, and almost always finds solutions of lower cost as well. The low-dispersion sequences particularly outperform i.i.d. sampling in terms of number of samples required for a 90% success rate.

Table 1 Summary of results. Each entry is divided by the results of i.i.d. sampling (averaged over 50 runs). For Halton sampling and lattice sampling, the number of samples at which 90% success is achieved and the cost at a medium number of samples (near 700) and a high number of samples are shown (highest samples simulated, always 3000 or greater). Note that nearly all table entries are below 100%, meaning the Halton and lattice sampling outperformed i.i.d. sampling

		Halton			Lattice		
Dim	Obstacles (%)	90% Success (%)	Medium (%)	High (%)	90% Success (%)	Medium (%)	High (%)
2	Rectangular	38	118	80	15	56	80
3	Rectangular	36	88	94	19	80	87
2	Rect Maze	13	98	99	13	100	99
2	Sphere	16	93	99	7	93	99
3	Sphere	36	97	100	8	97	99
4	Sphere	100	97	97	100	97	100
2	Recursive Maze	33	100	100	18	100	100
3	Recursive Maze	22	95	99	22	96	98
4	Recursive Maze	56	95	98	56	100	100
5	Recursive Maze	45	97	96	60	95	96
6	Recursive Maze	56	95	97	75	94	96
8	Recursive Maze	56	98	99	75	99	99
8	Chain	67	112	91	7	76	87
3	SE(2)	81	96	100	81	101	101
6	SE(3)	32	96	93	42	94	95

5.5 *Nondeterministic Sampling Sequences*

The above simulations showed deterministic lattice sampling, with a fixed rotation around each axis, and the deterministic Halton sequence outperform i.i.d. sampling. Both deterministic sequences have low ℓ_2-dispersions of $O(n^{-1/d})$, but sequences with the same order ℓ_2-dispersion need not be deterministic. Figure 4 shows results for a randomly rotated and randomly offset version of the lattice (again, the ℓ_2-dispersion and neighborhoods are all still deterministically the same). The same cases in Table 1 were run for the randomly rotated lattice and the results showed it performed as well as or better than i.i.d. sampling (over 50 runs). In general, low-dispersion random sequences might provide some advantages, e.g., eliminating axis

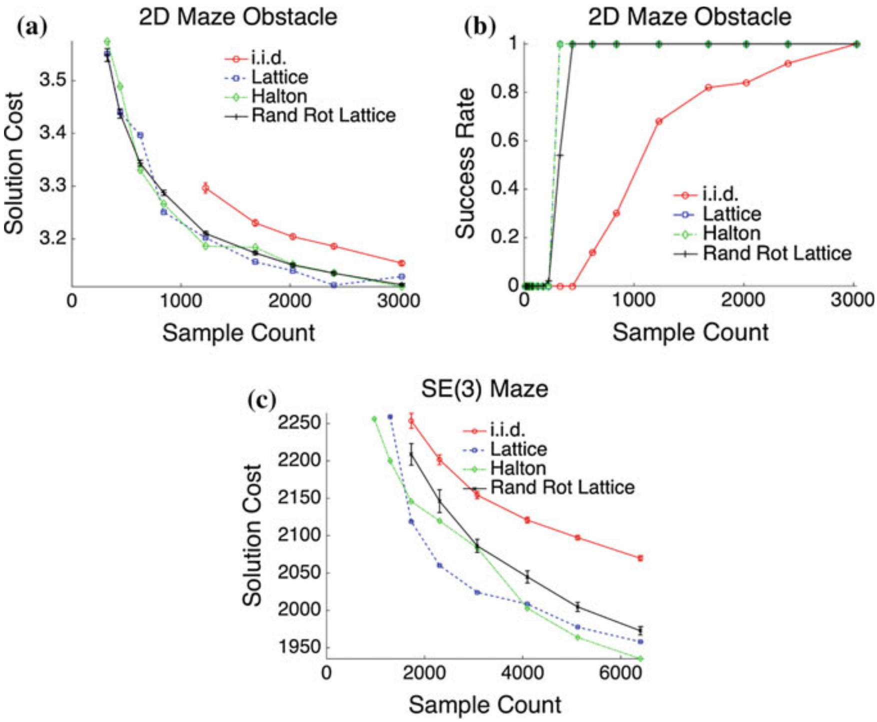


Fig. 4 Results for deterministic and nondeterministic low-dispersion sampling. "Rand Rot Lattice" refers to a randomly rotated lattice

alignment issues while still enjoying deterministic guarantees (see Sect. 4). Their further study represents an interesting direction for future research.

5.6 *Radius Scaling of* $(\log(\log(n))/n)^{1/d}$

To validate the results in Sect. 4 (specifically, Theorem 1), we investigate the effects of scaling the radius as $\gamma_{\text{PRM}}(\log(\log(n))/n)^{1/d}$. The results of these simulations are shown in Fig. 5 for 2D rectangular obstacles. In the case of $\gamma_{\text{PRM}}\,(\log(\log(n))/n)^{1/d}$, the lattice planning appears to still converge to the optimal cost, as predicted by Theorem 1.

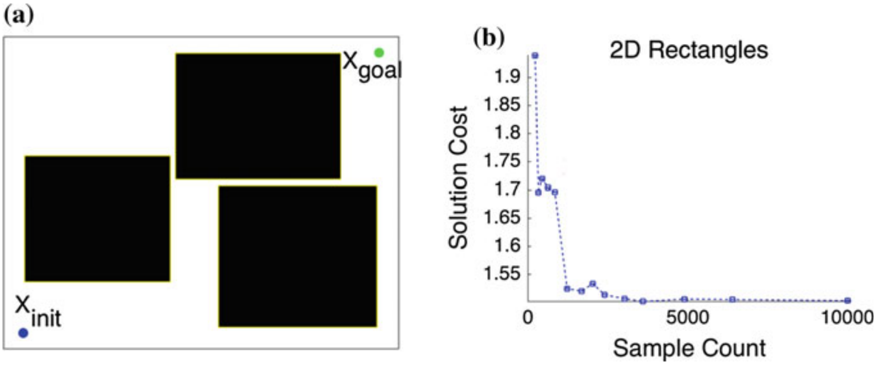


Fig. 5 **a** planning setup for study of radius scaling. **b** planning results for lattice sampling with radius scaling of $\gamma_{\mathrm{PRM}}\,(\log(\log(n))/n)^{1/d}$

6 Conclusions

This paper has shown that using low-dispersion sampling strategies (in particular, deterministic) can provide substantial benefits for solving the optimal path planning problem with sampling-based algorithms, in terms of deterministic performance guarantees, reduced computational complexity per given number of samples, and superior practical performance.

This paper opens several directions for future research. First, we plan to extend our results to the case of kinodynamic motion planning. In particular, the ℓ_2-dispersion is a natural quantity in geometric planning as it corresponds to Euclidean balls, and Euclidean distance is exactly the relevant measure of distance between two points. However, differential constraints require a different notion of distance based on reachability sets, and as such if we can find sampling schemes that achieve low dispersion in terms of reachability sets, we can extend the results here to the kinodynamic setting. Second, it is of interest to extend the results herein to other classes of sampling-based motion planning algorithms, especially the large class of *anytime* algorithms (e.g., RRT/RRT*). This leads directly into a third key direction, which is to study alternative low-dispersion sampling strategies beyond the few considered here, particularly *incremental* sequences for use in anytime algorithms. There is already some work in this area, although thus far it has focused on the use of such sequences for the feasibility problem [17, 27, 28]. It may also be of interest to study low-dispersion sampling strategies that incorporate prior knowledge of the problem by sampling non-uniformly. Non-uniform i.i.d. sampling has found success by, for instance, sampling more densely near obstacles or more intricate parts of the configuration space; we believe only minor modifications are needed to apply the same principles to low-dispersion sampling, but defer such inquiry to future work. Fourth, we plan to investigate the topological relationship between the optimal path cost and that of the best strong-δ-clear path, in order to frame the convergence rate in terms of the true optimal cost. Fifth, from a practical standpoint, it is of interest to adapt existing algorithms or design new ones that explicitly leverage the structure of low-dispersion sequences (e.g., fast nearest neighbor indexing or precomputed data structures). This would be especially beneficial in the domain of kinodynamic motion planning. Finally, leveraging our convergence rate results, we plan to investigate the issue of certification for sampling-based planners, e.g., in the context of trajectory planning for drones or self-driving cars.

References

1. Alterovitz, R., Patil, S., Derbakova, A.: Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning. In: Proceedings of the IEEE Conference on Robotics and Automation, pp. 3706–3712 (2011)
2. Arslan, O., Tsiotras, P.: Use of relaxation methods in sampling-based algorithms for optimal motion planning. In: Proceedings of the IEEE Conference on Robotics and Automation, pp. 2421–2428. Karlsruhe, Germany (2013). doi:10.1109/ICRA.2013.6630906. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6630906
3. Barraquand, J., Kavraki, L., Motwani, R., Latombe, J.-C., Li, T.-Y., Raghavan, P.: A random sampling scheme for path planning. Int. J. Robot. Res. **16**(6), 759–774 (2000). doi:10.1177/027836499701600604. http://ijr.sagepub.com/content/16/6/759.short
4. Bohlin, R., Kavraki, L.: Path planning using lazy PRM. In: Proceedings of the IEEE Conference on Robotics and Automation, pp. 521–528 (2000)
5. Branicky, M.S., LaValle, S.M., Olson, K., Yang, L.: Quasi-randomized path planning. In: Proceedings of the IEEE Conference on Robotics and Automation, vol. 2, pp. 1481–1487 (2001)
6. Deheuvels, P.: Strong bounds for multidimensional spacings. Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete **64**(4), 411–424 (1983)
7. Dobson, A., Moustakides, G., Bekris, K.E.: Geometric probability results for bounding path quality in sampling-based roadmaps after finite computation. In: Proceedings of the IEEE Conference on Robotics and Automation (2015)
8. Halton, J.H.: On the efficiency of certain quasirandom sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik **2**(1), 84–90 (1960). doi:10.1007/BF01386213. http://link.springer.com/article/10.1007%2FBF01386213?LI=true
9. Hsu, D., Latombe, J.-C., Motwani, R.: Path planning in expansive configuration spaces. Int. J. Comput. Geom. Appl. **9**(4), 495–512 (1999). doi:10.1142/S0218195999000285. http://www.worldscientific.com/doi/abs/10.1142/S0218195999000285
10. Hsu, D., Latombe, J.-C., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robot. Res. **25**(7), 627–643 (2006)
11. Janson, L., Schmerling, E., Clark, A., Pavone, M.: Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions. Int. J. Robot. Res. **34**(7), 883–921 (2015). doi:10.1177/0278364915577958
12. Karaman, Sertac, Frazzoli, Emilio: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011). doi:10.1177/0278364911406761
13. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996). doi:10.1109/70.508439
14. Lavalle, S.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
15. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5), 378–400 (2001). doi:10.1177/02783640122067453. http://ijr.sagepub.com/content/20/5/378.short

16. LaValle, S.M., Branicky, M.S., Lindemann, S.R.: On the relationship between classical grid search and probabilistic roadmaps. Int. J. Robot. Res. **23**(7–8), 673–692 (2004)
17. Lindemann, S.R., Yershova, A., LaValle, S.M.: Incremental grid sampling strategies in robotics. In: Workshop on Algorithmic Foundations of Robotics, pp. 313–328 (2005)
18. Niederreiter, H.: Random number generation and Quasi-Monte Carlo methods. In: CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial & Applied Mathematics, vol. 63 (1992)
19. Orlin, J.B., Madduri, K., Subramani, K., Williamson, M.: A faster algorithm for the single source shortest path problem with few distinct positive lengths. J. Discret. Algorithms **8**(2), 189–198 (2010)
20. Phillips, J.M., Bedrossian, N., Kavraki, L.E.: Guided expansive spaces trees: a search strategy for motion- and cost-constrained state spaces. In: Proceedings of the IEEE Conference on Robotics and Automation, vol. 4, pp. 3968–3973. New Orleans, LA (2004). doi:10.1109/ROBOT.2004.1308890. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1308890
21. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. J. Field Robot. **26**(3), 308–333 (2009)
22. Plaku, E., Bekris, K.E., Chen, B.Y., Ladd, A.M., Kavraki, L.E.: Sampling-based roadmap of trees for parallel motion planning. IEEE Trans. Robot. **21**(4), 597–608 (2005)
23. Stentz, A.: Optimal and efficient path planning for unknown and dynamic environments. Int. J. Robot. Autom. **10**(3), 89–100 (1995)
24. Şucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. IEEE Robot. Autom. Mag. **19**(4), 72–82 (2012). doi:10.1109/MRA.2012.2205651. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6377468
25. Sukharev, A.G.: Optimal strategies of the search for an extremum. USSR Comput. Math. Math. Phys. **11**(4), 119–137 (1971). doi:10.1016/0041-5553(71)90008-5. http://www.sciencedirect.com/science/article/pii/0041555371900085
26. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
27. Yershova, A., LaValle, S.M.: Deterministic sampling methods for spheres and SO(3). In: Proceedings of the IEEE Conference on Robotics and Automation, vol. 4, pp. 3974–3980 (2004)
28. Yershova, A., Jain, S., Lavalle, S.M., Mitchell, J.C.: Generating uniform incremental grids on SO(3) using the Hopf fibration. Int. J. Robot. Res. **29**(7), 801–812 (2010)

Path Following for Mobile Manipulators

Rajan Gill, Dana Kulić and Christopher Nielsen

1 Introduction

The problem of designing feedback control laws that make a robot follow a desired curve in its workspace can be broadly classified as either a trajectory tracking or path following problem. Trajectory tracking consists of tracking a curve with an assigned time parametrization [10]. This approach may not be suitable for certain applications as it may limit the attainable performance, be unnecessarily demanding or even infeasible [8]. On the other hand, path following or manoeuvre regulation controllers, as previously defined in [10], drive a system's output to a desired path and make the output traverse the path without a pre-specified time parametrization. In general, path following results in smoother convergence to the desired path compared to trajectory tracking, the control signals are less likely to saturate [17], and performance limitations for non-minimum phase systems are removed [1]. Path following controllers can also render the desired path attractive and invariant [11]. This is a useful property in robotics, since if a disturbance or obstacle is preventing the output of a robot to proceed along the path, the robot will stay on the path until the disturbance is removed [11]. Furthermore, if the robot's output is perturbed so that it leaves the path, the path following controller will drive the output back to the desired path. There are several approaches for path following, see [9] for a complete literature review.

R. Gill (✉) · D. Kulić · C. Nielsen
Department of Electrical and Computer Engineering, University of Waterloo,
200 University Avenue West, Waterloo, ON N2L 3G1, Canada
e-mail: rjgill@uwaterloo.ca

D. Kulić
e-mail: dkulic@uwaterloo.ca

C. Nielsen
e-mail: cnielsen@uwaterloo.ca

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_30

The majority of applications and experiments of mobile manipulators do not simultaneously control the manipulator and the mobile base. That is, they control the manipulator and the mobile base in a sequential, not parallel (as is done in this paper), manner [3, 18, 21, 24]. Mobile manipulator systems can also be modelled as a single redundant system [25]. Previous works have tackled the trajectory tracking problem by resolving the redundancies at the kinematic level, but, unlike in this paper, a trajectory for the end-effector as well as the mobile base is specified [12, 22]. In this paper, the motion planning process is simplified since there is no need to plan separate trajectories (or paths) for the base and the end-effector. Khatib extended the dynamic-level redundancy resolution developed for torque-input model of robot manipulators [14] to holonomic mobile platforms [15], whereas the approach in this paper can also be applied directly to combined motor-input manipulator mounted on nonholonomic ground vehicles. In [25], the authors look at the coordinated mobile manipulator problem, where, unlike in this paper, the manipulator dynamics are neglected, to track a reference trajectory in the workspace of the mobile platform while maximizing the manipulability measure.

In this paper, we apply the path following control approach of [9] to general mobile manipulator systems. We use dynamic extension [2, 13] to transform the dynamics of the mobile manipulator system that are tangential and transversal to the path into linear subsystems, and doesn't result in a singularity when the base velocity is zero unlike in previous work [2]. The remaining dynamics, redundant to following the path, appear as internal dynamics of the closed-loop system. A novel redundancy resolution technique is used at the dynamic level, and is experimentally shown to yield bounded internal dynamics, while maintaining a preferred manipulator posture. This scheme can easily be tuned by the designer to achieve various desired poses. The result is an automatic unified path following controller that compensates for the dynamics of the mobile manipulator system, controlling the mobile base and manipulator simultaneously, rendering a desired path in the output space of the system to be attractive and invariant. There is no trajectory that needs to be planned and tracked by the manipulator nor the mobile base, and the coordination between the two is done automatically by our proposed path following controller.

Notation: Let $\langle x, \ y \rangle$ denote the inner product of vectors x and y in $\mathbb{R}^n$. The Euclidean norm of a vector and induced matrix norm are both denoted by $\|\cdot\|$. The notation $s \circ h : A \to C$ represents the composition of maps $s : B \to C$ and $h : A \to B$. The ith element of a vector x is denoted x_i, and the row i to j and column k to l submatrix of A is denoted as $A_{i:j,k:l}$. The symbol $:=$ means equal by definition. Given a C^1 mapping $\phi : \mathbb{R}^n \to \mathbb{R}^m$ let $\mathrm{d}\phi_x$ be its Jacobian evaluated at $x \in \mathbb{R}^n$. If $f, \ g : \mathbb{R}^n \to \mathbb{R}^n$ are smooth vector fields we use the following standard notation for iterated Lie derivatives $L_f^0\phi := \phi$, $L_f^k\phi := L_f(L_f^{k-1}\phi) = \langle \mathrm{d}L_f^{k-1}\phi_x, \ f(x)\rangle$, $L_gL_f\phi := L_g(L_f\phi) = \langle \mathrm{d}L_f\phi_x, \ g(x)\rangle$. Finally, $0_{n\times m}$ is the n by m zero matrix and $I_{n\times m}$ is a n by m matrix of zeros with ones on its main diagonal.

2 Class of Systems

We consider a fully actuated manipulator mounted on a mobile ground vehicle. For simplicity, the combined systems are assumed to be decoupled. This is a valid system model assumption for general mobile manipulator systems [4, 15] and for our experimental platform (Sect. 5).

Manipulator Subsystem: The first subsystem is a fully actuated manipulator system with N configuration variables and N control inputs. The dynamic model is of the familiar form [23]

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = A(q)u_m \tag{1}$$

where $u_m \in \mathbb{R}^N$ is the manipulator control input (usually motor voltages [23]), and $(q,\dot{q}) \in \mathbb{R}^N \times \mathbb{R}^N$ are joint positions and velocities.[1]

The output of system (1), the task space of the manipulator, lives in a P-dimensional space, and is given in local frame $\mathscr{B}$ attached to the vehicle base (see Fig. 1) as

$$y_m = h(q) \tag{2}$$

where $h : \mathbb{R}^N \to \mathbb{R}^P$ is smooth. In addition, let $J : \mathbb{R}^N \to \mathbb{R}^{P \times N}, q \mapsto \mathrm{d}h_q$ represent the manipulator Jacobian [23]. We assume, without loss of generality, that $P \geq 3$ and that the first 3 components of h correspond to the Euclidean position of the end-effector in the local frame $\mathscr{B}$ (Fig. 1). Any additional outputs ($P > 3$) of h represent orientations of the end-effector in the world frame. For example, if $P = 4$, the fourth row of h could represent the end-effector angle with respect to the horizontal ground plane [20].

Vehicle Subsystem: The position $x^b \in \mathbb{R}^2$ and orientation $\theta \in \mathbb{R}$ of frame $\mathscr{B}$ in an inertial frame $\mathscr{O}$ (see Fig. 1) are governed by the vehicle's dynamics. A general vehicle kinematic model is given as [5]:

$$\begin{bmatrix} \dot{x}^b \\ \dot{\theta} \end{bmatrix} = R(\theta)\Sigma(\sigma)\gamma_b \tag{3}$$

$$\dot{\sigma} = \gamma_s \tag{4}$$

where $R(\theta) \in \mathsf{SO}(3)$ is the rotation matrix

$$R(\theta) := \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

[1] The results of this paper do not rely on the assumption that the state space be Euclidean. One could replace $\mathbb{R}^N$ by a smooth Riemannian manifold. Nonetheless, we assume $x \in \mathbb{R}^N$ to avoid unnecessarily cumbersome notation.

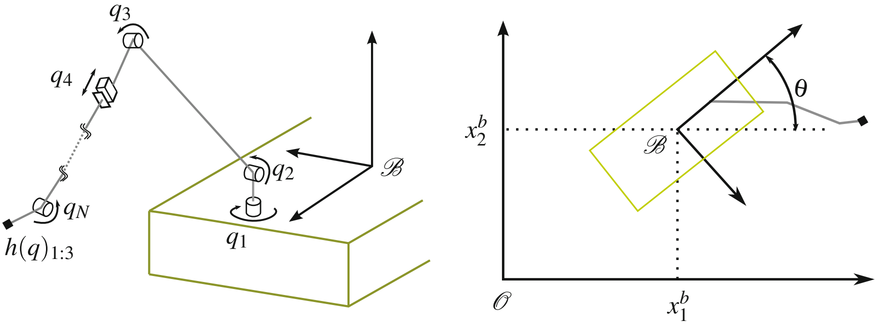


Fig. 1 Schematic of the mobile manipulator. The end-effector position $h(q)$ is expressed in the local-frame $\mathscr{M}$ attached to the *bottom* of the mobile base. The position and orientation of the local-frame in the inertial frame $\mathscr{O}$ are determined by x^b, θ

$\Sigma : \mathbb{R}^{\delta_s} \to \mathbb{R}^{3\times\delta_b}$ is a full rank matrix [5, pg. 283] that depends on the steering angle $\sigma \in \mathbb{R}^{\delta_s}$ of the steering wheels, $\delta_s \in \{0, 1, 2\}$ is the *degree of steerability* [5, pg. 273], $\delta_b \in \{1, 2, 3\}$ is the *degree of mobility* of the mobile base [5, pg. 272], and $\gamma_b \in \mathbb{R}^{\delta_b}$, $\gamma_s \in \mathbb{R}^{\delta_s}$ are the inputs to the vehicle. The degrees of mobility and steerability δ_b, δ_s depend on the design of the vehicle and satisfy the constraint $\delta_b + \delta_s \in \{2, 3\}$. The kinematic model (3) encompasses many wheeled-ground vehicle designs including car-like vehicles, unicycles (see Sect. 5), and omnidirectional vehicles [5, Chap. 7.2]. If the vehicle has no steering wheels, such as those found on differential drive robots (see Sect. 5), then $\delta_s = 0$ and Σ is a constant matrix.

Output Model:

The p-dimensional output of the overall system is taken to be manipulator position $h(q)$ in the inertial frame in the following sense

$$y := H(q, x^b, \theta) = \begin{bmatrix} R(\theta)_{1:2,1:2} & 0_{2\times(P-2)} \\ 0_{(p-2)\times 2} & I_{(p-2)\times(P-2)} \end{bmatrix} h(q) + \begin{bmatrix} x_1^b \\ x_2^b \\ 0_{(p-2)\times 1} \end{bmatrix}. \qquad (6)$$

where $P \geq p \geq 2$ is the dimension of the output space. If $p = 2$, then only the planar position of the end-effector in the inertial frame is of concern. If both $P > 3$ and $p > 3$, then a component of the end-effector orientation is also to be specified.

Assumption 2.1 (*Dexterity*) The mobile manipulator satisfies $N + \delta_b \geq p$ (see Remark 1), i.e., the system has enough degrees of freedom to follow arbitrary paths in the reachable space contained in $\mathbb{R}^p$. □

3 Problem Formulation

The problem studied in this paper is to find a continuous feedback control law for u_m, γ_b, γ_s for the mobile manipulator system such that the output y exponentially approaches a path in the output space $\mathbb{R}^p$ and traverses it towards a given desired position or with a given velocity profile. This will be done via set stabilization [19]. Since the mobile manipulator system is redundant, the redundancy is used to avoid manipulator joint limits and steering limits (if any) of the vehicle.

We assume that we are given a $(p+1)$-times continuously differentiable, parametrized curve in the output space of the mobile manipulator system,

$$\sigma : \mathbb{R} \to \mathbb{R}^p \tag{7}$$

which can be generated by spline interpolating a series of waypoints [6]. If $p > 3$, then the path also specifies a desired orientation of the end-effector. Given $[\lambda_{\min}, \lambda_{\max}] \subseteq \mathbb{R}$, the desired path is the set $\mathscr{P} := \sigma([\lambda_{\min}, \lambda_{\max}])$. We assume that the curve (7) is framed:

Assumption 3.1 (*framed curves*) For all $\lambda \in [\lambda_{\min}, \lambda_{\max}]$, $\sigma'(\lambda), \ldots, \sigma^{(p)}(\lambda)$ are linearly independent. □

Assumption 3.1 allows for the Frenet-Serret frame (FSF) of the path to be well-defined [16], and the prospect of it being violated is very low when spline-interpolating waypoints (see Sect. 5). In this paper we use Assumption 3.1 to generate a zero-level set representation of $\mathscr{P}$ in the state space of (12). Assumption 3.1 can be relaxed if a zero-level set representation of the path is already available, see [9].

The goals of this paper are to determine a control law for u_m, γ_b, γ_s such that

PF 1: The set $\mathscr{P}$ is rendered output invariant and locally attractive.[2]
PF 2: A desired position or velocity profile (tangent to the path) $\eta^{\text{ref}} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is tracked.
PF 3: Redundant dynamics (internal dynamics for the closed-loop system) remain bounded while ensuring joint limits of the manipulator are respected.

4 Path Following Control Design

The control design approach relies on the output y having a well-defined relative degree of $\{2, \ldots, 2\}$. That is, the control inputs u_m, γ_b, γ_s all appear in the second time-derivative of y. It can be shown that this is not the case: the control inputs γ_b appear too soon by one derivative. Hence, by delaying the appearance of this input

[2] Invariance: if for some time $t = 0$ the state $x(0)$ is appropriately initialized with $y = H(x(0)) \in \mathscr{P}$, then $(\forall t \geq 0)\ H(x(t)) \in \mathscr{P}$. Attractiveness: for initial conditions $x(0)$ such that the output $H(x(0))$ is in a neighbourhood of the desired path $\mathscr{P}$, $H(x(t)) \to \mathscr{P}$ as $t \to \infty$.

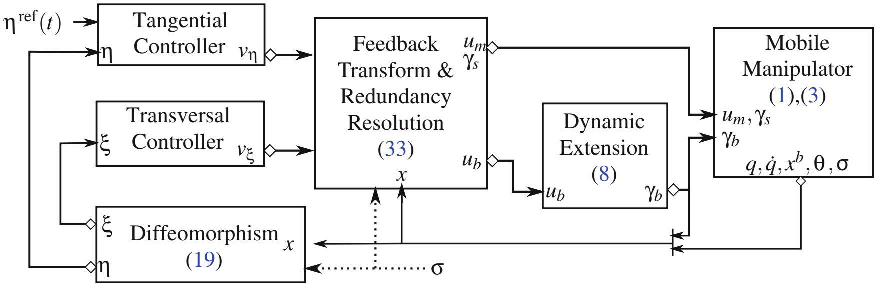


Fig. 2 Path following controller block diagram for mobile manipulator systems

via dynamic extension, we can make y have the desired relative degree. Then a "virtual" output is constructed that incorporates the desired path (7). Through partial feedback linearization, the virtual output partitions the system dynamics into controllable transversal and tangential sub-systems that govern the motion transversal to the path and along the path, respectively, and a redundant sub-system that governs the motion of the system that does not produce any output motion. A novel redundancy resolution approach is then used to ensure the redundant dynamics remain bounded, while maintaining joint limits of the manipulator. The overall block diagram can be found below in Fig. 2.

Dynamic Extension: Dynamic extension amounts to controlling the derivative(s) of the actual control input. Since γ_b appears too soon by one derivative when computing the derivatives of y, we introduce an auxiliary control variable $u_b \in \mathbb{R}^{\delta_b}$ as follows:

$$\dot{\gamma}_b = \rho\gamma_b + u_b \tag{8}$$

where $\rho < 0$ is a damping coefficient. It is typical with dynamic extension [2] to let $\rho = 0$, however we will see in Sect. 4 that setting $\rho = 0$ can result in unbounded internal dynamics.

To facilitate the control design procedure, we write the dynamics (1), (3), (8) with output (6) in state-space form. First, let $x_c := (q, x^b, \theta) \in \mathbb{R}^N \times \mathbb{R}^2 \times \mathbb{R}$, $x_v := (\dot{q}, \gamma_b, \sigma) \in \mathbb{R}^N \times \mathbb{R}^{\delta_b} \times \mathbb{R}^{\delta_s}$. Then,

$$\dot{x}_c = \begin{bmatrix} \dot{q} \\ R(\theta)\Sigma(\sigma)\gamma_b \end{bmatrix} =: F_c(x) \tag{9}$$

$$\dot{x}_v = \begin{bmatrix} -M^{-1}(q)\left(C(q,\dot{q})\dot{q} + G(q)\right) \\ \rho\gamma_b \\ 0_{\delta_s \times 1} \end{bmatrix} + \begin{bmatrix} M^{-1}(q)A(q) & 0_{N\times\delta_b} & 0_{N\times\delta_s} \\ 0_{\delta_b\times N} & I_{\delta_b\times\delta_b} & 0_{\delta_s\times\delta_s} \\ 0_{\delta_b\times N} & 0_{\delta_b\times\delta_b} & I_{\delta_s\times\delta_s} \end{bmatrix} \begin{bmatrix} u_m \\ u_b \\ \gamma_s \end{bmatrix} \tag{10}$$

$$=: F_v(x) + G_v(x)u \tag{11}$$

where $x := (x_c, x_v) \in \mathbb{R}^{2N+3+\delta_b+\delta_s}$ is the state of the system, $u := (u_m, u_b, \gamma_s) \in \mathbb{R}^{N+\delta_b+\delta_s}$ is the control input to the system, and y is the output. Note that $F_c(x)$ and $F_v(x)$ are smooth vector fields, $G_v(x)$ is smooth and has full rank, and $\frac{\partial H(x_c)}{\partial x_v} = 0$. The control inputs of the physical system (1), (3) are u_m, γ_b, γ_s, while (8) constitutes the dynamic portion of the controller implemented in the control algorithm.

The state space equations can be compactly written as

$$\dot{x} = \begin{bmatrix} \dot{x}_c \\ \dot{x}_v \end{bmatrix} = \begin{bmatrix} F_c(x) \\ F_v(x) \end{bmatrix} + \begin{bmatrix} 0_{(N+3)\times(N+\delta_b+\delta_s)} \\ G_v(x) \end{bmatrix} u =: f(x) + g(x)u \tag{12}$$

$$y = H(x_c) \tag{13}$$

The mobile manipulator system is redundant in the sense that $\dim u = N + \delta_b + \delta_s \geq \dim y = p$.

Virtual Outputs: The first virtual output, $\eta_1(x)$, is the tangential position of the output y along the desired path. Denote the parameter of curve σ that corresponds to the closest point to the output y as $\lambda^* \in [\lambda_{\min}, \lambda_{\max}]$

$$\lambda^* := \varpi(y) := \underset{\lambda \in [\lambda_{\min}, \lambda_{\max}]}{\arg\inf} \|y - \sigma(\lambda)\| . \tag{14}$$

The minimization for λ^* is done numerically using a gradient-descent-like algorithm [9]. The first virtual output is the projected, traversed arc length along the curve:

$$\eta_1 = \eta_1(x) := \int_{\lambda_{\min}}^{\lambda^*} \left\| \frac{\mathrm{d}\sigma(\lambda)}{\mathrm{d}\lambda} \right\| \mathrm{d}\lambda \Bigg|_{\lambda^* = \varpi \circ H(x_c)} . \tag{15}$$

This integral does not have to be computed for real time implementation if only tangential velocity control is required, as in the case of our experiment (see Sect. 5).

The remaining virtual outputs are selected to represent the cross track error to the path using the remaining FSF normal vectors, known as transversal positions [9]. The generalized Frenet-Serret (FS) vectors are constructed applying the Gram-Schmidt Orthonormalization process to the vectors $\sigma'(\lambda)$, $\sigma''(\lambda)$,..., $\sigma^{(p)}(\lambda)$:

$$\mathbf{e}_j(\lambda) := \frac{\bar{\mathbf{e}}_j(\lambda)}{\|\bar{\mathbf{e}}_j(\lambda)\|} \tag{16}$$

where

$$\bar{\mathbf{e}}_j(\lambda) := \sigma^{(j)}(\lambda) - \sum_{i=1}^{j-1} \left\langle \sigma^{(j)}(\lambda), \mathbf{e}_i(\lambda) \right\rangle \mathbf{e}_i(\lambda),$$

for $j \in \{1, \ldots, p\}$. This formulation is well-defined by Assumption 3.1. If FSF are not feasible, other frames may work [8].

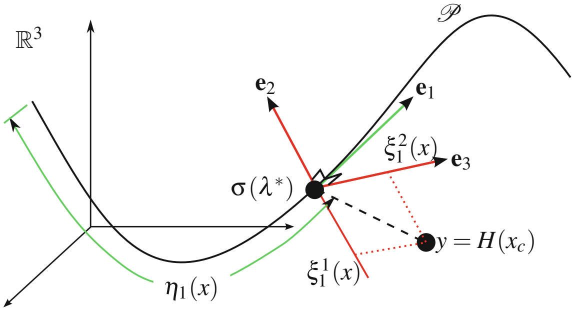


Fig. 3 The tangential and transversal state positions when $p = 3$

The transversal positions can be computed by projecting the error to the path, $y - \sigma(\lambda^*)$, onto each of the FS normal vectors:

$$\xi_1^{j-1} = \xi_1^{j-1}(x) := \left\langle \mathbf{e}_j(\lambda^*), H(x_c) - \sigma_{k^*}(\lambda^*) \right\rangle \big|_{\lambda^* = \varpi \circ H(x_c)} \tag{17}$$

for $j \in \{2, \dots, p\}$.

The tangential and transversal positions are illustrated in Fig. 3 for $p = 3$.

Dynamics and Control: It will be shown that the virtual output $(\eta_1(x), \xi_1^1(x), \dots, \xi_1^{p-1}(x)) \in \mathbb{R}^p$ has a well-defined vector relative degree of $\{2, 2, \dots, 2\}$ for all $x \in \mathscr{U}$, where

$$\mathscr{U} := \left\{ x \in \mathbb{R}^{2N+3+\delta_b+\delta_s} \mid \operatorname{rank}\left(\frac{\partial H}{\partial x_c} \frac{\partial F_c}{\partial x_v} G_v(x) \right) = p \right\} \tag{18}$$

(see (24)) when (14) is solved numerically [9]. We emphasize that (14) is solved for numerically by a local search since if the output y is equidistant to multiple points on the path, then (14) is not well defined and neither is the relative degree. Under these conditions, there exists a local coordinate transformation

$$T : \mathscr{U} \to \mathbb{R}^{2N+3+\delta_b+\delta_s} \tag{19}$$

$$x \mapsto (\eta_1(x), L_f\eta_1(x), \xi_1^1(x), L_f\xi_1^1(x), \dots, \xi_1^{p-1}(x), L_f\xi_1^{p-1}(x), \varphi(x)) \tag{20}$$

$$= (\eta_1, \eta_2, \xi_1^1, \xi_2^1, \dots, \xi_1^{p-1}, \xi_2^{p-1}, \zeta), \tag{21}$$

which is a diffeomorphism onto its image for a suitable choice of the function $\varphi : \mathbb{R}^{2N+3+\delta_b+\delta_s} \to \mathbb{R}^{2N+3+\delta_b+\delta_s-2p}$ [13] (which may limit the domain of T by [13, Proposition 5.1.2], but in practice (see Sect. 5), these functions do not need to be computed unless one wants to visualize the redundant dynamics or study their stability properties). Note that since the virtual output η_1 represents the tangential position of the output y along the path, η_2 represents the tangential velocity of the output along the path. Similarly, since ξ_1^j represents the transversal positions to the path, ξ_2^j represents the transversal velocities, $j \in \{1, \dots, p-1\}$. Note that there will always be an internal state ζ, since by Assumption 2.1, $N + \delta_b + \delta_s \geq N + \delta_m \geq p$ and

$\delta_b + \delta_s \in \{2, 3\}$ always implies $2N + 3 + \delta_b + \delta_s - 2p > 0$, which is expected as mobile manipulators are highly redundant systems.

The dynamics in the transformed coordinates are

$$\begin{aligned}
&\dot{\eta}_1 = \eta_2 && \dot{\xi}_1^j = \xi_2^j \\
&\dot{\eta_2} = L_f^2\eta_1(x) + L_g L_f \eta_1(x)u && \dot{\xi}_2^j = L_f^2\xi_1^j(x) + L_g L_f \xi_1^j(x)u \\
&\dot{\zeta} = L_f\varphi(x) + L_g\varphi(x)u =: f_\zeta(x, u) &&
\end{aligned} \tag{22}$$

for $j \in \{1, \ldots, p-1\}$ and $x = T^{-1}(\eta, \xi, \zeta)$. The expressions for the Lie derivatives can be found in [7]. Based on the virtual output construction (Sect. 4), the η-subsystem represents the dynamics tangent to the path, and the ξ-subsystem represents the dynamics transversal to the path. Thus, the dynamics f_ζ represent the dynamics of the system (12) that do not produce any output motion. These are the redundant dynamics to path following. Next we perform standard partial feedback linearization for non-square systems Let

$$\alpha(x) := \left[L_f^2\eta_1(x)\ L_f^2\xi_1^1(x)\ \ldots\ L_f^2\xi_1^{p-1}(x)\right]^\top \in \mathbb{R}^p \tag{23}$$

$$\beta(x) := \begin{bmatrix} L_g L_f \eta_1(x) \\ L_g L_f \xi_1^1(x) \\ \vdots \\ L_g L_f \xi_1^{p-1}(x) \end{bmatrix} \in \mathbb{R}^{p\times(N+\delta_b+\delta_s)} \tag{24}$$

where the terms in $\beta(x)$ are

$$L_g L_f \eta_1(x) = \mathbf{e}_1(\lambda^*)^\top \frac{\partial H(x_c)}{\partial x_c} \frac{\partial F_c(x)}{\partial x_v} G_v(x_c) \tag{25}$$

$$\begin{aligned}
L_g L_f \xi_1^{j-1}(x) =& \mathbf{e}_j(\lambda^*)^\top \frac{\partial H(x_c)}{\partial x_c} \frac{\partial F_c(x)}{\partial x_v} G_v(x) + \\
& (H(x_c) - \sigma(\lambda^*))^\top \mathbf{e}_j'(\lambda^*) \frac{L_g L_f \eta_1(x)}{\|\sigma'(\lambda^*)\|}
\end{aligned} \tag{26}$$

for $j \in \{2, \ldots, p\}$ and $\lambda^* = \varpi \circ H(x_c)$ [7]. The decoupling matrix $\beta(x)$ has full row rank p in the set $\mathscr{U}$ (see (18)) since each $\mathbf{e}_i$ are orthogonal by FSF construction, thus in U the virtual output has a well-defined relative degree [13].

Remark 1 A necessary condition for $\beta(x)$ to have rank p is that each matrix $\frac{\partial H}{\partial x_c}$, $\frac{\partial F_c}{\partial x_v}$ and $G_v(x)$ have rank of at least p. The matrix $\frac{\partial H(x_c)}{\partial x_c}$ has the form

$$\frac{\partial H(x_c)}{\partial x_c} = \left[\begin{bmatrix} R(\theta)_{1:2,1:2} & 0_{2\times(P-2)} \\ 0_{(p-2)\times 2} & I_{(p-2)\times(P-2)} \end{bmatrix} J(q) \left| \begin{matrix} I_{2\times 2} \\ 0_{(p-2)\times 2} \end{matrix} \right| \begin{bmatrix} R'(\theta)_{1:2,1:2} & 0_{2\times(P-2)} \\ 0_{(p-2)\times 2} & 0_{(p-2)\times(P-2)} \end{bmatrix} h(q) \right] \tag{27}$$

and always has rank of at least 2 due to the ground vehicle (thus is always full rank if $p = 2$). For $p > 2$, it has full rank if $J(q)$ is non-singular.

The matrix $\frac{\partial F_c(x)}{\partial x_v}$ has the form

$$\frac{\partial F_c}{\partial x_v} = \begin{bmatrix} I_{N\times N} & 0_{N\times\delta_b} & 0_{N\times\delta_s} \\ 0_{3\times N} & R(\theta)\Sigma(\sigma) & R(\theta)\frac{\partial\Sigma(\sigma)}{\partial\sigma} \end{bmatrix}, \tag{28}$$

thus $N + 3 \geq \text{rank}(\frac{\partial F_c}{\partial x_v}) \geq N + \delta_b \geq p$ (by Assumption 2.1) and will depend on the particular ground vehicle. For a unicycle (Sect. 5), for all x, $\text{rank}(\frac{\partial F_c(x)}{\partial x_v}) = N + \delta_b = N + 2$. The matrix $G_v(x)$ (see (9)) always has full rank of $N + \delta_b + \delta_s \geq p$. ♦

Next, introduce an auxiliary input $v := (v_\eta, v_\xi) \in \mathbb{R} \times \mathbb{R}^{p-1}$ such that

$$v = \beta(x)u + \alpha(x). \tag{29}$$

The dynamics (22) then become

$$\dot{\zeta} = f_\zeta(x, u) \qquad \begin{aligned} \dot{\eta}_1 &= \eta_2 \\ \dot{\eta}_2 &= v_\eta \end{aligned} \qquad \begin{aligned} \dot{\xi}_1^j &= \xi_2^j \\ \dot{\xi}_2^j &= v_{\xi^j} \end{aligned}$$

where $j \in \{1, \ldots, p-1\}$ and $x = T^{-1}(\eta, \xi, \zeta)$.

In the set $\mathscr{U}$ the ξ-subsystem is linear and controllable, and can be stabilized to ensure attractiveness and invariance of the path $\mathscr{P}$ [11], thereby satisfying **PF1**. The η-subsystem is also linear and controllable, thus a tangential controller can be designed for v_η to track a desired tangential position or velocity profile $\eta^{\text{ref}}(t)$, thereby satisfying **PF2**. Figure 2 shows a block diagram of the entire control system.

Redundancy Resolution: Once the auxiliary control signal $v \in \mathbb{R}^p$ is generated to stabilize the η and ξ-subsystems for **PF1** and **PF2**, the feedback transformation (29) must be solved to generate the actual control signal $u \in \mathbb{R}^{N+\delta_b+\delta_s}$. When $N + \delta_b + \delta_s = p$, there is a unique solution to (29), and one must ensure that the internal dynamics $\dot{\zeta} = f_\zeta(x, u)$ remain bounded.

For most mobile manipulator systems (see Sect. 5 for an example), $N + \delta_b + \delta_s > p$. Thus there is some freedom in the choice of the control input u under the feedback transform (29). This freedom can be used to enforce boundedness of the internal dynamics. We apply the approach from [9] to mobile manipulator systems. Consider the static optimization

$$\begin{aligned} &\min_u \,(u - r(x))^\top W (u - r(x)) \\ &\text{s.t.} \quad v = \beta(x)u + \alpha(x) \end{aligned} \tag{30}$$

This is a static minimization of a quadratic function of u under a linear constraint, for which a closed form solution for u can be found using Lagrange multipliers. One might be tempted to use an inequality constraint over x to respect joint limits, but it is unclear how to do so since we are optimizing over the control effort u and x

is treated as a parameter. The matrix $W \in \mathbb{R}^{(N+\delta_b+\delta_s)\times(N+\delta_b+\delta_s)}$ is a positive-definite weighting matrix, and the function $r : \mathbb{R}^{N+3} \to \mathbb{R}^{N+\delta_b+\delta_s}$ is used to bias the control input u to achieve desired behaviour in the internal dynamics. For example, if W is the identity matrix and $r(x)$ is the zero function, then we are minimizing control effort while achieving **PF1** and **PF2**.

In this application, we want to bias u such that the manipulator stays away from joint and actuation limits (**PF3**). If a joint $q_i, i \in \{1, \dots, N\}$ of the manipulator is at its minimum limit $q_i^{\min}$, setting the control effort $u_i, i \in \{1, \dots, N\}$ (i.e., u_m) corresponding to this joint to be the maximum control effort $u_i^{\max}$ will *likely* increase the value of the joint, thereby pushing it away from the negative joint limit. These joint limits can be set artificially to bias the manipulator in a preferred position (so as to avoid singular configurations of $J(q)$, see Remark 1) or set to their true joint limit values. For $u_i, i \in \{N+1, \dots, N+\delta_b+\delta_s\}$ (i.e., u_b, γ_s) we'd like to minimize the control effort so that the mobile base moves as little as possible (however if there are steering limits on σ, a similar logic used for joint limits on a manipulator can be applied). The corresponding r function to achieve this behaviour in (30) is

$$r(x)_i := -\frac{u_i^{\max} - u_i^{\min}}{q_i^{\max} - q_i^{\min}}(q_i - q_i^{\min}) + u_i^{\max}, \quad i \in \{1, \dots, N\} \tag{31}$$

$$r(x)_i := 0, \quad i \in \{N+1, N+\delta_b+\delta_s\} \tag{32}$$

Using Lagrangian multipliers, the solution to (30) is

$$u = \beta^\dagger(x)(v - \alpha(x)) + \left(I_{n\times n} - \beta^\dagger(x)\beta(x)\right) r(x) \tag{33}$$

where $\beta^\dagger(x) := W^{-1}\beta(x)^\top \left(\beta(x)W^{-1}\beta(x)^\top\right)^{-1}$.

In [9, Conjecture 3.5], we showed via examples and experiments on manipulators that a similar control law (33) seems to provide boundedness of the internal dynamics while maintaining joint limits of a manipulator when each degree of freedom has inherent viscous friction. In this paper, for mobile manipulator systems, boundedness of internal dynamics and maintaining joint limits of the manipulator holds when $\rho < 0$ in the dynamic extension (8) (**PF3**). If $\rho = 0$, **PF1** and **PF2** can still be achieved (i.e. controllability is unchanged), but the internal dynamics change and at least one of the base position states (x^b, θ) may become unbounded, see Example. 1. The reason is u_b controls the mobile base acceleration through (8). So although (31) is chosen such that base acceleration is minimized, ideally keeping $u_b = 0$, then the base velocities remain at a some constant, not necessarily zero, value, resulting in base states (namely θ) that continuously grow. Thus, $\rho < 0$ is used to dampen these dynamics. The same doesn't happen for the steering dynamics of σ since the steering dynamics are governed by a single integrator. So γ_s is minimized, ideally 0 when possible, implies the steering position is constant when possible, thus remaining bounded.

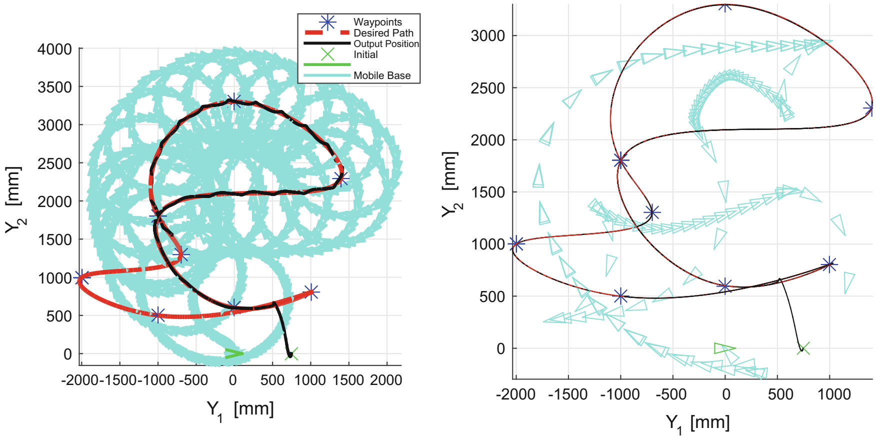


Fig. 4 Example 1: Simulation of $\rho = 0$ (*left*) and $\rho = -10$ (*right*). When $\rho = 0$, the mobile base orbits the output position $y = H(x_c(t))$ such that it maintains the preferred position and is on the desired path $\mathscr{P}$. Each snapshot of the mobile base is taken every 1.5 s

Example 1 (*Internal Dynamics*) Consider the mobile manipulator system of Sect. 5. In this system and in most systems, $u_i^{\max} = -u_i^{\min}$ for use in (31). The joint limits are set to their true values except for the waist joint q_1 where $q_1^{\min} := 0°$, $q_1^{\max} := 20°$. Thus the optimal control effort for the waist $r(x)_1 = 0$ when $q_1 = 10°$, so that the desired position of the waist is 10°, that is, the manipulator should face to the right of the vehicle (or the vehicle is to stay towards the left of the desired path). The path is shown in Fig. 4, and $\eta_2^{\text{ref}} = 100\,\text{mm/s}$. Two simulation results are included that show **PF1** and **PF2** are achieved, and there is automatic coordination between the manipulator and the mobile base. When $\rho = 0$, **PF3** is not achieved as the yaw state θ of the mobile base continuously increases and the vehicle orbits the end-effector as the end-effector follows the path.

When $\rho < 0$, **PF3** is achieved as discussed above. At the very start of the run, the base automatically orients itself by rotating, travels backwards until the end-effector is at its preferred position while the end effector is on the path $\mathscr{P}$, then proceeds forward as it achieves **PF2**. Note that the initial position of the base could have been placed further away from the path, and the same controller will automatically move the base to a neighbourhood of the path. Further notice the controller automatically speeds up the mobile base when the end-effector is traversing the higher curvature areas of the path in order to maintain the desired constant tangential velocity profile. ▲

Remark 2 The weighting matrix W plays an important role to ensure the joint limits are satisfied. If the associated weights for the manipulator joint actuation are too low, the controller may not move the mobile base at all before the manipulator reaches a singular configuration. If the weights are high enough, then the controller will increase control effort to the base so that the base moves while the manipulator satisfies its joint limits. ♦

5 Experiment

System Model: The mobile manipulator system is a platform designed by Clearpath Robotics (see Fig. 5). The manipulator is a 4-degree-of-freedom system. The complete modelling procedure can be found in [8], yields the dynamics of the manipulator in form (1) with $N = 4$ and u_m representing the vector of control input voltages to the motors. The mobile platform runs a low level control loop that controls the vehicle's heading rate and tangential velocity. The vehicle has no steering wheels. Thus, for this system $\delta_s = 0$ and $\delta_b = 2$ and the general kinematic model (3) reduces to the standard unicycle equations [5]

$$\dot{x}_1^b = \gamma_{b_1} \cos(\theta) \tag{34}$$

$$\dot{x}_2^b = \gamma_{b_1} \sin(\theta) \tag{35}$$

$$\dot{\theta} = \gamma_{b_2} \tag{36}$$

The output space is taken to be the 3-dimensional Euclidean space, that is $p = 3$.

Path Following Controller: The desired path is generated by spline-interpolating 3-dimensional waypoints using quintic polynomials [6]. The control design approach then follows from Sect. 4. For this mobile manipulator system, significant modelling uncertainties arise due to inaccurate manipulator modelling [8] and the assumption of a perfect kinematic model of the mobile base, as well as ignoring the coupling dynamics between the two. This results in imperfect cancellation of the actual system dynamics using (23), (24). The Lyapunov redesign based robust controller in [8, Eq. (27)] is used to overcome the modelling uncertainties, thereby achieving **PF1**.

The η-subsystem is also linear and controllable, thus a tangential controller can be designed for v_η to track some desired tangential position or velocity profile $\eta^{\text{ref}}(t)$. Our goal is to track a desired constant velocity profile η_2^{ref}. This can be done using a PI controller [8, Eq. (24)] where the integral action is used for robustness, thereby achieving **PF2**. Figure 2 has the complete block diagram.

The manipulator has a Labview Real-Time Module® which is used to read the linear actuator distances using optical encoders and to control the motor PWM amplifiers. The encoder readings are converted to joint angles q and numerically differentiated to approximate $\dot{q}$. This module communicates with ROS via the Rosbridge package. The Husky is a ROS-enabled robot which takes in γ_b to control the robot, and gives out (x^b, θ) data based on a sensor fusion of wheel odometry and an onboard IMU. The path following controller is implemented using MATLAB Robotics System Toolbox.

Results: The joint limits in (31) are set so that in Figs. 6, 7, 8 and 9, the waist of the manipulator prefers 90° (the manipulator is ahead of the vehicle), and again in another run in Fig. 10, so that the waist prefers 160° (the manipulator is to the left of the vehicle). In all cases, $\eta_2^{\text{ref}} = 100\,\text{mm/s}$.

As shown in the 3D and 2D plots, the proposed path following controller successfully satisfies the goals **PF1** and **PF3**. The output y automatically goes towards the closest point on the path $\mathscr{P}$ due to the coordinate transformation employed. A key advantage of the proposed approach is the automatic coordination of the manipulator and mobile base. Based on the preferred position of the manipulator via (31), the mobile base automatically orients itself. In the second lap of the path (indicated in magenta), the mobile base actually moves backwards until $x^b \approx$ (-1000, 500) mm, at which point the base turns and moves forward in order to keep the manipulator at its preferred position and to traverse the path at the desired rate η_2^{ref}, without any explicit trajectory planning and tracking for the mobile base.

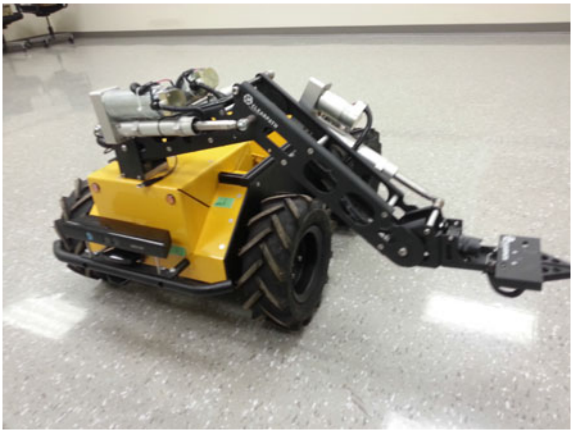


Fig. 5 Clearpath manipulator mounted on a clearpath A200 mobile platform

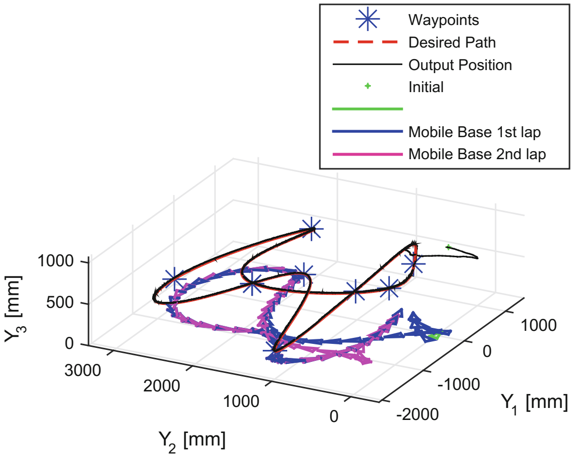


Fig. 6 Experiment 3D response – preferred $q_1 = 90°$

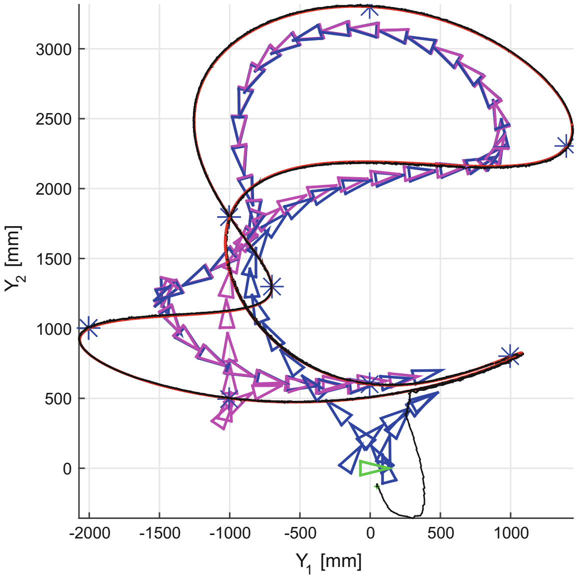


Fig. 7 Experiment response, *top* view – preferred $q_1 = 90°$. Each snapshot of the mobile base is taken every 4 s

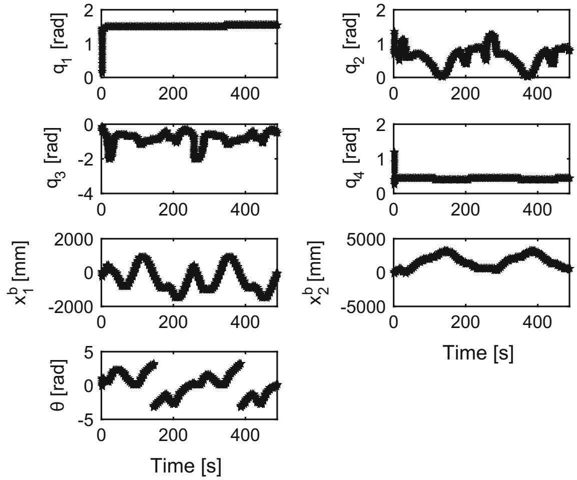


Fig. 8 Preferred $q_1 = 90°$. State position x_c trajectories

We see that in the state trajectory (Fig. 8), the manipulator positions q quickly converge to their preferred positions when possible, in particular, the waist q_1 approaches 90°. The virtual output trajectories can be found in Fig. 9 and show that the system is traversing the path at a constant (desired) rate (**PF2**), while the transversal errors quickly approach 0 resulting in convergence to the desired path $\mathscr{P}$ (**PF1**). At steady state, the cross-track errors are less than ≈ 15 mm.

When the preferred waist position is adjusted, the mobile base automatically takes another route for the same desired path $\mathscr{P}$. In Fig. 10, it can be seen that the mobile

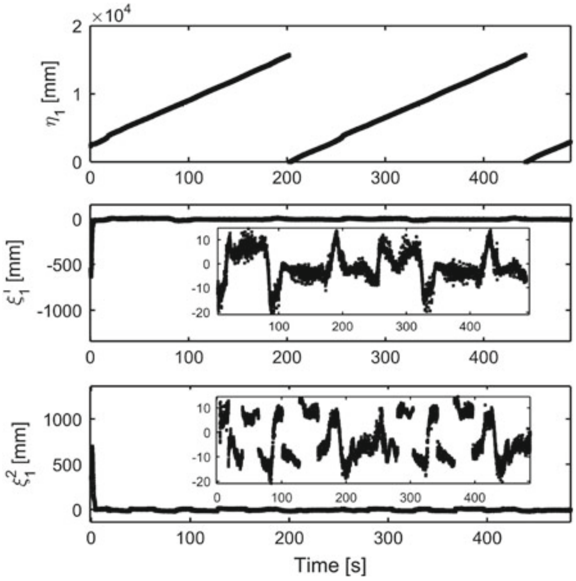


Fig. 9 Preferred $q_1 = 90°$. Virtual output trajectories. Notice from η_1 just over 2 laps are completed

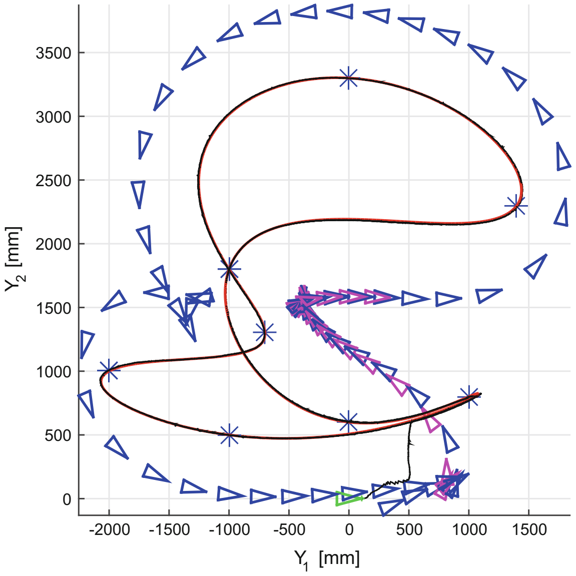


Fig. 10 Experiment response, *top* view – preferred $q_1 = 160°$. Each snapshot of the mobile base is taken every 4 s

base tries to stay to the right of the path in order to respect the artificial preferred position of the waist of the manipulator. Note that at the high curvature areas, the mobile base speeds up significantly (apparent by the decreased density of the mobile base snapshots) in order to respect **PF2**.

6 Conclusions and Future Work

This paper proposes a unified path following controller for mobile manipulator systems. The controller automatically moves the mobile base and the manipulator such that the end-effector traverses a path in the output space towards a given desired position or with a given velocity profile. There is no explicit trajectory required for the mobile base or the end-effector to follow. The desired path is rendered invariant and attractive, and the redundancy resolution scheme employed allows for the manipulator to stay away from joint limits. Dynamic extension with damping was used so that the virtual output employed for path following has a full relative degree and to ensure boundedness of the internal dynamics.

Dynamically changing the r function in (31) for real-time obstacle avoidance is a direction for future work. Analyzing when the virtual output constructed in the mobile manipulator path following controller loses full relative degree (that is when (24) loses rank) is another direction for future work. This will help determine which configurations of the mobile manipulator with respect to the path should be avoided.

References

1. Aguiar, A., Hespanha, J., Kokotovic, P.: Path-following for nonminimum phase systems removes performance limitations. IEEE Trans. Autom. Control **50**(2), 234–239 (2005)
2. Akhtar, A., Nielsen, C., Waslander, S.: Path following using dynamic transverse feedback linearization for car-like robots. IEEE Trans. Robot. **31**(2), 269–279 (2015)
3. Cameron, J., MacKenzie, D., Ward, K., Arkin, R., Book, W.: Reactive control for mobile manipulation. In: IEEE ICRA, vol. 3, pp. 228–235 (1993)
4. Chung, J.H., Velinsky, S.: Modeling and control of a mobile manipulator. Robotica **16**(06), 607–613 (1998)
5. De Wit, C.C., Bastin, G., Siciliano, B.: Theory of robot control. Springer, New York (1996)
6. Erkorkmaz, K., Altintas, Y.: High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. Int. J. Mach. Tools Manuf. **41**(9), 1323–1345 (2001)
7. Gill, R.: Robust spline path following for redundant mechanical systems. Master's thesis, University of Waterloo (2015)
8. Gill, R., D Kulić, D., Nielsen, C.: Robust path following for robot manipulators. In: IEEE IROS, pp. 3412–3418 (2013)
9. Gill, R., Kulic, D., Nielsen, C.: Spline path following for redundant mechanical systems. IEEE Trans. Robot. **31**(6), 1378–1392 (2015)
10. Hauser, J., Hindman, R.: Maneuver regulation from trajectory tracking: feedback linearizable systems. In: Proceedings of IFAC Symposium Nonlinear Control Systems Design, pp. 595–600 (1995)
11. Hladio, A., Nielsen, C., Wang, D.: Path following for a class of mechanical systems. IEEE Trans. Control Syst. Technol. **21**(6), 2380–2390 (2013)
12. Inoue, F., Muralami, T., Ihnishi, K.: A motion control of mobile manipulator with external force. IEEE/ASME Trans. Mech. **6**(2), 137–142 (2001)
13. Isidori, A.: Nonlinear Control Systems, 3rd edn. Springer, New York (1995)
14. Khatib, O.: A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE J. Robot. Autom. **3**(1), 43–53 (1987)

15. Khatib, O.: Mobile manipulation: the robotic assistant. Robot. Auton. Syst. **26**(23), 175–183 (1999)
16. Kreyszig, E.: Differential Geometry. Dover Publications, New York (1991)
17. Lapierre, L., Soetanto, D.: Nonlinear path-following control of an auv. Ocean Eng. **34**(11), 1734–1744 (2007)
18. Nagatani, K., Yuta, S.: Designing strategy and implementation of mobile manipulator control system for opening door. In: IEEE ICRA, vol. 3, pp. 2828–2834 (1996)
19. Nielsen, C., Maggiore, M.: Output stabilization and maneuver regulation: a geometric approach. Syst. Control Lett. **55**(5), 418–427 (2006)
20. Pennock, G., Kassner, D.: The workspace of a general geometry planar three-degree-of-freedom platform-type manipulator. J. Mech. Des. **115**(2), 269–276 (1993)
21. Peterson, L., Austin, D., Kragic, D.: High-level control of a mobile manipulator for door opening. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2333–2338 (2000). doi:10.1109/IROS.2000.895316
22. Seraji, H.: A unified approach to motion control of mobile manipulators. Int. J. Robot. Res. **17**(2), 107–118 (1998)
23. Spong, M., Hutchinson, S., Vidyasagar, M.: Robot Modeling and Control. Wiley, New York (2006)
24. Xu, Y., Brown, J.H.B., Friedman, M., Kanade, T.: Control system of the self-mobile space manipulator. IEEE Trans. Control Syst. Technol. **2**(3), 207–219 (1994)
25. Yamamoto, Y., Yun, X.: Coordinating locomotion and manipulation of a mobile manipulator. In: Proceedings of IEEE Conference on Decision and Control, pp. 2643–2648 (1992)

Incremental Sparse GP Regression for Continuous-Time Trajectory Estimation and Mapping

Xinyan Yan, Vadim Indelman and Byron Boots

1 Introduction and Related Work

Simultaneously recovering the location of a robot and a map of its environment from sensor readings is a fundamental challenge in robotics [1, 7, 14]. Well-known approaches to this problem, such as square root smoothing and mapping (SAM) [4], have focused on regression-based methods that exploit the sparse structure of the problem to efficiently compute a solution. The main weakness of the original SAM algorithm was that it was a *batch* method: all of the data must be collected before a solution can be found. For a robot traversing an environment, the inability to update an estimate of its trajectory online is a significant drawback. In response to this weakness, Kaess et al. [9] developed a critical extension to the batch SAM algorithm, iSAM, that overcomes this problem by *incrementally* computing a solution. The main drawback of iSAM, was that the approach required costly periodic batch steps for variable reordering to maintain sparsity and relinearization. This approach was extended in iSAM 2.0 [11], which employs an efficient data structure called the *Bayes tree* [10] to perform incremental variable reordering and just-in-time relinearization, thereby eliminating the bottleneck caused by batch variable reordering and relinearization. The iSAM 2.0 algorithm and its extensions are widely considered to be state-of-the-art in robot trajectory estimation and mapping.

The majority of previous approaches to trajectory estimation and mapping, including the smoothing-based SAM family of algorithms, have formulated the problem

X. Yan (✉) · B. Boots
College of Computing, Georgia Institute of Technology, Atlanta, Ga, USA
e-mail: xinyan.yan@cc.gatech.edu

B. Boots
e-mail: bboots@cc.gatech.edu

V. Indelman
Department of Aerospace Engineering, Technion - Israel Institute of Technology,
Haifa, Israel
e-mail: vadim.indelman@technion.ac.il

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_31

in discrete time [1, 3, 4, 7, 11, 12, 14]. However, discrete-time representations are restrictive: they are not easily extended to trajectories with irregularly spaced waypoints or asynchronously sampled measurements. A continuous-time formulation of the SAM problem where measurements constrain the trajectory at any point in time, would elegantly contend with these difficulties. Viewed from this perspective, the robot trajectory is a *function* $\mathbf{x}(t)$, that maps any time t to a robot state. The problem of estimating this function along with landmark locations has been dubbed *simultaneous trajectory estimation and mapping* (STEAM) [2].

Tong et al. [15] proposed a Gaussian process (GP) regression approach to solving the STEAM problem. While their approach was able to accurately model and interpolate asynchronous data to recover a trajectory and landmark estimate, it suffered from significant computational challenges: naive Gaussian process approaches to regression have notoriously high space and time complexity. Additionally, Tong et al.'s approach is a *batch* method, so updating the solution necessitates saving all of the data and completely resolving the problem. In order to combat the computational burden, Tong et al.'s approach was extended in Barfoot et al. [2] to take advantage of the sparse structure inherent in the STEAM problem. The resulting algorithm significantly speeds up solution time and can be viewed as a continuous-time analog of Dellaert's original square-root SAM algorithm [4]. Unfortunately, like SAM, Barfoot et al.'s GP-based algorithm remains a batch algorithm, which is a disadvantage for robots that need to continually update the estimate of their trajectory and environment.

In this work, we provide the critical extensions necessary to transform the existing Gaussian process-based approach to solving the STEAM problem into an extremely efficient incremental approach. Our algorithm combines the benefits of Gaussian processes and iSAM 2.0. Like the GP regression approaches to STEAM, our approach can model continuous trajectories, handle asynchronous measurements, and naturally interpolate states to speed up computation and reduce storage requirements, and, like iSAM 2.0, our approach uses a Bayes tree to efficiently calculate a *maximum a posteriori* (MAP) estimate of the GP trajectory while performing incremental factorization, variable reordering, and just-in-time relinearization. The result is an online GP-based solution to the STEAM problem that remains computationally efficient while scaling up to large datasets.

2 Batch Trajectory Estimation and Mapping as Gaussian Process Regression

We begin by describing how the simultaneous trajectory estimation and mapping (STEAM) problem can be formulated in terms of Gaussian process regression. Following Tong et al. [15] and Barfoot et al. [2], we represent robot trajectories as functions of time t sampled from a Gaussian process:

$$\mathbf{x}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t')), \quad t_0 < t, t' \tag{1}$$

Here, $\mathbf{x}(t)$ is the continuous-time trajectory of the robot through state-space, represented by a Gaussian process with mean $\boldsymbol{\mu}(t)$ and covariance $\boldsymbol{\mathcal{K}}(t, t')$ functions.

We next define a finite set of measurements:

$$\mathbf{y}_i = \mathbf{h}_i(\boldsymbol{\theta}_i) + \mathbf{n}_i, \quad \mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i), \quad i = 1, 2, \ldots, N \tag{2}$$

The measurement $\mathbf{y}_i$ can be any linear or nonlinear functions of a set of related variables $\boldsymbol{\theta}_i$ plus some Gaussian noise $\mathbf{n}_i$. The related variables for a range measurement are the robot state at the corresponding measurement time $\mathbf{x}(t_i)$ and the associated landmark location $\boldsymbol{\ell}_j$. We assume the total number of measurements is N, and the number of trajectory states at measurement times is M.

Based on the definition of Gaussian processes, any finite collection of robot states has a joint Gaussian distribution [13]. So the robot states at measurement times are normally distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\mathcal{K}}$.

$$\begin{aligned} &\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\mathcal{K}}), \quad \mathbf{x} = [\, \mathbf{x}(t_1)^\intercal \ldots \mathbf{x}(t_M)^\intercal \,]^\intercal \\ &\boldsymbol{\mu} = [\, \boldsymbol{\mu}(t_1)^\intercal \ldots \boldsymbol{\mu}(t_M)^\intercal \,]^\intercal, \quad \boldsymbol{\mathcal{K}}_{ij} = \boldsymbol{\mathcal{K}}(t_i, t_j) \end{aligned} \tag{3}$$

Note that any point along the continuous-time trajectory can be estimated from the Gaussian process model. Therefore, the trajectory does not need to be discretized and robot trajectory states do not need to be evenly spaced in time, which is an advantage of the Gaussian process approach over discrete-time approaches (e.g. Dellaert's square-root SAM [4]).

The landmarks $\boldsymbol{\ell}$ which represent the map are assumed to conform to a joint Gaussian distribution with mean $\mathbf{d}$ and covariance $\mathbf{W}$ (Eq. 4). The prior distribution of the combined state $\boldsymbol{\theta}$ that consists of robot trajectory states at measurement times and landmarks is, therefore, a joint Gaussian distribution (Eq. 5).

$$\boldsymbol{\ell} \sim \mathcal{N}(\mathbf{d}, \mathbf{W}), \quad \boldsymbol{\ell} = [\, \boldsymbol{\ell}_1^\intercal \; \boldsymbol{\ell}_2^\intercal \ldots \boldsymbol{\ell}_O^\intercal \,]^\intercal \tag{4}$$

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\eta}, \boldsymbol{\mathcal{P}}), \quad \boldsymbol{\eta} = [\, \boldsymbol{\mu}^\intercal \; \mathbf{d}^\intercal \,]^\intercal, \quad \boldsymbol{\mathcal{P}} = \operatorname{diag}(\boldsymbol{\mathcal{K}}, \mathbf{W}) \tag{5}$$

To solve the STEAM problem, given the prior distribution of the combined state and the likelihood of measurements, we compute the *maximum a posteriori* (MAP) estimate of the combined state *conditioned* on measurements via Bayes' rule:

$$\begin{aligned} \boldsymbol{\theta}^* \triangleq \boldsymbol{\theta}_{MAP} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \; p(\boldsymbol{\theta}|\mathbf{y}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \; \frac{p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})}{p(\mathbf{y})} \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \; p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left(-\log p(\boldsymbol{\theta}) - \log p(\mathbf{y}|\boldsymbol{\theta})\right) \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left(\|\boldsymbol{\theta} - \boldsymbol{\eta}\|^2_{\boldsymbol{\mathcal{P}}} + \|\mathbf{h}(\boldsymbol{\theta}) - \mathbf{y}\|^2_{\mathbf{R}}\right) \end{aligned} \tag{6}$$

where the norms are Mahalanobis norms defined as: $\|\mathbf{e}\|^2_{\boldsymbol{\Sigma}} = \mathbf{e}^\intercal \boldsymbol{\Sigma}^{-1}\mathbf{e}$, and $\mathbf{h}(\boldsymbol{\theta})$ and $\mathbf{R}$ are the mean and covariance of the measurements collected, respectively:

$$\mathbf{h}(\boldsymbol{\theta}) = [\,\mathbf{h}_1(\boldsymbol{\theta}_1)^\intercal\ \mathbf{h}_2(\boldsymbol{\theta}_2)^\intercal \ldots \mathbf{h}_N(\boldsymbol{\theta}_N)^\intercal\,]^\intercal, \quad \mathbf{R} = \mathrm{diag}(\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N) \tag{7}$$

Because both covariance matrices $\boldsymbol{\mathcal{P}}$ and $\mathbf{R}$ are positive definite, the objective in Eq. 6 corresponds to a least squares problem. Consequently, if some of the measurement functions $\mathbf{h}_i(\cdot)$ are nonlinear, this becomes a nonlinear least squares problem, in which case iterative methods including Gauss–Newton and Levenberg–Marquardt [5] can be utilized; in each iteration, an optimal update is computed given a linearized problem at the current estimate. A linearization of a measurement function at current state estimate $\bar{\boldsymbol{\theta}}_i$ can be accomplished by a first-order Taylor expansion:

$$\mathbf{h}_i\left(\bar{\boldsymbol{\theta}}_i + \delta\boldsymbol{\theta}_i\right) \approx \mathbf{h}_i(\bar{\boldsymbol{\theta}}_i) + \left.\frac{\partial \mathbf{h}_i}{\partial \boldsymbol{\theta}_i}\right|_{\bar{\boldsymbol{\theta}}_i} \delta\boldsymbol{\theta}_i \tag{8}$$

Combining Eq. 8 with Eq. 6, the optimal increment $\delta\boldsymbol{\theta}^*$ is:

$$\delta\boldsymbol{\theta}^* = \underset{\delta\boldsymbol{\theta}}{\mathrm{argmin}}\ \left(\|\bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta} - \boldsymbol{\eta}\|^2_{\boldsymbol{\mathcal{P}}} + \|\mathbf{h}(\bar{\boldsymbol{\theta}}) + \mathbf{H}\delta\boldsymbol{\theta} - \mathbf{y}\|^2_{\mathbf{R}}\right) \tag{9}$$

$$\mathbf{H} = \mathrm{diag}(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_N), \qquad \mathbf{H}_i = \left.\frac{\partial \mathbf{h}_i}{\partial \boldsymbol{\theta}_i}\right|_{\bar{\boldsymbol{\theta}}_i} \tag{10}$$

where $\mathbf{H}$ is the measurement Jacobian matrix. To solve the linear least squares problem in Eq. 9, we take the derivative with respect to $\delta\boldsymbol{\theta}$, and set it to zero, which gives us $\delta\boldsymbol{\theta}^*$ embedded in a set of linear equations

$$\underbrace{(\boldsymbol{\mathcal{P}}^{-1} + \mathbf{H}^\intercal \mathbf{R}^{-1}\mathbf{H})}_{\boldsymbol{\mathcal{I}}}\,\delta\boldsymbol{\theta}^* = \underbrace{\boldsymbol{\mathcal{P}}^{-1}(\boldsymbol{\eta} - \bar{\boldsymbol{\theta}}) + \mathbf{H}^\intercal \mathbf{R}^{-1}(\mathbf{y} - \bar{\mathbf{h}})}_{\mathbf{b}} \tag{11}$$

with covariance $\mathrm{cov}(\delta\boldsymbol{\theta}^*, \delta\boldsymbol{\theta}^*) = \boldsymbol{\mathcal{I}}^{-1}$.

The positive definite matrix $\boldsymbol{\mathcal{I}}$ is the a posteriori information matrix. To solve the linear equations for $\delta\boldsymbol{\theta}^*$, factorization-based methods can provide a fast, numerically stable solution. For example, $\delta\boldsymbol{\theta}^*$ can be found by first performing a Cholesky factorization $\boldsymbol{\mathcal{L}}\boldsymbol{\mathcal{L}}^\intercal = \boldsymbol{\mathcal{I}}$, and then solving by back substitution. At each iteration we perform a *batch* state estimation update $\bar{\boldsymbol{\theta}} \leftarrow \bar{\boldsymbol{\theta}} + \delta\boldsymbol{\theta}^*$ and repeat the process until convergence. If $\boldsymbol{\mathcal{I}}$ is dense, the time complexity of a Cholesky factorization and back substitution are $O(n^3)$ and $O(n^2)$ respectively, where $\boldsymbol{\mathcal{I}} \in \mathbb{R}^{n\times n}$ [8]. However, if $\boldsymbol{\mathcal{I}}$ has sparse structure, then the solution can be found much faster. For example, for a narrowly banded matrix, the computation time is $O(n)$ instead of $O(n^3)$ [8]. Fortunately, we can guarantee sparsity for the STEAM problem (see Sect. 2.2).

2.1 State Interpolation

An advantage of the Gaussian process representation of the robot trajectory is that any trajectory state can be interpolated from other states by computing the posterior mean [15]:

$$\bar{\mathbf{x}}(t) = \boldsymbol{\mu}(t) + \boldsymbol{\mathcal{K}}(t)\boldsymbol{\mathcal{K}}^{-1}(\bar{\mathbf{x}} - \mu) \tag{12}$$

$$\bar{\mathbf{x}} = [\,\bar{\mathbf{x}}(t_1)^{\intercal} \ldots \bar{\mathbf{x}}(t_M)^{\intercal}\,]^{\intercal}, \quad \boldsymbol{\mathcal{K}}(t) = [\,\boldsymbol{\mathcal{K}}(t, t_1) \ldots \boldsymbol{\mathcal{K}}(t, t_M)\,]$$

By utilizing interpolation, we can reduce the number of robot trajectory states that we need to estimate in the optimization procedure [15]. For simplicity, assume $\boldsymbol{\theta}_i$, the set of the related variables of the ith measurement according to the model (Eq. 2), is $\mathbf{x}(\tau)$. Then, after interpolation, Eq. 8 becomes:

$$\begin{aligned} \mathbf{h}_i\left(\bar{\boldsymbol{\theta}}_i + \delta\boldsymbol{\theta}_i\right) &= \mathbf{h}_i\left(\bar{\mathbf{x}}(\tau) + \delta\mathbf{x}(\tau)\right) \\ &\approx \mathbf{h}_i(\bar{\mathbf{x}}(\tau)) + \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}(\tau)} \cdot \left.\frac{\partial \mathbf{x}(\tau)}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}} \delta\mathbf{x} \\ &= \mathbf{h}_i\left(\boldsymbol{\mu}(\tau) + \boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1}(\bar{\mathbf{x}} - \mu)\right) + \mathbf{H}_i\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1}\delta\mathbf{x} \end{aligned} \tag{13}$$

By employing Eq. 13 during optimization, we can make use of measurement i without explicitly estimating the trajectory states that it relates to. We exploit this advantage to greatly speed up the solution to the STEAM problem in practice (Sect. 4).

2.2 Sparse Gaussian Process Regression

The efficiency of the Gaussian process Gauss–Newton algorithm presented in Sect. 2 is dependent on the choice of kernel. It is well-known that if the information matrix $\boldsymbol{\mathcal{I}}$ is sparse, then it is possible to very efficiently compute the solution to Eq. 11 [4]. Barfoot et al. suggest a kernel matrix with a sparse inverse that is well-suited to the simultaneous trajectory estimation and mapping problem [2]. In particular, Barfoot et al. show that $\boldsymbol{\mathcal{K}}^{-1}$ is exactly block-tridiagonal when the GP is assumed to be generated by linear, time-varying (LTV) stochastic differential equation (SDE) which we describe here:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(t)\mathbf{x}(t) + \mathbf{v}(t) + \mathbf{F}(t)\mathbf{w}(t), \\ \mathbf{w}(t) &\sim \mathcal{GP}(\mathbf{0},\ \mathbf{Q}_c\delta(t - t')) \qquad t_0 < t, t' \end{aligned} \tag{14}$$

where $\mathbf{x}(t)$ is trajectory, $\mathbf{v}(t)$ is known exogenous input, $\mathbf{w}(t)$ is process noise, and $\mathbf{F}(t)$ is time-varying system matrix. The process noise $\mathbf{w}(t)$ is modeled by a Gaussian process, and $\delta(\cdot)$ is the *Dirac delta function*. (See [2] for details.) We consider a specific case of this model in the experimental results in Sect. 4.1. Because the mean function $\boldsymbol{\mu}(t)$ is an integral of the known exogenous input $\mathbf{v}(t)$, the assumption of zero $\mathbf{v}(t)$ leads to Gaussian process with zero mean $\boldsymbol{\mu}(t)$.

Assuming the GP is generated by Eq. 14, the measurements are landmark and odometry measurements, and the variables are ordered in XL ordering,[1] the sparse information matrix becomes

$$\boldsymbol{\mathcal{I}} = \begin{bmatrix} \boldsymbol{\mathcal{I}}_{xx} & \boldsymbol{\mathcal{I}}_{x\ell} \\ \boldsymbol{\mathcal{I}}_{x\ell}^{\mathsf{T}} & \boldsymbol{\mathcal{I}}_{\ell\ell} \end{bmatrix} \tag{15}$$

where $\boldsymbol{\mathcal{I}}_{xx}$ is block-tridiagonal and $\boldsymbol{\mathcal{I}}_{\ell\ell}$ is block-diagonal. $\boldsymbol{\mathcal{I}}_{x\ell}$'s density depends on the frequency of landmark measurements, and how they are taken.

When the GP is generated by LTV SDE, $\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1}$ in Eq. 12 has a specific sparsity pattern — only two column blocks that correspond to trajectory states at t_{i-1} and t_i are nonzero ($t_{i-1} < \tau < t_i$) [2]:

$$\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1} = \begin{bmatrix} 0 \ldots 0 \; \boldsymbol{\Lambda}(\tau) \; \boldsymbol{\Psi}(\tau) \; 0 \ldots 0 \end{bmatrix} \tag{16}$$

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_{i-1}) - \mathbf{Q}_\tau \boldsymbol{\Phi}(t_i, \tau)^{\mathsf{T}} \mathbf{Q}_i^{-1} \boldsymbol{\Phi}(t_i, t_{i-1}), \qquad \boldsymbol{\Psi}(\tau) = \mathbf{Q}_\tau \boldsymbol{\Phi}(t_i, \tau)^{\mathsf{T}} \mathbf{Q}_i^{-1}$$

$\boldsymbol{\Phi}(\tau, s)$ is the state transition matrix from s to τ. $\mathbf{Q}_\tau$ is the integral of $\mathbf{Q}_c$, the covariance of the process noise $\mathbf{w}(t)$ (Eq. 14):

$$\mathbf{Q}_\tau = \int_{t_{i-1}}^{\tau} \boldsymbol{\Phi}(\tau, s)\mathbf{F}(s)\mathbf{Q}_c\mathbf{F}(s)^{\mathsf{T}} \boldsymbol{\Phi}(\tau, s)^{\mathsf{T}} ds \tag{17}$$

And $\mathbf{Q}_i$ is the integral from t_{i-1} to t.

Consequently, based on Eqs. 12 and 16, $\bar{\mathbf{x}}(\tau)$ is an affine function of only two nearby states $\bar{\mathbf{x}}(t_{i-1})$ and $\bar{\mathbf{x}}(t_i)$ (the current estimate of the states at t_{i-1} and t_i):

$$\bar{\mathbf{x}}(\tau) = \boldsymbol{\mu}(\tau) + \begin{bmatrix} \boldsymbol{\Lambda}(\tau) \; \boldsymbol{\Psi}(\tau) \end{bmatrix} \left(\begin{bmatrix} \bar{\mathbf{x}}(t_{i-1}) \\ \bar{\mathbf{x}}(t_i) \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}(t_{i-1}) \\ \boldsymbol{\mu}(t_i) \end{bmatrix} \right), \;\; t_{i-1} < \tau < t_i \tag{18}$$

Thus, it only takes $O(1)$ time to query any $\bar{\mathbf{x}}(\tau)$ using Eq. 18. Moreover, because interpolation of a state is only determined by the two nearby states, measurement interpolation in Eq. 13 can be simplified to:

[1] XL ordering is an ordering where process variables come before landmarks variables.

$$\begin{aligned}\mathbf{h}_k\left(\bar{\boldsymbol{\theta}}_k + \delta\boldsymbol{\theta}_k\right) &= \mathbf{h}_k\left(\bar{\mathbf{x}}(\tau) + \delta\mathbf{x}(\tau)\right) \\ &\approx \mathbf{h}_k(\bar{\mathbf{x}}(\tau)) + \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}(\tau)} \cdot \frac{\partial \mathbf{x}(\tau)}{\partial \mathbf{x}}\bigg|_{\bar{\mathbf{x}}} \delta\mathbf{x} \\ &= \mathbf{h}_k(\bar{\mathbf{x}}(\tau)) + \mathbf{H}_k \left[\boldsymbol{\Lambda}(\tau)\,\boldsymbol{\Psi}(\tau)\right] \begin{bmatrix}\delta\mathbf{x}(t_{i-1}) \\ \delta\mathbf{x}(t_i)\end{bmatrix}\end{aligned} \tag{19}$$

with $\bar{\mathbf{x}}(\tau)$ defined in Eq. 18.

3 The Bayes Tree Data Structure for Fast Incremental Updates to Sparse Gaussian Process Regression

Previous work on batch continuous-time trajectory estimation as sparse Gaussian process regression [2, 15] assumes that the information matrix $\mathcal{I}$ is sparse (Eq. 15) and applies standard block elimination to factor and solve Eq. 11. But for large numbers of landmarks, this process is very inefficient. In square root SAM [4], matrix column reordering has been applied for efficient Cholesky factorization in a discrete-time context. Similarly, naive periodic variable reordering can be employed here to solve the STEAM problem. (See [16] for details.)

However, despite the efficiency of periodic batch updates, it is still repeatedly executing a batch algorithm that requires reordering and refactoring $\mathcal{I}$, and periodically relinearizing the measurement function for all of the estimated states each time new data is collected. Here we provide the extensions necessary to avoid these costly steps and turn the naive batch algorithm into an efficient, truly incremental, algorithm. The key idea is to perform just-in-time relinearization and to efficiently *update* an existing sparse factorization instead of re-calculating one from scratch.

3.1 The Bayes Tree Data Structure

We base our approach on iSAM 2.0 proposed by Kaess et al. [11], which was designed to efficiently solve a nonlinear estimation problem in an incremental and real-time manner by directly operating on the factor graph representation of the SAM problem. The core technology behind iSAM 2.0 is the *Bayes tree* data structure which allows for incremental variable reordering and fluid relinearization [10]. We apply the same data structure to sparse Gaussian process regression in the context of the STEAM problem, thereby eliminating the need for periodic batch computation.

The Bayes tree data structure captures the formal equivalence between the sparse QR factorization in linear algebra and the inference in graphical models, translating *abstract updates* to a matrix factorization into *intuitive edits* to a graph. Here we give a brief introduction of Bayes trees (see [10] for details), and how they help solve the sparse Gaussian process regression incrementally.

A Bayes tree is constructed from a Bayes net, which is further constructed from a factor graph. A factor graph is a bipartite graph $G = (\boldsymbol{\theta}, \mathcal{F}, \mathcal{E})$, representing the factorization of a function (Eq. 20). $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_m\}$ are *variables*, $\mathcal{F} = \{f_1, \dots, f_n\}$ are *factors* (functions of variables), and $\mathcal{E}$ are the *edges* that connect these two types of nodes. $e_{ij} \in \mathcal{E}$ if and only if $\theta_j \in \boldsymbol{\theta}_i$ and $f_i(\cdot)$ is a function of θ_i.

$$f(\boldsymbol{\theta}) = \prod_i f_i(\boldsymbol{\theta}_i) \tag{20}$$

In the context of localization and mapping, a factor graph encodes the complex probability estimation problem in a graphical model. It represents the *joint density* of the variables consisting of both trajectory and mapping, and factors correspond to the soft constraints imposed by the measurements and priors. If we assume that the priors are Gaussian, measurements have Gaussian noise, and measurement functions are linear or linearized, as in Sect. 2, the joint density becomes a product of Gaussian distributions:

$$f(\boldsymbol{\theta}) \propto \exp\left\{-\frac{1}{2}\sum \|\mathbf{A}_i\theta_i - \mathbf{b}_i\|_2^2\right\} = \exp\left\{-\frac{1}{2}\|\mathbf{A}\theta - \mathbf{b}\|_2^2\right\} \tag{21}$$

Here $\mathbf{A}_i$ and $\mathbf{b}_i$ are derived from factor $f_i(\cdot)$. $\mathbf{A}$ is a square-root information matrix, with $\boldsymbol{\mathcal{I}} = \mathbf{A}^\intercal\mathbf{A}$ [4], so the QR factor $\mathbf{R}$ of $\mathbf{A}$ is equal to the transpose of the Cholesky factor $\boldsymbol{\mathcal{L}}$ of $\boldsymbol{\mathcal{I}}$. Maximizing the joint density is equivalent to the least-square problem in Eq. 9.

A Gaussian process generated from linear, time-varying (LTV) stochastic differential equations (SDE), as discussed in Sect. 2.2, has a block-tridiagonal inverse kernel matrix $\boldsymbol{\mathcal{K}}^{-1}$ and can be represented by a *sparse* factor graph [2]. In this case, the factors derived from the Gaussian process prior are (suppose $f_j(\cdot)$ is the GP factor between $\mathbf{x}(t_{i-1})$ and $\mathbf{x}(t_i)$):

$$f_j(\boldsymbol{\theta}_j) = f_j(\mathbf{x}(t_{i-1}), \mathbf{x}(t_i)) \propto \exp\left\{-\frac{1}{2}\left\| \boldsymbol{\Phi}(t_i, t_{i-1})\mathbf{x}(t_{i-1}) + \mathbf{v}_i - \mathbf{x}(t_i)\right\|_{\mathbf{Q}_i}^2\right\} \tag{22}$$

where $\boldsymbol{\Phi}(t_i, t_{i-1})$ is the state transition matrix, $\mathbf{Q}_i$ is the integral of the covariance of the process noise (Eq. 17), and $\mathbf{v}_i$ is the integral of the exogenous input $\mathbf{v}(t)$ (Eq. 14):

$$\mathbf{v}_i = \int_{t_{i-1}}^{t_i} \boldsymbol{\Phi}(t_i, s)\mathbf{v}(s)ds \tag{23}$$

An illustrative sparse factor graph example including the GP factors is presented in Fig. 1a. Note that although the Gaussian process representation of the trajectory is continuous in time, to impose this prior knowledge only $M - 1$ factors connecting adjacent states are required, where M is the total number of states [2].

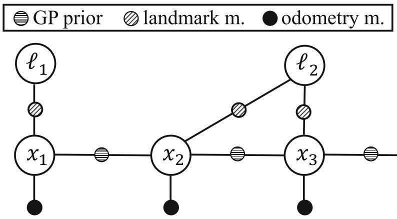


Fig. 1 A simple factor graph that includes landmark measurements, odometry measurements, and Gaussian process priors. The odometry measurements are unitary, when they measure the instant velocity in the robot state

The key of just-in-time relinearization and fluid variable reordering is to identify the portion of a graph impacted by a new or modified factor, which is difficult to achieve directly from a factor graph. So the factor graph is first converted to a Bayes net through the iterative elimination algorithm related to Gaussian elimination. In each step, one variable θ_i is eliminated from the joint density $f(\theta_i, \mathbf{s}_i)$ and removed from the factor graph, resulting in a new conditional $P(\theta_i|\mathbf{s}_i)$ and a new factor $f(\mathbf{s}_i)$, satisfying $f(\theta_i, \mathbf{s}_i) = P(\theta_i|\mathbf{s}_i)f(\mathbf{s}_i)$. The joint density $f(\theta_i, \mathbf{s}_i)$ is the product of the factors adjacent to θ_i, and $\mathbf{s}_i$ is the set of variables that are connected to these factors, excluding θ_i. The new conditional is added to the Bayes net, and the new factor is added back to the factor graph.

The unnormalized joint density $f(\theta_i, \mathbf{s}_i)$ is Gaussian, due to Eq. 21:

$$f(\theta_i, \mathbf{s}_i) \propto \exp\left\{-\frac{1}{2}\|\mathbf{a}\theta_i + \mathbf{A}_s\mathbf{s}_i - \mathbf{b}_i\|_2^2\right\} \tag{24}$$

where $\mathbf{a}$, $\mathbf{A}_s$ and $\mathbf{b}_i$ correspond to the factors that are currently adjacent to θ_i. These factors can be the factors included in the original factor graph, or the factors induced by the elimination process. The conditional $P(\theta_i|\mathbf{s}_i)$ is obtained by evaluating Eq. 24 with a given $\mathbf{s}_i$:

$$P(\theta_i|\mathbf{s}_i) \propto \exp\left\{-\frac{1}{2}(\theta_i + \mathbf{r}^\mathsf{T}\mathbf{s}_i - d)^2\right\} \tag{25}$$

where $\mathbf{r} = (\mathbf{a}^\dagger\mathbf{A}_s)^\mathsf{T}$, $d = \mathbf{a}^\dagger\mathbf{b}_i$, and $\mathbf{a}^\dagger = (\mathbf{a}^\mathsf{T}\mathbf{a})^{-1}\mathbf{a}^\mathsf{T}$. $f(\mathbf{s}_i)$ can be further computed by substituting $\theta_i = d - \mathbf{r}^\mathsf{T}\mathbf{s}_i$ into Eq. 24. This elimination step is equivalent to one step of Gram-Schmidt. Thus the new conditional $P(\theta_i|\mathbf{s}_i)$ specifies one row in the $\mathbf{R}$ factor of the QR factorization of $\mathbf{A}$. The sequence of the variables to be eliminated is selected to reduce fill-in in $\mathbf{R}$, just as in the case of matrix column reordering. The joint density $f(\boldsymbol{\theta})$ represented by the Bayes net is maximized by assigning $d - \mathbf{r}^\mathsf{T}\mathbf{s}_i$ to θ_i, due to Eq. 25, starting from the variable that is eliminated last. This procedure is equivalent to the back-substitution in linear algebra. The Bayes net is further transformed into a directed tree graphical model – the Bayes tree, by grouping together conditionals belonging to a clique in the Bayes net in reverse elimination order.

When a factor is modified or added to the Bayes tree, the impacted portion of the Bayes tree is re-interpreted as a factor graph, the change is incorporated to the graph, and the graph is eliminated with a new ordering. During elimination, information only flows upward in the Bayes tree, from leaves to the root, so only the ascendants of the nodes that contain the variables involved in the factor are impacted.

The Bayes tree can be used to perform fast incremental updates to the Gaussian process representation of the continuous-time trajectory. As we demonstrate in the experimental results, this can greatly increase the efficiency of Barfoot et al.'s batch sparse GP algorithm when the trajectory and map need to be updated online.

Despite the interpretation of the trajectory as a Gaussian process, the approach described above is algorithmically identical to iSAM2.0 when the states associated with each measurement are explicitly estimated. In Sect. 3.2 below, we extend our incremental algorithm to use Gaussian process interpolation within the Bayes tree. By interpolating missing states, we can handle asynchronous measurements and even remove states in order to speed computation. In Sects. 4.1 and 4.2 we show that this results in a significant speedup over iSAM2.0.

3.2 Faster Updates Through Interpolation

To handle asynchronous measurements or to further reduce computation time, we take advantage of Gaussian process state interpolation, described in Sect. 2.1, within our incremental algorithm. This allows us to reduce the total number of estimated states, while still using all of the measurements, including those that involve interpolated states. By only estimating a small fraction of the states along the trajectory, we realize a large speedup relative to a naive application of the Bayes tree (see Sect. 4). This is an advantage of continuous-time GP-based methods compared to discrete-time methods like iSAM 2.0.

To use Gaussian process interpolation within our incremental algorithms, we add a new type of factors that correspond to missing states (states to be interpolated).

We start by observing that, from Eq. 2, the factor $f_j(\cdot)$ derived from the measurement $h_k(\cdot)$ is:

$$f_j(\boldsymbol{\theta}_j) \propto \exp\left\{-\frac{1}{2}\|\mathbf{h}_k(\boldsymbol{\theta}_k + \delta\boldsymbol{\theta}_k) - \mathbf{y}_k\|^2_{\mathbf{R}_k}\right\} \tag{26}$$

where $\boldsymbol{\theta}_j$ (the variables adjacent to factor $f_j(\cdot)$), and $\boldsymbol{\theta}_k$ (the variables related to measurement $h_k(\cdot)$), are the same set of variables.

Without loss of generality, we assume that $\mathbf{x}(\tau)$ is the set of variables related to the measurement and the factor, with $t_{i-1} < \tau < t_i$, so f_j is a unitary factor of $\mathbf{x}(\tau)$:

$$f_j(\boldsymbol{\theta}_j) \propto \exp\left\{-\frac{1}{2}\|\mathbf{h}_k\left(\bar{\mathbf{x}}(\tau) + \delta\mathbf{x}(\tau)\right) - \mathbf{y}_k\|^2_{\mathbf{R}_k}\right\}, \quad \boldsymbol{\theta}_j \triangleq \delta\mathbf{x}(\tau) \tag{27}$$

If $\mathbf{x}(\tau)$ is *missing*, then this factor can not be added to the factor graph directly, because a missing state implies that it should not be estimated explicitly. Instead of creating a new state directly, we interpolate the state and utilize the linearized measurement function after interpolation (Eq. 13):

$$f_j(\boldsymbol{\theta}_j) \propto \exp\left\{-\frac{1}{2}\|\mathbf{h}_k\left(\bar{\mathbf{x}}(\tau)\right) + \mathbf{H}_k\boldsymbol{\mathcal{K}}(\tau)\boldsymbol{\mathcal{K}}^{-1}\delta\mathbf{x} - \mathbf{y}_k\|^2_{\mathbf{R}_k}\right\}, \ \boldsymbol{\theta}_j \triangleq \delta\mathbf{x} \qquad (28)$$

We apply the interpolation equations for the sparse GP (Eqs. 16 and 18), so that the factor becomes a function of the two nearby states (in contrast to the missing state):

$$f_j(\boldsymbol{\theta}_j) \propto \exp\left\{-\frac{1}{2}\|\mathbf{h}_k(\bar{\mathbf{x}}(\tau)) + \mathbf{H}_k\left[\boldsymbol{\Lambda}(\tau)\,\boldsymbol{\Psi}(\tau)\right]\boldsymbol{\theta}_j - \mathbf{y}_k\|^2_{\mathbf{R}_k}\right\}, \boldsymbol{\theta}_j \triangleq \begin{bmatrix}\delta\mathbf{x}(t_{i-1})\\ \delta\mathbf{x}(t_i)\end{bmatrix} \qquad (29)$$

where $\bar{\mathbf{x}}$ is specified in Eq. 18.

A factor graph augmented with the factors associated with measurements at missing states has several advantages. We can avoid estimating a missing state at time t explicitly, but still make use of a measurement at time t. This allows our algorithm to naturally handle asynchronous measurements. We can also reduce the size of the Bayes tree and the associated matrices by skipping states, which results in a reduction of computation time. Importantly, incorporating GP state interpolation and regression (Sects. 2.1 and 2.2) within Bayes tree closely follows MAP inference. In particular, we show in Sects. 4.1, and 4.2 that skipping large numbers of states can reduce computation time by almost 70% with only a small reduction in accuracy. The full incremental algorithm is described in Algorithm 1.

Algorithm 1 Incremental Sparse GP Regression visa the Bayes tree with Gaussian Process Priors (BTGP)

```
Set the sets of affected variables, variables involved in new factors, and relinearized variables to
empty sets, $\boldsymbol{\theta}_{aff} := \boldsymbol{\theta}_{nf} := \boldsymbol{\theta}_{rl} := \varnothing$.
while collecting data do
    1. Collect measurements, store as new factors. Set $\boldsymbol{\theta}_{nf}$ to the set of variables involved in the new
    factors. If $\mathbf{x}(\tau) \in \boldsymbol{\theta}_{nf}$ is a missing state, replace it by nearby states (Eq. 18); If $\mathbf{x}(\tau) \in \boldsymbol{\theta}_{nf}$ is a
    new state to estimate, a GP prior (Eq. 22) is stored, and $\boldsymbol{\theta}_{nf} := \boldsymbol{\theta}_{nf} \cup \mathbf{x}_{i-1}$.
    2. For all $\theta_i \in \boldsymbol{\theta}_{aff} = \boldsymbol{\theta}_{rl} \cup \boldsymbol{\theta}_{nf}$, remove the corresponding cliques and ascendants up to the
    root of the Bayes tree.
    3. Relinearize the factors required to create the removed part, using interpolation when missing
    states are involved (Eq. 29).
    4. Add the cached marginal factors from the orphaned sub-trees of the removed cliques.
    5. Eliminate the graph by a new ordering into a Bayes tree, attach back orphaned sub-trees.
    6. Partially update estimate from the root, stop when updates are below a threshold.
    7. Collect variables, for which the difference between the current estimate and the previous
    linearization point is above a threshold, into $\boldsymbol{\theta}_{rl}$.
end while
```

4 Experimental Results

We evaluate the performance of our incremental sparse GP regression algorithm to solving the STEAM problem on synthetic and real-data experiments and compare our approach to the state-of-the-art. In particular, we evaluate how variable reordering can dramatically speed up the batch solution to the sparse GP regression problem, and how, by utilizing the Bayes tree and interpolation for incremental updates, our algorithm can yield even greater gains in the online trajectory estimation scenario. We compare:

- **PB**: Periodic batch (described in Sect. 2). This is the state-of-the-art algorithm presented in Barfoot et al. [2] (XL variable ordering), which is periodically executed as data is received.
- **PBVR**: Periodic batch with variable reordering [16]. Variable reordering is applied to achieve efficient matrix factorization.
- **BTGP**: The proposed approach - Bayes tree with Gaussian process prior factors (described in Sect. 3).

If the GP is only used to estimate the state at measurement times, the proposed approach offers little beyond a reinterpretation of the standard discrete-time iSAM 2.0 algorithm. Therefore, we also compare our GP-based algorithm, which leverages interpolation, to the standard Bayes tree approach used in iSAM 2.0. We show that by interpolating large fractions of the trajectory during optimization, the GP allows us to realize significant performance gains over iSAM 2.0 with minimal loss in accuracy. For these experiments we compare:

- **without interpolation**: BTGP without interpolation at a series of lower temporal resolutions. The lower the resolution, the fewer the states to be estimated. Without interpolation BTGP is algorithmically identical to iSAM 2.0 with coarse discretization of the trajectory. Measurements between two estimated states are simply ignored.
- **with interpolation**: BTGP with interpolation at a series of lower resolutions. In contrast to the above case, measurements between estimated states are fully utilized by interpolating missing states at measurement times (described in Sect. 3.2).
- **finest estimate**: The baseline. BTGP at the finest resolution, estimating all states at measurement times. When measurements are synchronous with evenly-spaced waypoints and no interpolation is used, BTGP is identical to iSAM 2.0 applied to the full dataset with all measurements.

All algorithms are implemented with the same C++ library, GTSAM 3.2,[2] to make the comparison fair and meaningful. Evaluation is performed on two datasets summarized in Table 1. We first evaluate performance in a synthetic dataset (Sect. 4.1), analyzing estimation errors with respect to ground truth data. Results using a real-world dataset are then presented in Sect. 4.2.

[2] https://collab.cc.gatech.edu/borg/gtsam/.

Table 1 Summary of the experimental datasets

	# time steps	# odo. m.	# landmark m.	# landmarks	Travel dist. (km)
Synthetic	1,500	1,500	1,500	298	0.2
Auto. Mower	9,658	9,658	3,529	4	1.9

4.1 Synthetic SLAM Exploration Task

This dataset consists of an exploration task with 1,500 time steps. Each time step contains a trajectory state $\mathbf{x}(t_i) = [\,\mathbf{p}(t_i)^\intercal \; \dot{\mathbf{p}}(t_i)^\intercal\,]^\intercal$, $\mathbf{p}(t_i) = [\,x(t_i) \; y(t_i) \; \theta(t_i)\,]^\intercal$, an odometry measurement, and a range measurement related to a nearby landmark. The total number of landmarks is 298. The trajectory is randomly sampled from a Gaussian process generated from white noise acceleration $\ddot{\mathbf{p}}(t) = \mathbf{w}(t)$, i.e. constant velocity, and with zero mean.

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{F}\mathbf{w}(t) \tag{30}$$

$$\mathbf{x}(t) = \left[\mathbf{p}(t)^\intercal \; \dot{\mathbf{p}}(t)^\intercal\right]^\intercal, \quad \mathbf{p}(t) = \left[x(t) \; y(t) \; \theta(t)\right]^\intercal, \quad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$
$$\mathbf{F} = \left[\mathbf{0} \; \mathbf{I}\right]^\intercal, \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c \delta(t - t')) \tag{31}$$

Note that velocity $\dot{\mathbf{p}}(t)$ must be included in trajectory state to represent the motion in LTV SDE form [2]. This Gaussian process representation of trajectory is also applied to the real dataset. The odometry and range measurements with Gaussian noise are specified as:

$$\mathbf{y}_{io} = \begin{bmatrix} \cos\theta(t_i) \cdot \dot{x}(t_i) + \sin\theta(t_i) \cdot \dot{y}(t_i) \\ \dot{\theta}(t_i) \end{bmatrix} + \mathbf{n}_o, \;\; y_{ir} = \left\| \left[x(t_i) \; y(t_i)\right]^\intercal - \boldsymbol{\ell}_j \right\|_2 + n_r \tag{32}$$

where $\mathbf{y}_{io}$ consists of the robot-oriented velocity and heading angle velocity with Gaussian noise, and y_{ir} is the distance between the robot and a specific landmark $\boldsymbol{\ell}_j$ at t_i with Gaussian noise. The estimation results are shown in Fig. 2

We compare the computation time of the three approaches (PB, PBVR and BTGP) in Fig. 3. The incremental Gaussian process regression (BTGP) offers significant improvements in computation time compared to the batch approaches (PBVR and PB).

In Fig. 3, we also demonstrate that BTGP can further increase speed over a naive application of the Bayes tree (e.g. iSAM 2.0) without sacrificing much accuracy by leveraging interpolation. To illustrate the trade-off between the accuracy and time efficiency due to interpolation, we plot RMSE of distance errors and the total computation time by varying the time step difference (the rate of interpolation) between estimated states.

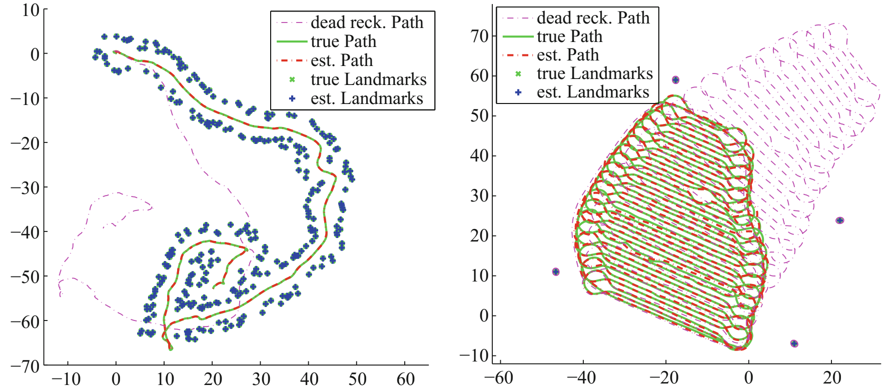


Fig. 2 (*left*) Synthetic dataset: Ground truth, dead reckoning path, and the estimates are shown. State and landmark estimates obtained from BTGP approach are very close to ground truth. (*right*) The Autonomous Lawnmower dataset: Ground truth, dead reckoning path and estimates are shown. The range measurements are sparse, noisy, and asynchronous. Ground truth and the estimates of path and landmarks obtained from BTGP are very close

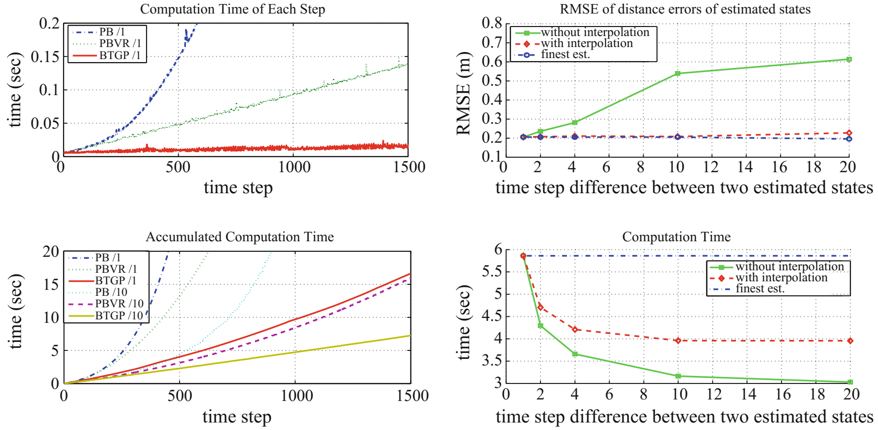


Fig. 3 Synthetic dataset: (*left*) Comparison of the computation time of three approaches PB, PBVR, and BTGP. The modifiers /1 and /10 indicate frequency of estimate updates — the number of range measurements between updates. Due to the large number of landmarks, 298, variable reordering dramatically improves the performance. (*right*) Trade-off between computation time and accuracy if BTGP makes use of interpolation. The y-axis measures the RMSE of distance errors of the estimated trajectory states and total computation time with increasing amounts of interpolation. The x-axis measures the time step difference between two estimated (non-interpolated) states. The results indicate that interpolating ∼90% of the states (i.e. estimating only ∼10% of the states) while running BTGP can result in a 33% reduction in computation time over iSAM 2.0 without sacrificing accuracy

4.2 The Autonomous Lawnmower

The second experiment evaluates our approach on real data from a freely available range-only SLAM dataset collected from an autonomous lawn-mowing robot [6]. The "Plaza" dataset consists of odometry data and range data to stationary landmarks collected via time-of-flight radio nodes. (Additional details on the experimental setup can be found in [6].) Ground truth paths are computed from GPS readings and have 2 cm accuracy according to [6]. The environment, including the locations of the landmarks and the ground truth paths, are shown in Fig. 2. The robot travelled 1.9 km, occupied 9,658 poses, and received 3,529 range measurements, while following a typical path generated during mowing. The dataset has sparse range measurements, but contains odometry measurements at each time step. The results of incremental BTGP are shown in Fig. 2 and demonstrate that we are able to estimate the robot's trajectory and map with a very high degree of accuracy.

As in Sect. 4.1, performance of three approaches – PB, PBVR, and BTGP are compared in Fig. 4. In this dataset, the number of landmarks is 4, which is extremely small relative to the number of trajectory states, so there is no performance gain from reordering. However, the Bayes tree-based approach dramatically outperforms the

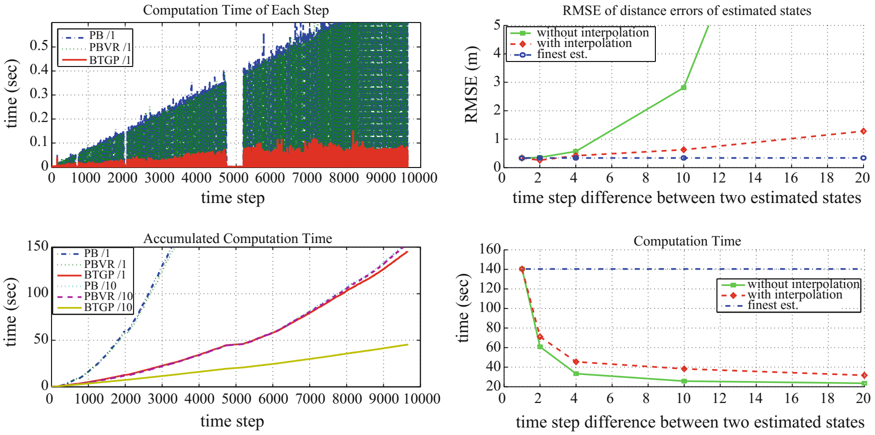


Fig. 4 Autonomous Lawnmower dataset: (*left*) Comparison of the computation time of PB, PBVR, and BTGP. As in Fig. 3, /1 and /10 are modifiers — the number of range measurement between updates, and no interpolation is used by BTGP. The 'gap' in the upper graph is due to a long stretch around timestep 5000 with no range measurements. Due to the low number of landmarks, variable reordering does not help The incremental BTGP approach dramatically reduces computation time. (*right*) Trade-off between computation time and accuracy if BTGP makes use of interpolation. The y-axis measures the RMSE of distance errors and total computation time with increasing amounts of interpolation. The x-axis measures the time step difference between two estimated (non-interpolated) states. The results indicate that interpolating ∼80% of the states within BTGP results in only an 8 cm increase in RSME while reducing the overall computation time by 68% over iSAM 2.0

other two approaches. As the problem size increases, there is negligible increase in computation time, even for close to 10,000 trajectory states.

In Fig. 4, the results of interpolation at different levels of resolutions are presented, which indicate a significant reduction in computation time can be achieved with minor sacrifice in accuracy.

5 Conclusion

We have introduced an incremental sparse Gaussian process regression algorithm for computing the solution to the continuous-time simultaneous trajectory estimation and mapping (STEAM) problem. The proposed algorithm elegantly combines the benefits of Gaussian process-based approaches to STEAM while simultaneously employing state-of-the-art innovations from incremental discrete-time algorithms for smoothing and mapping. Our empirical results show that by parameterizing trajectories with a small number of states and utilizing Gaussian process interpolation, our algorithm can realize large gains in speed over iSAM 2.0 with very little loss in accuracy (e.g. reducing computation time by 68% while increasing RMSE by only 8 cm on the Autonomous Lawnmower Dataset).

References

1. Bailey, T., Durrant-Whyte, H.: Simultaneous localisation and mapping (SLAM): part II state of the art. Robot. Autom. Mag. **13**(3), 108–117 (2006)
2. Barfoot, T., Tong, C.H., Sarkka, S.: Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In: Proceedings of Robotics: Science and Systems, Berkeley, USA (2014)
3. Boots, B., Gordon, G.J.: A spectral learning approach to range-only SLAM. In: Proceedings of the 30th International Conference on Machine Learning (ICML) (2013)
4. Dellaert, F., Kaess, M.: Square root sam: simultaneous localization and mapping via square root information smoothing. Int. J. Robot. Res. **25**, 2006 (2006)
5. Dennis, Jr. J.E., Schnabel, R.B.: Numerical methods for unconstrained optimization and nonlinear equations (Classics in Applied Mathematics, 16). Soc. Ind. Appl. Math. (1996). ISBN 0898713641
6. Djugash, J.: Geolocation with Range: Robustness, Efficiency and Scalability. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2010)
7. Durrant-Whyte, H., Bailey, T.: Simultaneous localisation and mapping (slam): part i the essential algorithms. IEEE Robot. Autom. Mag. **2**, 2006 (2006)
8. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
9. Kaess, M., Ranganathan, A., Dellaert, F.: iSAM: incremental smoothing and mapping. IEEE Trans. Robot. **24**(6), 1365–1378 (2008). ISSN 1552-3098. 10.1109/TRO.2008.2006706
10. Kaess, M., Ila, V., Roberts, R., Dellaert, F.: The bayes tree: an algorithmic foundation for probabilistic robot mapping. In: Algorithmic Foundations of Robotics IX, pp. 157–173. Springer (2011)

11. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J.J., Dellaert, F.: iSAM2: incremental smoothing and mapping using the Bayes tree. Int. J. Robot. Res. IJRR **31**(2), 217–236 (2012)
12. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: a factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence, pp. 593–598. AAAI (2002)
13. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
14. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press (2005). ISBN 0262201623
15. Tong, C.H., Furgale, P., Barfoot, T.D.: Gaussian process gauss-newton for non-parametric simultaneous localization and mapping. Int. J. Robot. Res. **32**(5), 507–525 (2013)
16. Xinyan, Y., Indelman, V., Boots, B.: Incremental sparse gp regression for continuous-time trajectory estimation & mapping. arXiv preprint arXiv:1504.02696 (2015)

A General Region-Based Framework for Collaborative Planning

Jory Denny, Read Sandström and Nancy M. Amato

1 Introduction

Motion planning is the problem of computing a feasible trajectory for a robot in a complex environment. Motion planning has applications in robotics, virtual reality [19], bioinformatics [29], and computer-aided design [2], among others. However, motion planning is known to be computationally difficult [26].

To overcome this problem, attention has turned to sampling-based paradigms, such as Probabilistic RoadMaps (PRMs) [13] or Rapidly-exploring Random Trees (RRTs) [17], that address this complexity by constructing an approximation of the planning space. Despite the success of these fully automated approaches, some scenarios, e.g., certain types of narrow passages, remain problematic for these approaches [9]. In these scenarios, research has turned to developing heuristics that restrict the search and bias planning.

User-guided planners instead address this difficulty by attempting to harness the power of human intuition [2, 10]. In these systems, the human often performs a global scene analysis of the workspace, while the machine handles high-precision tasks such as collision detection and low level path-finding [6, 12].

Recent work has explored sampling-based planning strategies that incorporate interactive and collaborative planning techniques. Interactive-RRT (I-RRT) [30] allows a user to interactively steer RRT growth through the control of a robot avatar.

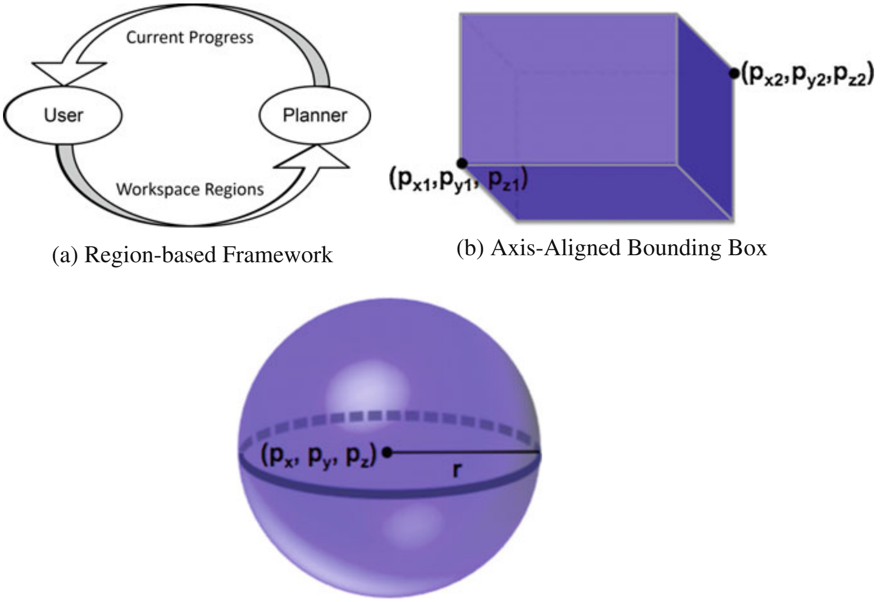
J. Denny (✉)
Department of Mathematics and Computer Science, University of Richmond,
Richmond, VA, USA
e-mail: jdenny@richmond.edu

R. Sandström · N.M. Amato
Parasol Lab, Department of Computer Science and Engineering,
Texas A&M University, College Station, TX, USA
e-mail: readamus@cse.tamu.edu

N.M. Amato
e-mail: amato@cse.tamu.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_32



(a) Region-based Framework

(b) Axis-Aligned Bounding Box

(c) Bounding Sphere

Fig. 1 **a** In one direction, the user specifies workspace regions, and in the other direction, the planner displays the current progress in planning. **b**, **c** Region examples

This method is restricted to RRT techniques and constrains the interface to one that can fully control the avatar. Region Steering [5] allows the user to bias PRM planners by specifying workspace regions for the planner to prefer or avoid. Both of these approaches yield speedup over standard automated approaches because the provided information biases the planner to prioritize difficult regions. However, no user-guided planner has yet been proposed that supports fully-interactive, collaborative planning for all sampling-based planning approaches, and all except Region Steering have constraints on their interfaces.

In this work, we generalize Region Steering into a region-based framework that supports collaboration with any sampling-based planner. We show three variants of this framework applied to graph-based, tree-based, and hybrid planners to demonstrate the generality and applicability of the approach. In this collaborative framework, shown in Fig. 1(a), the user specifies regions to either attract or repel the planner while the planner displays its current progress as feedback. Like other sampling-based techniques, this methodology biases the search to effectively guide a planner. Our framework maintains the probabilistic completeness of the underlying planner(s). One goal of this work is to study the benefits of this information for sampling-based approaches. Our contributions include:

- a general region-based framework for collaborative sampling-based planning,

- three variants of this framework employing graph-based, tree-based, and hybrid planning methods, and
- results demonstrating that this approach reduces planning time in two scenarios including a planar robot and an eight degree of freedom mobile manipulator.

Our framework is demonstrated in an interactive system that requires only intermittent user action on a standard computer interface, e.g., a mouse. The goal of this work is to understand how these hints and cooperation affect sampling-based planning. The analysis and optimization of the user interface is the subject of future work.

2 Preliminaries and Related Work

In this section, we present some basics of motion planning and review previous algorithms for sampling-based and user-guided motion planning.

2.1 Motion Planning Preliminaries

A robot is a movable object whose position and orientation can be described by n parameters, or *degrees of freedom* (DOFs), each corresponding to an axis of movement (e.g., positions, orientations, joint angles, etc.). Hence, a robot's placement, or configuration, can be described by a unique point $q = \langle x_1, x_2, ..., x_n \rangle$ in an n-dimensional space where x_i is the ith DOF. This space, consisting of all possible robot configurations (feasible or not), is called *configuration space* (C_{space}) [21]. The subset of all feasible configurations is the *free space* (C_{free}), while the union of the infeasible configurations is the *obstacle space* (C_{obst}). Thus, the motion planning problem becomes that of finding a continuous trajectory in C_{free} from a given start configuration q_s to a goal configuration q_g.

In general, it is intractable to compute explicit C_{obst} boundaries [26], but we can often determine whether a configuration is valid or invalid quite efficiently, e.g., by performing a collision detection (CD) test in the *workspace*, the robot's natural space.

Collision detection libraries often employ bounding volumes in the workspace to expedite this validity check [20]. The most common examples are axis-aligned bounding boxes (AABBs) and bounding spheres (BSs), shown in Fig. 1(b–c). In this work, we label such bounding volumes as *regions*. Specifically, we define a region R as any bounding volume in the workspace for which every point $p \in R$ maps to one or more configurations $Q \subseteq C_{space}$. To make the placement of configurations within R more obvious to users, we additionally require that for each $q \in Q$, the robot must lie entirely within R when configured at q.

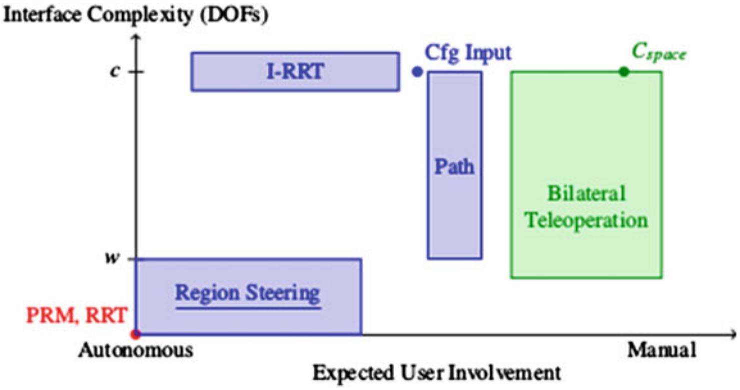


Fig. 2 Interface complexity in terms of degrees of freedom manipulated versus level of autonomy in expected behavior, where c is the degrees of freedom of the C_{space} and w is the degrees of freedom of the workspace. Our approach, Region Steering (underlined), is a combination of high autonomy mixed with a simple interface

2.2 *Related Work*

There is a broad spectrum of work in robotics dealing with motions for autonomous or semi-autonomous robots. We categorize these into a few related areas (Fig. 2): autonomous planning (red), human-in-the-loop planners (blue), and bilateral teleoperation (green). We examine these broad categories in terms of their respective autonomies and interface complexities.

2.2.1 Sampling-Based Motion Planners

Because of the high cost of explicitly computing C_{obst} boundaries, research has turned to sampling-based techniques [13, 17] to efficiently explore C_{free} for valid paths. In this section, we give a brief review of popular paradigms in sampling-based planning. Generally speaking, fully autonomous planners allow no user involvement and have no interface complexity.

PRMs [13] construct a map of C_{free} by first randomly sampling valid configurations. Nearby samples are then connected to form the edges of the map by validating simple paths between them. Finally, start and goal configurations are connected to the roadmap and a graph search, e.g., A*, is used to extract a solution path. These approaches show much success in multi-query problems, but lack efficiency in narrow and cluttered spaces.

RRTs [14, 17] gradually explore C_{space} from some start configuration q_{root} by iteratively sampling a random configuration q_{rand}, stepping q_{near} (the nearest node in the tree to q_{rand}) towards q_{rand} to create q_{new}, and adding q_{new} to a tree. RRTs are typically used in single-query problems. Also, RRTs have deficiencies in exploring narrow spaces of the environment and often fail to discover the entrances of narrow passages.

There have even been successful approaches combining PRMs and RRTs with the goal of achieving scalability on high performance computers [24] or more effectively exploring narrow passages [28]. Commonly, these approaches use some sort of global sampling over the space, i.e., PRM techniques, to cover C_{space}, and then use RRTs for local exploration to achieve high roadmap connectivity.

There are many variants of all three paradigms designed to address the narrow passage problem [1, 3, 8, 27, 34]. However, none of these planners are suited to every scenario and still have difficulties with certain classes of problems.

Region-based Frameworks. Some motion planning frameworks utilize region decomposition to bias sampling toward specific areas of the environment. One approach, Feature Sensitive Motion Planning [23], subdivides the space, individually constructs a map in each region, and merges them together to efficiently map heterogeneous environments. Other approaches utilize workspace decompositions to find narrow or difficult areas of the workspace to bias C_{space} sampling [25, 31]. However, by automatically identifying regions and disallowing dynamic region specification and modification, the planner might have inefficiencies, such as oversampling in a fully covered region. Additionally, these planners do not typically consider avoidance regions.

2.2.2 Human-in-the-Loop Planning

In many approaches to human-assisted planning, a human operator (user) performs global analysis of the workspace to determine an approximate solution while the machine handles high-precision tasks such as collision detection. In [2, 10, 32] the user can select configurations that are critical to finding a collision-free path, while the planner performs collision checking and path-finding between sub-goals (shown as Cfg Input in Fig. 2). Certain approaches allow a user to input an approximate path in the scene, and an autonomous planner then morphs this motion into a feasible plan [2, 16, 33] (shown as Path in Fig. 2). Often, these types of planners have distinct phases for user input and automated planning.

More recently, two-way communication approaches have been separately developed for RRTs (Interactive-RRT (I-RRT) [30]) and PRMs (Region Steering [5]). In these systems, the planner and the user interact in an online fashion to cooperatively solve the problem. I-RRT allows the user to control a robot avatar in a virtual scene that biases RRT growth. This approach, however, is limited to single-query scenarios, requires continuous user input, and is constrained to robotic systems that are fully controllable by the avatar interface (as seen in Fig. 2). Region Steering overcomes some of these weaknesses by allowing a user to specify workspace regions to bias PRM construction. In contrast to I-RRT, Region Steering requires neither continual user input nor a relationship between the robot and interface, which makes it applicable to a broader range of robotic systems. The work presented here generalizes Region Steering to be suitable for any sampling-based planning paradigm.

Bilateral Teleoperation. Teleoperation approaches provide closed-loop interactions between an operator and a robot such that the operator has a sense of presence-at-a-distance [7]. Teleoperation focuses on capturing a user's mechanical skills directly. Often, these approaches try to assist the human operator by predicting where the user is headed, ensuring proper collision avoidance, and maximizing user control over the system [18]. In contrast, human-in-the-loop planning aims to leverage a user's high-level intuition by augmenting an automated planner with information about difficult aspects of the problem.

Nonetheless, both seek to provide a form of two-way communication, referred to as bilateral control in teleoperation literature. Often these approaches have a high interface complexity, sometimes requiring interaction with haptic devices with many degrees of freedom, while attempting to provide as much control to the user as possible (Fig. 2). A recent study in teleoperation [22] shows that this form of interaction can be burdensome on the user, e.g., in situations with cyclic or repetitive motions, and takes steps to provide the robot with greater autonomy so that the user need only provide global guidance rather than direct control. Other teleoperation systems allow the user to control a subset of the robot's DOFs through a precomputed C_{space}, as in [11, 12], but these are limited to low dimensional problems. This approach is referred to as C_{space} in Fig. 2.

3 Framework

In this section, we describe our general framework for region-based collaborative planning, through which an automated planner and a human operator, referred to as a user, interact to cooperatively discover a solution to a motion planning problem.

Motivation. In essence, our framework provides a methodology to limit the search space of the planner. By specifying a presumably important (or unimportant) workspace region r, the user restricts the planner to focus on a particular subset of C_{space}. For example, if a region biases PRM construction, e.g., to a narrow passage, then the planner will focus and build a more dense roadmap in the narrow passage as compared with the portions of the C_{space} not covered by the region. Thus, our collaborative planner has the potential to provide more effective and efficient planning in terms of coverage and plan construction time.

There are a few important design considerations when implementing this framework for a specific algorithm. First, rendering and feedback should be real-time, or close to real-time, to allow a seamless collaboration. Second, an intuitive mechanism for region specification is needed. We chose Axis-Aligned Bounding Box (AABB) and Bounding Sphere (BS) regions for their simplicity and the ability to specify them with a common mouse interface. Finally, proper user feedback should be customized for the planner. For example, a region might be colored based upon an algorithm-specific perception of usefulness [5].

"Region" is a very broad term. A region might be seen as a path or constraint in the workspace to bias planning in C_{space}. For example, one could imagine selecting a wall as a contact-constraint for an end-effector of an articulated robot. There are many possibilities, and we selected a few representative scenarios for our study.

While our framework is quite general, it is important to note that our workspace regions are primarily effective in problems where the translational degrees of freedom dominate the system. This occurs often in systems such as mobile robots, unmanned aerial vehicles, or certain CAD applications. In contrast, applying this framework in other scenarios such as strongly rotational problems (e.g., the alpha puzzle) would require an alternative region type to successfully focus the planner.

Our framework does not necessarily have to be restricted to human-planner collaboration. A subject of future work is to examine planner-planner collaborations, where one planner identifies attract regions for another to focus on or avoid.

3.1 Overview

Our two-way communication framework is shown in Fig. 1(a).

In one direction, the user specifies workspace regions to bias the planner. Regions have "types" that can be arbitrarily extended for a particular application. By default, we allow two kinds of regions: attract (bias the planner) and avoid (disallow plans in this region). We allow these regions to be modified at any time. This includes addition of new regions, resizing/repositioning of current regions, and deletion of regions from the planning scene.

In the other direction, the planner presents its current progress in the scene. For sampling-based planners, this would be the current roadmap being constructed whether it be a graph or a tree. Additionally, we present the current regions to the user. Our framework is general enough to present a perceived usefulness of regions and even recommend regions to the user [5], however these are not fully explored in this work.

The algorithmic framework for the automated planner shown in Algorithm 1 follows this scheme. Generally, it performs a loop until the planner is finished, e.g., solves an example query or reaches a specific number of nodes. In each iteration, a specific planner biased by the user-defined regions performs automated planning. Next, the planner provides feedback to the user through the roadmap and region displays. This is important visual information to help the user adjust regions appropriately.

3.2 Completeness

To retain the completeness properties of the underlying planner, we consider the entire workspace as a region. Drawing a sample from this region is identical to drawing a sample from the entire C_{space}, i.e., is equivalent to the behavior of the

Algorithm 1 Region-based Collaborative Motion Planning Framework

```
Input: Environment $e$
Output: Roadmap $G$
$G = (V, E) \leftarrow (\emptyset, \emptyset)$
while $\neg done$ do
    $r \leftarrow$ SelectRegion($e.regions$)
    RegionBiasedPlanner($G, e, r$)
    $G$.UpdateMap()
    $e$.UpdateRegions()
return $G$
```

underlying unguided planner. By having a probability to select this global region, we inherit the probabilistic completeness property of the underlying planner.

There are two cases in which our framework cannot guarantee probabilistic completeness. The first is when the underlying planner is not probabilistically complete: while the user may be able to manipulate regions to solve the problem, a solution is not guaranteed. The second is when the user places an avoid region that changes the topology of C_{free}. Recall that avoid regions are hard constraints: placing such a region is equivalent to placing a virtual obstacle in the scene. Hence, the user can make the problem unsolvable by placing avoid regions to block all available solutions. This is a powerful tool that requires some discretion. While the user can unintentionally block valid solutions, they can also employ avoid regions to block out unimportant areas of the workspace.

4 Framework Variants

In this section, we show three variants of our framework: collaborative region-biased roadmap construction, collaborative region-biased tree construction, and a collaborative region-biased hybrid method.

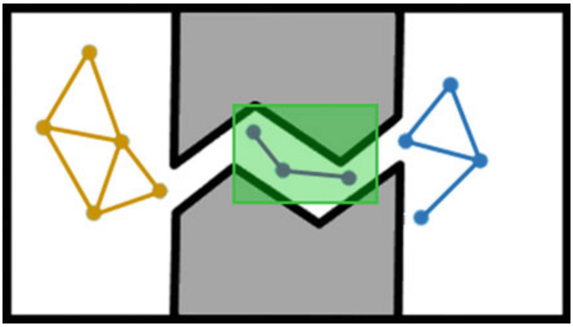


Fig. 3 A user has drawn a region to mark the narrow passage and set the sampler to OBPRM [1] in the region. As you can see, the samples generated in the region are close to the obstacles. However, outside the region, the samples are still generated using uniform sampling

4.1 Collaborative Region-Biased Roadmap Construction

This variant, called Region-biased PRM, is a collaborative roadmap construction approach in which a user specifies regions to bias a graph-based planning method, e.g., PRM. Shown in Algorithm 2 and Fig. 3, Region-biased PRM extends [5] by allowing the user to select a different sampler for each region. For example, if the user specifies a region around a narrow passage, they could also direct the planner to use obstacle-based sampling [1] within that region.

During sampling, an attract region is first selected at random (recall that the entire environment is also considered an attract region). Then, a sample is generated within that region using the user-selected sampler. Once a sample is created, it is checked to ensure it does not lie within any avoid regions. If the sample meets the criterion, it is added and connected to the roadmap. If the sample fails to connect, i.e., it is in a difficult area of C_{space}, the planner can additionally recommend regions to the user. After each iteration, the perceived usefulness of each region to the planner is shown (fully described in [5]).

Algorithm 2 Region-biased PRM

```
Input: A Roadmap $G$, an Environment $e$, and a Region $r$
$q \leftarrow r.sampler.$SAMPLE($r$)
if $q \notin a, \forall a \in e.avoidRegions$ then
    $G$.ADDANDCONNECT($q$)
    if ISDIFFICULTNODE($q$) then
        $e$.RECOMMENDREGION($q$)
```

4.2 Collaborative Region-Biased Tree Construction

The region-based strategy can also be extended to bias tree-based approaches, e.g., RRT. Our algorithm, called Region-biased RRT, is shown in Algorithm 3 and Fig. 4. The algorithm proceeds like a typical RRT by selecting a configuration q_{rand}, though in this case the selection is biased by a region. First, the planner selects a random region and generates a new configuration q_{rand} within. Then, the nearest node in the tree q_{near} is determined, and a node q_{new} is generated by extending q_{near} towards q_{rand}. In our system, this extension is not permitted to extend through avoid regions in the environment. It is important to note that the RRT also treats the whole environment as an attract region in order to maintain probabilistic completeness.

Algorithm 3 Region-biased RRT

```
Input: A Roadmap $G$, an Environment $e$, and a Region $r$
$q_{rand} \leftarrow r.\textsc{GetRandomCfg}()$
$q_{near} \leftarrow \textsc{NearestNeighbor}(G, q_{rand})$
$q_{new} \leftarrow \textsc{Extend}(q_{near}, q_{rand}, \Delta q)$
$G.\textsc{Update}(q_{near}, q_{new})$
```

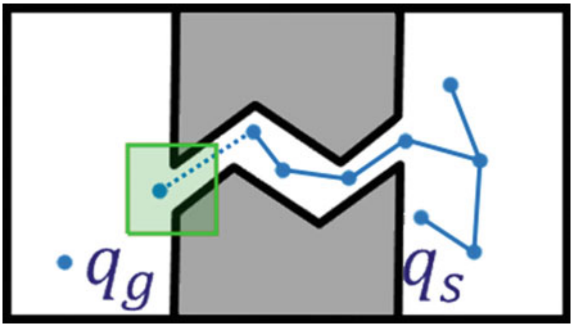


Fig. 4 A user has drawn a region to act as a waypoint for the RRT, influencing it to grow through the narrow passage. It also grows in random directions as seen by the upward extending portions of the tree

4.3 Collaborative Region-Biased Hybrid Methods

Here, we extend a recent hybrid approach called Spark PRM [28], a PRM planner that grows or "sparks" RRTs in narrow passages to increase roadmap connectivity. Using our framework, a user can specify regions to control where RRTs are sparked in the environment to aid PRM connection. These sparked RRTs are grown until they connect to the roadmap or reach a maximum number of iterations. Our algorithm, Region-biased Spark PRM, is shown in Algorithm 4 and Fig. 5.

Algorithm 4 Region-biased Spark PRM

```
Input: A Roadmap $G$, an Environment $e$, and a Region $r$
$q \leftarrow r.sampler.\textsc{Sample}(r)$
if $q \notin a, \forall a \in e.avoidRegions$ then
    $G.\textsc{AddAndConnect}(q)$
    if $r \neq e \wedge \textsc{InNarrowPassage}(q)$ then
        $G \leftarrow G \cup \textsc{ConstructRRT}(q)$
```

5 Experimental Demonstration

In this section, we evaluate our collaborative planners against their fully automated counterparts in two scenarios, involving a 2 DOF omnidirectional robot and an eight DOF mobile manipulator, respectively. We do not claim that the user interface is optimal or intuitive: it is merely sufficient for the user to communicate with the

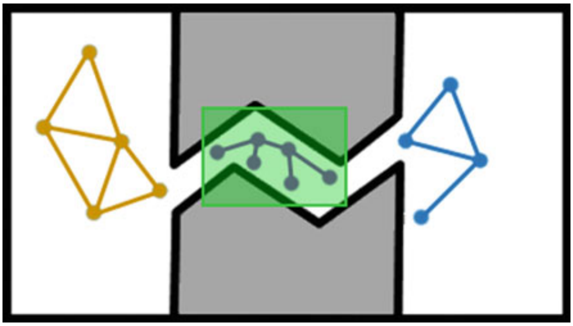
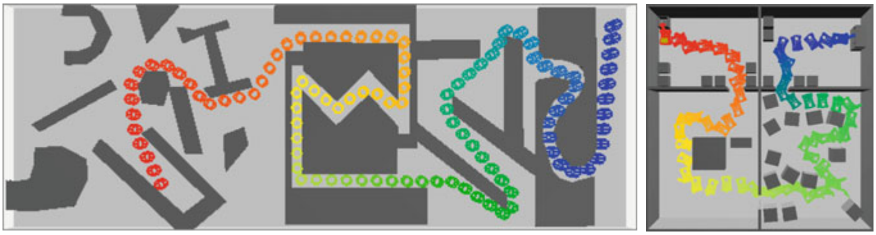


Fig. 5 A user has created a region to mark a narrow passage and to begin growing a RRT. The planner also generates samples throughout the environment



(a) `Planar` (b) `Manipulator`

Fig. 6 Example scenarios used in experimental analysis. **a** A planar 2DOF scenario. **b** An 8DOF mobile manipulator. All queries require traversal through narrow passages between the start (*red*) and goal (*blue*) configurations

planner and allows us to study the usefulness of our collaboration framework. We leave further development of the interface to future work.

General Setup All methods were implemented in a C++ motion planning library developed in the Parasol Lab at Texas A&M University. It uses a distributed graph data structure from the Standard Template Adaptive Parallel Library (STAPL) [4], a C++ library designed for parallel computing. Experiments were run on Dell Optiplex 780 computers running Fedora 19 with Intel Core 2 Quad CPU 2.83 GHz processors with the GNU gcc compiler version 4.8.

We evaluate each method in two scenarios as seen in Fig. 6. Queries are shown in start configuration (red) and goal configuration (blue) pairs.

- In `Planar` (Fig. 6a), a planar 2 DOF robot must traverse a series of difficult narrow passages and cluttered areas from the left to the right of the environment.
- In `Manipulator` (Fig. 6b), an 8 DOF KUKA youBot [15] model begins by reaching into an open box. It must then pass through doorways while navigating around boxes in an industrial scene. The query ends with the robot reaching into a cabinet on a table. This robot has an omnidirectional base and an arm with five joints.

We are interested in the success rate and the total time for the planner to solve each scenario (including user input and collaboration time). Experiments are run with 10 trials, and the metrics reported are averages of the successful runs.

The user-guided executions were performed by graduate and undergraduate students studying motion planning. In order to minimize the impact of user variance, the same user performed all of the executions in a given environment. Additionally, the users were allowed to practice with the system until they developed familiarity with the interface and environments.

5.1 Region-Biased PRM

Setup and Results. In this experiment, we analyze Region-biased PRM by comparing its performance with Basic PRM (referred to as Uniform) [13], OBPRM [1], and Gaussian PRM [3] (referred to as Gaussian). We analyze Region-biased PRM with a homogeneous use of uniform random sampling (referred to as RBPRM-U) and a heterogeneous use of uniform and obstacle-based sampling (RBPRM-H). For Gaussian PRM, we use both a tuned and untuned d value of the Gaussian distribution based upon twice the robot radius for the environment (referred to as Gaussian-T and Gaussian-U respectively). All methods use Euclidean distance, straight-line local planning, and a $k = 10$-closest neighbor connection strategy.

Each planner is run until either a construction query is solved or 5,000 nodes are sampled. The construction query is designed to verify that the roadmap well-represents C_{free} by requiring connectivity through the major areas of the environment. Success rates are shown in Table 1 and speedups compared with Gaussian-T are shown in Fig. 7.

Performance Comparison. In terms of success rates, Region-biased PRM outperforms each method solving both problems 100% of the time. Even with including the interaction time, Region-biased PRM generally solves each problem faster when

Table 1 Success rates of various PRM construction methods

	Uniform (%)	Gaussian-T (%)	Gaussian-U (%)	OBPRM(%)	RBPRM-U (%)	RBPRM-H (%)
`Planar`	0	90	40	10	100	100
`Manipulator`	100	100	100	100	100	100

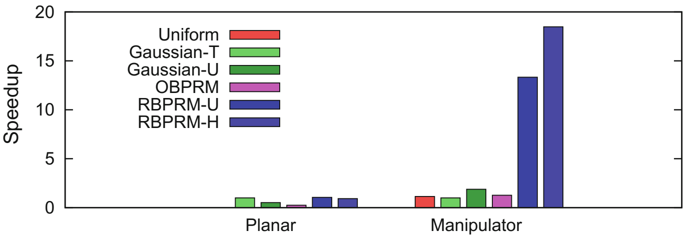


Fig. 7 Speedups of various PRM construction methods compared with Gaussian-T

compared to the fully automated approaches. In the easier environment, using a tuned Gaussian PRM has comparable performance to ours. Typically, we saw improvements up to two times compared with the best PRM performance in each problem. We note that tuning Gaussian PRM can be difficult and require running the problem many times to find an optimal value. Region-biased PRM is capable of acquiring more consistent results because the user can visually determine where the PRM has yet to map in the environment and focus planning there.

User Strategy. In `Planar`, the user's strategy generally involved allowing the planner to progress for a split second, identifying the narrow passage where configurations were scarce, and using regions to patch the roadmap in these areas. Similarly, the user's strategy in `Manipulator` was to create attract regions wherever the planner had not yet connected in the first three quadrants of the environment (i.e., those quadrants not containing the goal). However, the strategy changed for the goal quadrant because it contained a joint-posturing aspect. Here, the user employed avoid regions to remove unproductive configurations that sampled near the goal but did not connect to it. This improved the likeliness that a connectible configuration would see the goal as a nearest neighbor, thereby expediting the difficult task of connecting the goal to the map. The user deleted or moved attract regions to new locations as soon as the roadmap connected through them to limit redundant configurations.

5.2 Region-Biased RRT

Setup and Results. We compare Region-biased RRT (referred to as RBRRT) with RRT [17], OBRRT [27], and I-RRT [30]. We use $\Delta q = 10$ which is approximately 10% of the diagonal of each environment. With OBRRT, we use appropriate distribution of growth methods for each environment, but note there could be a more effective parameterization to optimize performance. I-RRT's parameters were selected based on recommendations in [30]. We only compare against I-RRT in the `Planar` environment because this is the only robot fully controllable by our interface, a mouse with 2 DOF (recall that I-RRT requires a 1-to-1 mapping between interface and robot DOF). Success rates are shown in Table 2 and speedups compared with RRT are shown in Fig. 8.

Performance Comparison. The interactive methods were able to solve both scenarios 100% of the time, whereas the automated planners were not able to in `Planar`. Unguided RRTs performed poorly because the maze-like shape of the problem's

Table 2 Success rates of various RRT construction methods

	RRT(%)	OBRRT(%)	I-RRT(%)	RBRRT (%)
`Planar`	10	90	100	100
`Manipulator`	90	100	–	100

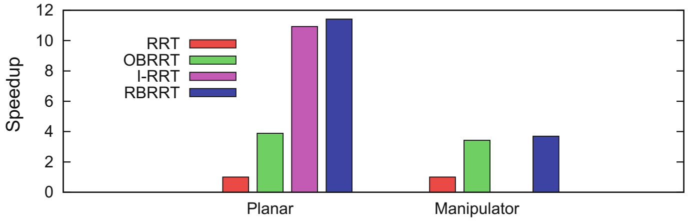


Fig. 8 Speedup of various RRT techniques compared with RRT

narrow passages made growth difficult. In contrast, the user directed the expansion of the tree through the passages in the interactive approaches. Compared with RRT, both I-RRT and RBRRT had speedups of up to 12 times in `Planar`, and RBRRT had speedups of up to four times in `Manipulator`. These results show the power of the interactive methods when compared to the fully automated approaches.

User Strategy. The user's strategy in both problems was to start the planner immediately and drag a single attract region through the environment, staying just ahead of the tree growth. This differed from the PRM-based strategies because RRTs always grow outward from the existing roadmap. This leading strategy allowed the user to effectively guide the RRT through narrow passages. The attract region used was roughly twice the size of the robot's base, which provided a good mix of flexibility and precision for steering the RRT. This is similar to the strategy used by I-RRT explaining their comparable performance in `Planar`.

5.3 Region-Biased Spark PRM

Setup and Results. Here, we compare Region-biased Spark PRM (referred to as RB Spark PRM) to Spark PRM [28]. Parameters for Region-biased Spark PRM and Spark PRM are identical and based upon recommendations in [28]. Both methods use Euclidean distance, straight-line local planning, and a $k = 10$-closest neighbor connection strategy.

Each planner is run until either a construction query is solved (identical to the Region-biased PRM experiments) or 5,000 nodes are sampled. Success rates are shown in Table 3 and speedups compared with Spark PRM are shown in Fig. 9.

Table 3 Success rates for Region-biased Spark PRM and Spark PRM

	Spark PRM(%)	RB Spark PRM(%)
`Planar`	90	100
`Manipulator`	100	100

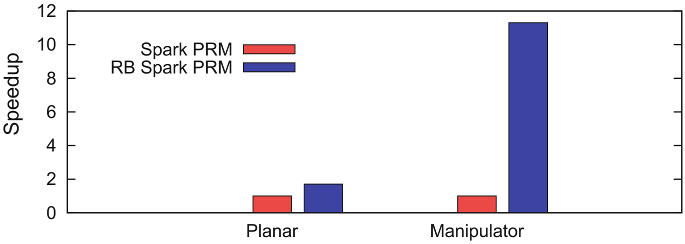


Fig. 9 Speedup comparison between Region-biased Spark PRM and Spark PRM

Performance Comparison. In this experiment, we can see that RB Spark PRM was able to focus Spark PRM's planning process effectively to provide more consistent and faster results, up to nine times speedup. The interactivity of these methods and our framework is extensible enough to speedup very efficient hybrid approaches to planning.

User Strategy. The user's strategy for RB Spark PRM was similar to that of Region-biased PRM. The primary difference in `Manipulator` was that fewer attract regions were required as the sparked RRTs quickly bridged unconnected components of the roadmap.

6 Conclusion

In conclusion, we presented a general region-based framework for collaborative motion planning. We examined three variants of this approach, designed for PRMs, RRTs, and hybrid PRM and RRT methods, respectively. As our results show, collaborative planning provides more consistent and efficient results compared with other interactive planning methods and fully automated approaches. In the future, we would like to gain a deeper theoretical understanding of this phenomena and see how this would extend past the classical planning problem into more difficult variants of motion planning.

Acknowledgements We would like to thank Brennen Taylor, a student at Texas A&M University, Ariana Ramirez, a student at Jimmy Carter Early College high school in La Joya, TX, USA, and Jonathon Colbert, a student at A&M Consolidated high school in College Station, TX, USA, for their participation in a summer research study on this topic in 2014.

This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-1423111, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11.

Jory Denny contributed to this research as a Ph.D. student at Texas A&M University. During this time, he was supported in part by an NSF Graduate Research Fellowship.

References

1. Amato, N.M., Bayazit, O.B., Dale, L.K., Jones, C., Vallejo, D.: OBPRM: an obstacle-based PRM for 3D workspaces. In: Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR ‘98), pp. 155–168. A. K. Peters, Ltd., Natick, MA, USA (1998)
2. Bayazit, O.B., Song, G., Amato, N.M.: Enhancing randomized motion planners: Exploring with haptic hints. In: Proceedings of IEEE International Conference on Robotics and Automation. (ICRA), pp. 529–536 (2000)
3. Boor, V., Overmars, M.H., van der Stappen, A.F.: The Gaussian sampling strategy for probabilistic roadmap planners. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1018–1023 (1999)
4. Buss, A., Harshvardhan, Papadopoulos, I., Pearce, O., Smith, T., Tanase, G., Thomas, N., Xu, X., Bianco, M., Amato, N.M., Rauchwerger, L.: STAPL: Standard template adaptive parallel library. In: Proceedings of Annual Haifa Experimental Systems Conference (SYSTOR), pp. 1–10. ACM, New York, NY, USA (2010). doi:10.1145/1815695.1815713
5. Denny, J., Sandstrom, R., Julian, N., Amato, N.M.: A region-based strategy for collaborative roadmap construction. In: Proceedings of International Workshop on Algorithmic Foundations of Robotics (WAFR). Istanbul, Turkey (2014)
6. Guo, C., Tarn, T., Xi, N., Bejczy, A.: Fusion of human and machine intelligence for telerobotic systems. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 3110–3115 (1995)
7. Hokayem, P.F., Spong, M.W.: Bilateral teleoperation: an historical survey. Automatica **42**, 2035–2057 (2006)
8. Hsu, D., Jiang, T., Reif, J., Sun, Z.: Bridge test for sampling narrow passages with probabilistic roadmap planners. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 4420–4426 (2003)
9. Hsu, D., Latombe, J.C., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robot. Res. **25**, 627–643 (2006)
10. Hwang, Y., Cho, K., Lee, S., Park, S., Kang, S.: Human computer cooperation in interactive motion planning. In: Proceedings of IEEE International Conference on Advanced Robotics (ICAR), pp. 571–576 (1997)
11. Ivanisevic, I., Lumelsky, V.: Augmenting human performance in motion planning tasks- the configuration space approach. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 2649–2654 (2001)
12. Ivanisevic, I., Lumelsky, V.J.: Configuration space as a means for augmenting human performance in teleoperation tasks. IEEE Trans. Syst. Man Cybern. Part B Cybern. **30**(3), 471–484 (2000)
13. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
14. Kuffner, J.J., LaValle, S.M.: RRT-connect: an efficient approach to single-query path planning. In: Proceedings of IEEE International Conference on Robotics and Automatic (ICRA), pp. 995–1001 (2000)
15. KUKA: http://www.youbot-store.com
16. Ladeveze, N., Fourquet, J.Y., Puel, B., Taix, M.: Haptic assembly and disassembly task assistance using interactive path planning. In: Virtual Reality Conference, 2009. VR 2009. IEEE, pp. 19–25 (2009)
17. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5), 378–400 (2001)
18. Lee, S., Sukhatme, G., Kim, G.J., Park, C.M.: Haptic teleoperation of a mobile robot: a user study. Presence Teleoperat. Virtual Environ. **14**(3), 345–365 (2005)
19. Lien, J.M., Pratt, E.: Interactive planning for shepherd motion. The AAAI Spring Symposium (2009)

20. Lin, M.C.: Efficient collision detection for animation and robotics. Ph.D. thesis, University of California, Berkeley, CA (1993)
21. Lozano-Pérez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM **22**(10), 560–570 (1979)
22. Masone, C., Franchi, A., Bulthoff, H.H., Giordano, P.R.: Interactive planning of persistent trajectories for human-assisted navigation of mobile robots. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 2641–2648 (2012)
23. Morales, M., Tapia, L., Pearce, R., Rodriguez, S., Amato, N.M.: A machine learning approach for feature-sensitive motion planning. In: Algorithmic Foundations of Robotics VI, Springer Tracts in Advanced Robotics (WAFR '04), pp. 361–376. Springer, Berlin (2005)
24. Plaku, E., Bekris, K.E., Chen, B.Y., Ladd, A.M., Kavraki, L.E.: Sampling-based roadmap of trees for parallel motion planning. IEEE Trans. Robot. Autom. (2005)
25. Plaku, E., Kavraki, L., Vardi, M.: Motion planning with dynamics by a synergistic combination of layers of planning **26**(3), 469–482 (2010)
26. Reif, J.H.: Complexity of the mover's problem and generalizations. In: Proceedings of IEEE Symposium Foundations of Computer Science (FOCS), pp. 421–427. San Juan, Puerto Rico (1979)
27. Rodriguez, S., Tang, X., Lien, J.M., Amato, N.M.: An obstacle-based rapidly-exploring random tree. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2006)
28. Shi, K., Denny, J., Amato, N.M.: Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA). Hong Kong, P. R. China (2014)
29. Singh, A.P., Latombe, J.C., Brutlag, D.L.: A motion planning approach to flexible ligand binding. In: International Conference on Intelligent Systems for Molecular Biology (ISMB), pp. 252–261 (1999)
30. Taïx, M., Flavigné, D., Ferré, E.: Human interaction with motion planning algorithm. J. Intel. Robot. Syst. **67**(3–4), 285–306 (2012)
31. van den Berg, J.P., Overmars, M.H.: Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. Int. J. Robot. Res. **24**(12), 1055–1071 (2005)
32. Vargas Estrada, A., Lien, J.M., Amato, N.M.: Vizmo++: a visualization, authoring, and educational tool for motion planning. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 727–732 (2006)
33. Yan, Y., Poirson, E., Bennis, F.: Integrating user to minimize assembly path planning time in plm. Product lifecycle management for society. In: IFIP Advances in Information and Communication Technology, vol. 409, pp. 471–480. Springer, Berlin (2013)
34. Zhang, L., Manocha, D.: An efficient retraction-based RRT planner. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2008)

Robotic Invention: Challenges and Perspectives for Model-Free Design Optimization of Dynamic Locomotion Robots

Luzius Brodbeck, Simon Hauser and Fumiya Iida

1 Introduction

The performance of a robot at a certain task depends on the robot's body structure and control. It has been shown that an appropriate body structure can greatly simplify the control problem [10]. However, the initial design of a robot might not be the most suitable for a task and a more beneficial structures exists.

To repeatedly adapt a robot's shape to its task and optimize the robot's performance, two main challenges must be addressed. First, to adjust to a preferably large range of tasks, the robot must be able to assume diverse body shapes, ideally of different size and resolution. This is demanding, as the real-world fabrication processes and constraints must be considered, and not all parts of the design space can equally well be explored. Second, to continuously optimize its own shape, the robot must be able to evaluate its own performance and iteratively generate new designs based on the task and previous performance.

Reconfigurable and self-reconfigurable modular robots address this issue by possessing the ability to change their own body structure to better adapt to the task requirements [16]. The ability to change their physical shape enables self-reconfigurable robots to achieve tasks which might not be solvable with a fixed morphology [14]. The field of modular self-reconfigurable robotics employs mechatronic modules which can adapt the connectivity between themselves to change the

L. Brodbeck
Institute of Robotics and Intelligent Systems, ETH Zurich, 8092 Zurich, Switzerland
e-mail: luzius.brodbeck@mavt.ethz.ch

S. Hauser
Biorobotics Laboratory, EPFL—Ecole Polytechnique Fédérale de Lausanne,
1015 Lausanne, Switzerland
e-mail: simon.hauser@epfl.ch

F. Iida (✉)
Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK
e-mail: fi224@cam.ac.uk

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_33

overall structure [4, 6, 12]. Other solutions are provided by configurable systems which can adjust predefined components of the system, such as the compound eye robot by Lichtensteiger et al. [5]. A further approach is the synthesis of new structures from a suitable base material as demonstrated by Revzen et al. [11] using a robot equipped with hardening foam or our previous work using hot melt adhesives [1, 7].

It is shown in this paper, that by increasing the diversity of mechanical design through improved reconfigurability, robots can generate and implement nontrivial designs. The ability to explore intricate morphological designs also allows for the generation of more complex behaviors. This was achieved within a limited number of trial-and-error iterations, without the use of simulation tools. To implement such a process, sufficient manipulation dexterity is necessary to physically instantiate the diverse morphologies and the search method must be able to efficiently handle the large dimensionality of the design problem.

In our implementation, flexible assembly is employed to generate diverse robot morphologies, similar to the centralized generation of agents demonstrated by Weel et al. in simulation [15]. An evolutionary algorithm is applied directly to the encoded building process [3] of locomotion agents to vary their shapes and subsequently optimize the locomotion speed of physical agents in a model-free process. The results were obtained throughout five experiments with 100 candidate robots each. These experiments have previously been published in [2].

2 Processes and Outcome of the Experiments

The goal of this experiment is the morphological adaptation of physical robotic agents to a locomotion task through an evolutionary process. To iteratively adapt the locomotion agents, a “mother robot” can repeatedly assemble the agents from elementary modules. The details of each agent’s building process is encoded in its genotype. The population of candidate solutions can undergo evolution, which subsequently optimizes the fitness at the locomotion task.

In this section, Materials and Methods are introduced first, before an overview of the results is given. The experiments have previously been presented in [2], which also contains a more detailed description of the setup and all parameters.

2.1 Materials and Methods

The robotic arm shown in Fig. 1 is able to rotate and bond the active and passive elementary modules. These processes are parametrized, and each set of parameters results in specific outcome of the building process, i.e. a specific morphology of the locomotion agent. This morphology, together with its control parameters and the task

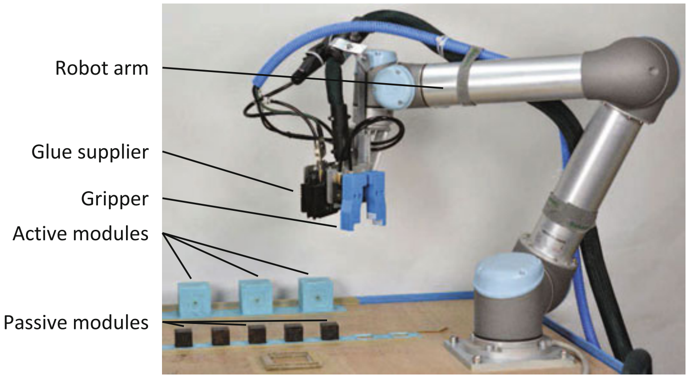


Fig. 1 The experimental setup with robotic arm (“mother robot”) and prepared active and passive modules

environment determine the agent’s performance, and thus its chances to be selected for further generations.

2.1.1 Hardware and Control

The robotic arm (Universal Robots, UR5) used is equipped with a pneumatic parallel gripper and a hot glue supplier. The gripper is used for the manipulation of the available modules, and the hot glue (ALFA Klebstoffe AG, ALFA H 5500/30) is used to bond the modules together.

The active modules are cubes with a side length of 6 cm and the passive modules are wooden cubes with 3 cm side length. The active modules contain a servo motor (Modelcraft, RS-3 JR) for actuation, a battery (Conrad Electronic AG, Conrad energy LiPo Akku 7.4 V, 800 mAh) and the electronics for control and wireless communication (Arduino, Pro Mini; Sparkfun, Bluetooth Mate Silver). One side of the cube is connected to the motor flange, such that it can be oscillated. Active and passive modules are supplied at predefined positions for the assembly of each agent.

For the rotation of active modules, a centering frame is mounted in the construction space to avoid that position errors sum up during repeated manipulations. The assembly is performed on a slightly adhesive and soft ground (3 mm foam rubber, covered with masking tape sticky side up) to ensure good ground contact and some error tolerance. In the testbed, three different ground surfaces are tested: plywood covered with fabric, carpet and polyurethane foam.

The experiment is controlled from the main controller on a desktop PC using Matlab. A TCP/IP connection is used for communication. The robot controller receives the command sequence from the main controller and executes it step by step. For the evaluation, the main controller sends the commands to the active modules using a Bluetooth connection.

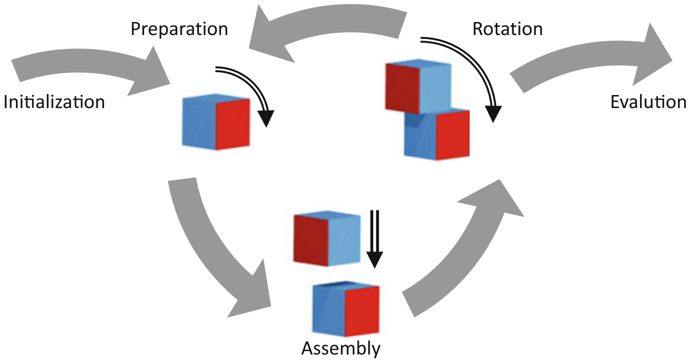


Fig. 2 Translation of the encoding into the building process based on three operations. For the preparation of a module, it is rotated. During assembly, the prepared module is bonded to the previously built structure, which in the last operation can undergo rotation as a whole

2.1.2 Evolutionary Process

All candidate locomotion agents are physically assembled from the modules and tested. To achieve a sufficiently large design space, the building process must be able to handle diverse solutions. To maintain the buildability for many parameter values, the fabrication process is structured into a fixed operation sequence. The parameter values are stored in an agent's genotype, which contains one gene per module, with each gene holding the parameters for the addition of one module. In Fig. 2, the encoded building process is illustrated. The three operations are the preparation of a module, assembly and the rotation of the structure.

For its preparation, a module is picked from the storage position and rotated around the global y and z-axes. Afterwards, the prepared module can be connected from the top to the previously built structure using the hot glue [13]. The assembled structure is then rotated again around the y and z-axes. After the rotation is terminated, either the next module is prepared and added, or the fabrication is concluded and the finished agent placed in the testbed for its evaluation. In the case an active module is added, its gene also defines the motor's amplitude and phase shift during the evaluation period.

Each gene contains the following fields defining the parameters of the fabrication process described above: Type of module, rotations during module preparation, rotations of previously built structure, relative offset at placement about x and y-axes, selection of attachment area in case of multiple options, a final rotation parameter executed after the building process and in case of active modules the control parameters amplitude and phase shift.

For the fitness evaluation of each agent, it is automatically placed in a prepared testbed by the robotic manipulator once its construction is finished. There, the motors are activated with the encoded control parameters for a fixed testing time. The behavior of the agent during the testing phase is recorded by an overhead camera. From the recorded footage, the position of the agent at the beginning and end of the test is extracted using computer vision techniques, and the distance travelled by the agent, divided by the testing time serves as a fitness measure.

After the fitness is evaluated for all agents of one generation, the genotypes of the next generation can be generated. An elite (usually the fittest three) advances to the next generation without any change to their genotype, to preserve this information. The other slots in the following generation are filled through mutation and recombination of genotypes. It is randomly determined for each new genotype, which mechanism is applied. For the mutation, one parent is selected, for recombination two parent genotypes are required. The selection in both cases is stochastic, with the selection probability for each genotype of the preceding generation proportional to its fitness.

Mutation can either add a new (randomly initialized gene), delete one gene from the genome or randomly change a parameter in a gene. It is probabilistically determined how many and which kind of mutation is performed. For the recombination, a one-point crossover scheme is applied. This combines the first n genes of the first parent with the last m genes of the second parent. Both integers n and m are randomly selected.

The physical implementation of candidate solutions introduces a number of constraints, mostly related to the specific implementation of the setup. For example the parallel gripper has a limited holding force, and the robotic arm's range is bounded. To minimize the time spent on candidate solutions which will violate one of these constraints, or are otherwise prone to fail (e.g. do not contain a single motor), a validation step is introduced. It checks each genotype for a number of elementary conditions. If the genotype fails at least one condition, it is regenerated. Conditions leading to the exclusion of a genotype are:

- Lack of stability during construction
- Servo-shafts colliding with other components
- Less than one or more than five elements
- Less than one or more than three active modules.

2.1.3 Experiment Details

Five experiments were performed, resulting in the instantiation and evaluation of 500 candidate solutions. Each experiment consisted of ten generations with ten agents each. Some parameters were varied between the experiments. The primary differences and parameters are indicated in this section. The complete specifications can be found in [2]. Unless specified otherwise, all experiments were randomly initialized, with genomes of one to three genes length.

Experiment 1a The first experiment was performed on the hard ground (plywood), with four instead of two rotations in the preparation and rotation operations. The final rotation of the agent was disabled and in the validation step, only the size limits were active.

Experiment 1b The agents were evaluated on the carpet. The motor amplitudes were restricted to 10°, 20° and 40°.

Experiment 1c To examine pure morphological adaptation, the motor control values (amplitude and phase shift) were fixed during this experiment. The agents were also evaluated on the carpet.

Experiment 1d Motor control was reactivated as an evolutionary parameter with the restricted parameter set from experiment 1b. Agents were evaluated on the polyurethane foam.

Experiment 2 To further increase the achievable morphological complexity, multiple parameters were adapted in this experiment and some manual interventions accepted. Successful agents from the previous experiments were selected for the initial population. In the validation step, the stability condition and collision detection were disabled. Consequently, a human operator had to assist to guarantee stability, and colliding motors were manually disabled. Furthermore, the more significant add and delete mutations were preferred over simple parameter changes.

2.2 Results

Throughout the experiments a large variety of locomotion robots were built and tested, which developed different successful locomotion strategies. A selection of successful agents from different experiments is shown in Fig. 3.

The stochastic optimization based on the evolutionary algorithm described in Sect. 2.1 optimizes the overall locomotion speed of the robotic agents. The increase of the resulting fitnesses over ten generations is documented in Fig. 4, which shows for each generation the mean of the best three agents in the population. Because of the real-world implementation, the evaluation is not deterministic and although elitism is applied, there is no guarantee that every generation reaches the previous fitness. However, over generations the fitness increased in all five experiments.

All agents of experiment 1c are shown in Fig. 5. This experiment is particularly interesting, as the motor control parameters were not subject to the evolutionary

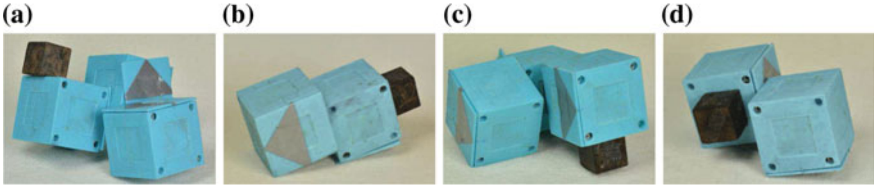


Fig. 3 Four sample locomotion agents generated by real-world evolution. All shown morphologies were amongst the most successful robots in their experiment

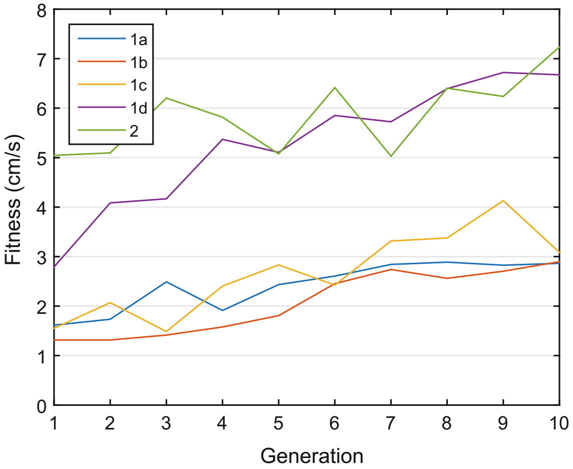


Fig. 4 Mean fitness of best three agents per generation of all five experiments. An improvement of fitness over ten generations can be found in all five experiments

optimization. Therefore, the fitness improvement was solely achieved through adaptations to the morphology of the locomotion agents. It also shows that despite the validation step, for a few agents the building process failed, with negative error codes indicating the reason (−13: glue connection failure, −14: collision during assembly, −16: other). Over all five experiments, the fabrication success rate was approximately 96%.

3 Design Diversity and Evolutionary Dynamics

To adapt the locomotion agents to different environments and explore different behaviors, diverse designs have to be generated and implemented. The encoding of designs and the fabrication process are closely coupled and largely define the design space. After addressing the initialization of diverse designs, the evolutionary process iterating on the designs must be set up such that it can maintain this diversity over generations to further explore additional solutions in the design space.

3.1 Encoding of Morphological Variations

Although a flexible assembly process is employed for the instantiation of the locomotion agents, the generation of morphological variation is not trivial. The fabrication constraints restrict the admissible ranges for parameter values. Therefore, not all regions of the design space are equally well reachable, which reduces the diversity. Furthermore, the ranges for parameter values must be set a priori and cannot depend on other values as this would conflict with mutation and crossover processes.

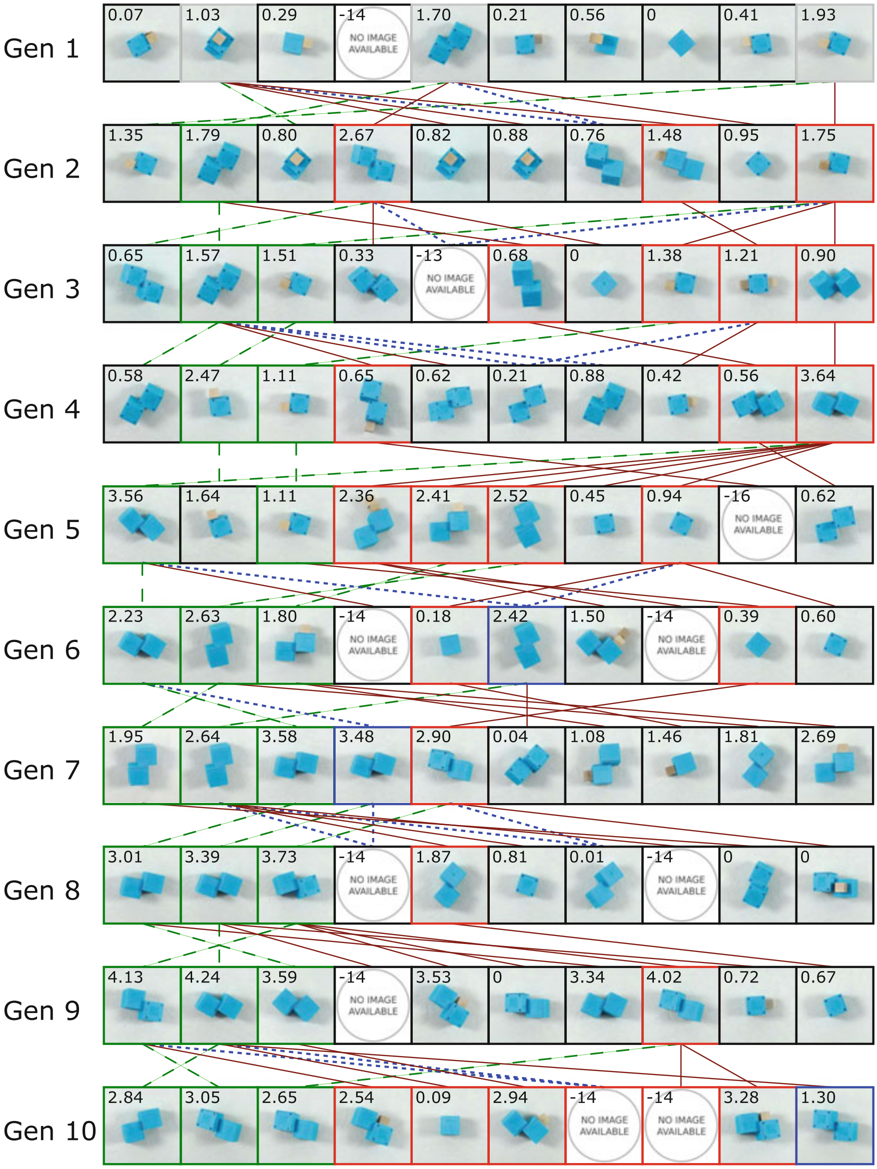


Fig. 5 Generation map of experiment 1c. In this experiment, the motor control parameters were not subject to the optimization, forcing the evolutionary process to improve the locomotion speed solely by morphological adaptation. The locomotion agents were evaluated on the carpet. The number with each agent indicates its fitness (cm/s) and the colors indicate the generation method (*green*: elite, *red*: mutation, *blue*: crossover). Negative fitnesses are the error codes for failed agents

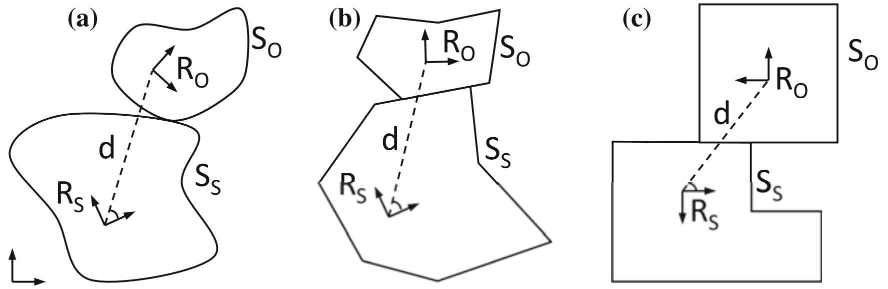


Fig. 6 Encoding generalization. Three different scenarios were considered for the attachment of an object O to a structure S. In the general case **a**, two bodies of arbitrary shape are connected by at least one contact point. For a physical realization **b**, a sufficiently large contact area is required. Therefore, in this scenario, both bodies are assumed to be polyhedra. Given the assembly constraints from the real-world experiment **c**, both bodies are from elementary cubic shapes. All illustrations are 2D, but the approach readily applies to the general 3D case

The modules employed in the experiments are of cubic shape, which simplifies the attachment process and thus the encoding of the genotypes, especially the definition of parameter ranges. The influence of shape and attachment constraints on the generation of diverse morphologies is analyzed in the following sections.

3.1.1 The General Attachment Problem

The goal is to attach an object O with shape S_{O} on a structure S with shape S_{S} as illustrated in Fig. 6. The rotation of the object is given by a rotation matrix R_{O}, and the rotation of the structure S is defined by the rotation matrix R_{S}. For the attachment, at least one contact point between the shapes S_{S} and S_{O} must be present without any overlap of the respective shapes. Therefore, there is a limited set of valid attachment vectors $\Gamma(\alpha, d)$, which is defined by the direction angle α and the distance d between the structure and the object. Given an angle α, the distance d is determined by the geometry of the problem:

$$d = f(S_{\mathrm{S}}, S_{\mathrm{O}}, R_{\mathrm{S}}, R_{\mathrm{O}}, \alpha) \ . \tag{1}$$

For this general attachment problem—assuming a point contact is sufficient to connect the two bodies—all parameters but the distance d can be freely chosen. Structure and object can have arbitrary shape and orientation, only the distance depends on the other parameters to fulfill the geometrical constraints for attachment as illustrated by the function f in (1).

3.1.2 Attachment of Flat Surfaces

However, for the practical realization of attachment, point contacts are not sufficient. For the connection with HMA, for example, both bodies must be in contact with a large enough attachment area $A \geq A_{\min}$. To realize this, in the next step it is assumed that both shapes S_{S} and S_{O} are polyhedra (the set of polyhedra here is denoted as Π). For the two-dimensional illustrations in Fig. 6, polygons are used. For attachment, one surface of each polyhedron must be brought into contact, which requires a parallel orientation of the surfaces. Given the shapes of both bodies $S_{\mathrm{S}}, S_{\mathrm{O}} \in \Pi$ and the orientation R_{S} of the structure, only a limited set of orientations R_{O} of the object is admissible. The choice of the object orientation further constrains the set of valid attachment vectors Γ, and also the angle α can no longer be freely chosen:

$$S_{\mathrm{S}}, S_{\mathrm{O}} \in \Pi \tag{2}$$

$$R_{\mathrm{O}} \in g(S_{\mathrm{S}}, S_{\mathrm{O}}, R_{\mathrm{S}}) \tag{3}$$

$$\alpha \in h(S_{\mathrm{S}}, S_{\mathrm{O}}, R_{\mathrm{S}}, R_{\mathrm{O}}, A_{\min}) \tag{4}$$

$$d = f(S_{\mathrm{S}}, S_{\mathrm{O}}, R_{\mathrm{S}}, R_{\mathrm{O}}, \alpha) \ . \tag{5}$$

The functions g and h which define the admissible set of rotations R_{O} and angles α are not necessarily easy to determine, depending on the geometry of the problem.

3.1.3 Practical Attachment of Cubic Shapes

For the practical attachment based on the presented experiments a set of cubic shapes Σ with side lengths s and $2s$ is considered. It is assumed the object is of such shape ($S_{\mathrm{O}} \in \Sigma$). The structure's shape is a combination of elementary cubes, i.e. $S_{\mathrm{S}} \in \hat{\Sigma} \supset \Sigma$.

Based on the body shapes and fabrication processes, additional constraints are introduced. Both bodies' rotations are restricted to a multiple of ±90° around the elementary axes. For the attachment, only the topmost surface of the structure is considered, restricting the admissible values of the direction angle α:

$$S_{\mathrm{S}} \in \hat{\Sigma} \supset \Sigma \tag{6}$$

$$S_{\mathrm{O}} \in \Sigma \tag{7}$$

$$R_{\mathrm{S}}, R_{\mathrm{O}} \in R_{\mathrm{x}}\left(k_{\mathrm{x}}\frac{\pi}{2}\right) + R_{\mathrm{y}}\left(k_{\mathrm{y}}\frac{\pi}{2}\right) + R_{\mathrm{z}}\left(k_{\mathrm{z}}\frac{\pi}{2}\right) , \quad k \in \mathbb{Z} \tag{8}$$

$$\alpha \in h'(S_{\mathrm{S}}, S_{\mathrm{O}}, R_{\mathrm{S}}) \tag{9}$$

$$d = f'(S_{\mathrm{S}}, S_{\mathrm{O}}, R_{\mathrm{S}}, \alpha) \ . \tag{10}$$

This is simplifies the problem in many ways as compared to the previous problem discussed in Sect. 3.1.2. The admissible values for the orientations are from a fixed set (8) as compared to the complex function g in (3), which depends on the problem geometry and the structure orientation. Furthermore, given the cubic shape of the object and the fact that only elementary rotations are considered, its overall shape is predefined, and thus does not have to be considered in the calculation of the attachment vector $\Gamma(\alpha, d)$ in Eqs. (9)–(10).

3.1.4 Real-World Fabrication Constraints

Apart from the shape and attachment mechanism, further system constraints have to be considered for the physical implementation of automated assembly processes. In the implementation presented in this paper, a validation step (Sect. 2.1.2) checks each genotype for a range of conditions to ensure most constraints are met.

To evaluate the effect of four main constraints of the physical assembly system, a simulation experiment was performed. 1.25 million genotypes were randomly generated with one to ten components. Their morphologies were built in simulation and based on the simulation results, they were checked for all of the four constraints. The constraints considered are the maximum agent weight, maximum agent dimensions, stability of agents during fabrication (no toppling) and the connection of new modules to the agent's topmost surface only (for details please refer to [2]).

In Fig. 7, the diversity of shape factors for a given number of components that was achieved by the simulated population is plotted. For the calculation of the diversity, the all agents were categorized based on their shape factor (see [2] for definition). The diversity is calculated as the effective number of types based on the population's Shannon index, an entropy measure [8]. The diversity measure takes into account the number of classes present in a population, as well as their relative abundance. The population was further categorized based on whether all four fabrication constraints are fulfilled, all but the stability constraint are fulfilled or none are fulfilled. The stability constraint is of particular interest, as it was relaxed in experiment 2.

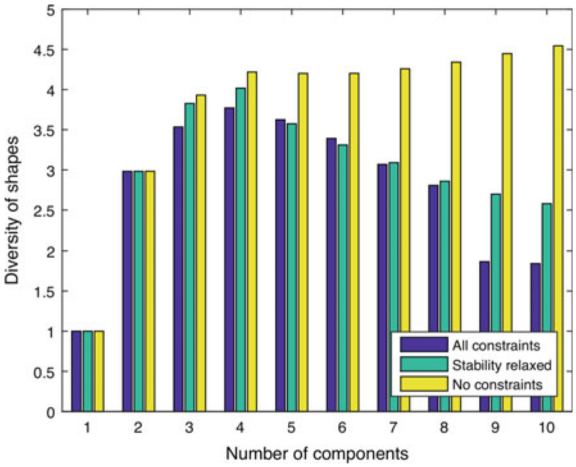


Fig. 7 Shape diversity for differently sized robots with different building constraints active. The diversities were obtained based on 1.25 million randomly generated genotypes and their corresponding morphologies calculated in simulation. A diversity value of 3.0 for example is equivalent to the diversity of a population with three equally abundant shape classes

The results show, that for small agents, the constraints have only a minor influence, as they are easily fulfilled. However, the constraints complicate the fabrication of large agents, and restrict their diversity. Therefore, to scale this approach to more complex scenarios, fabrication constraints must be carefully addressed.

3.2 Generating New Designs

After the evaluation of one generation in the real world is completed, the fitnesses of all candidate solutions are known. In a next step, the evolutionary algorithm needs to map the ten old genotypes of generation $n - 1$ to the ten new genotypes of generation n. The chosen process is a mixture of elitism, combined with random mutations and crossover. For the selection of parent genotypes, the selection probabilities are proportional to parent fitness.

In Fig. 8, the evolution of fitness in experiment 1c is shown, indicating the generation mechanism of new genotypes with color (green: elite, blue: crossover, red: mutation) and the relationships through lines from one to another generation. Crossover is based on two parent genotypes, the other mechanisms use a single parent. In the case of elitism, the child genotype is an exact copy of the parent genotype. However, because of the stochasticity in the real-world testing, also identical genotypes exhibit some fitness variation.

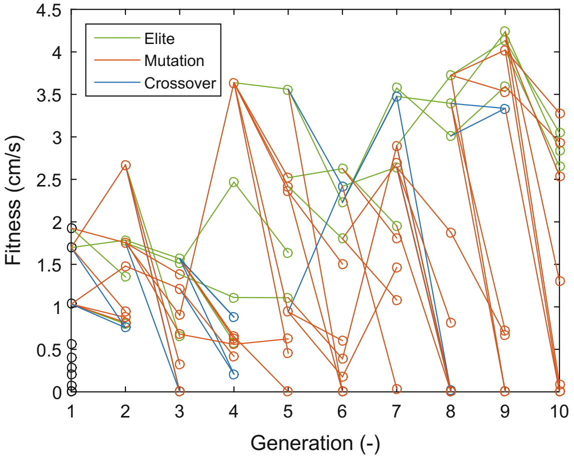


Fig. 8 Fitness evolution in experiment 1c on carpet. This figure shows the evolution of locomotion fitness through the course of one experiment. The color indicates the way each agent was generated (*green*: elite, *blue*: crossover, *red*: mutation). The graph shows the variance of identical genotypes due to real-world interactions (elite) as well as the increased fitness variation in positive and negative direction for new genotypes (mutation and crossover)

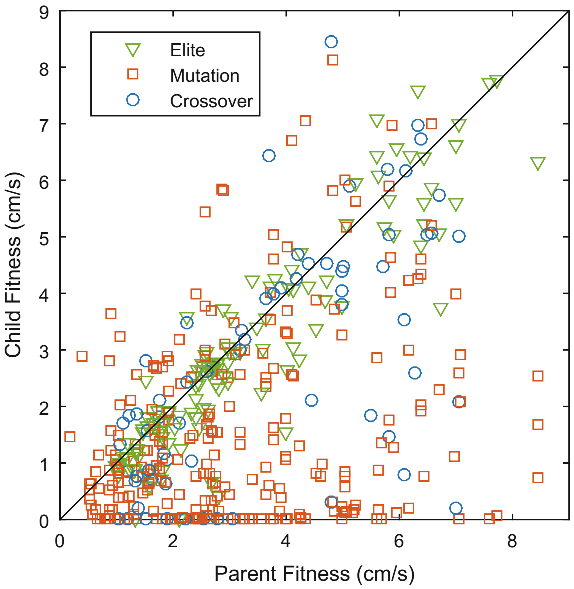


Fig. 9 Parent versus child fitness for all offsprings evaluated in the five experiments. The offsprings which are part of the elite mostly have comparable fitnesses to their parents, while mutated/crossed offsprings exhibit larger fitness variation. For crossover, the average fitness of parents is considered as the parent fitness. The marker color indicates the way each agent was generated (*green*: elite, *blue*: crossover, *red*: mutation)

Figure 9, which shows the parent versus child fitnesses over all five experiments, indicates that elitism (green triangles) results in child fitnesses comparable to the parent fitness as expected. Both, mutation (red boxes) and crossover (blue circles), produce a larger fitness variation. There is a chance that the child completely fails, but on the other hand, 30 offsprings were at least 50% better than their parents. While we are interested in the fitness optimization, there is no guarantee that the random changes of the genotypes result in a preferable outcome. However, through the combination with the selection strategy, the overall fitness increases through the course of evolution.

4 Analysis of Behavioral Diversity

The behavior of a robot emerges through the interactions of the robot's body and control with the task environment [10]. Most of the successful agents exhibit periodic behaviors, but some fit robots also showed more complex behaviors. The complete overview of the experiments shows that not only morphologies and behaviors are fine-tuned to the task, but also new morphologies (see Fig. 5) and behaviors are discovered by the evolutionary optimization.

For the fitness evaluation, only the start and end points of the robot trajectories were considered, but from the movies recorded by the overhead camera, the complete 2D trajectories can be extracted. In Fig. 10, a selection of such trajectories is plotted. The trajectories were selected from the top 10% of locomotion agents over all five experiments.

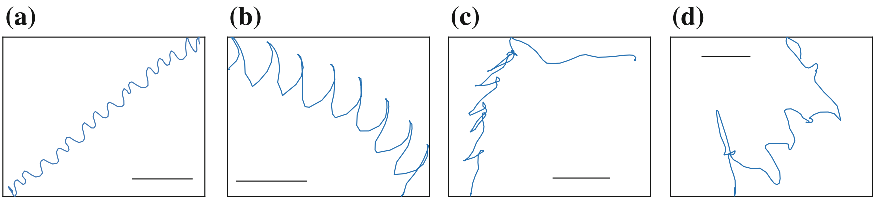
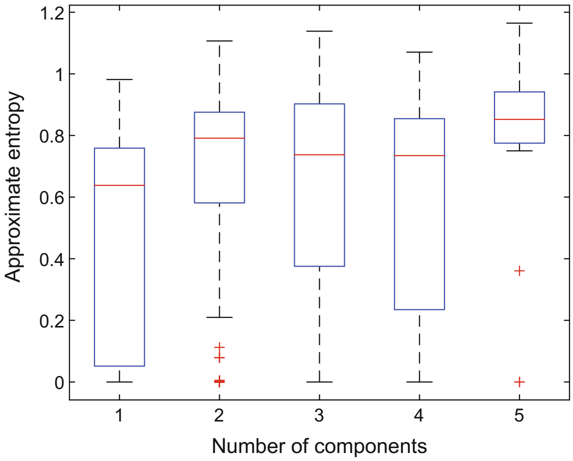


Fig. 10 These trajectories were selected from the 10% of fittest locomotion agents over all five experiments. The selected trajectories show that although most successful agents exhibit periodic motion patterns, also less structured behavior can be successfully developed. All scale bars measure 60 mm, i.e. one side length of an active module

Fig. 11 Correlation between morphological complexity (measured by the number of constituent components) and behavior complexity (measured by the approximate entropy (ApEn)) based on the real data from all five experiments



It is much harder to engineer complex, non-periodic behaviors than the steady solutions. To better understand under which conditions such innovations are more likely to develop, the behavioral complexity based on the agent's trajectories is further analyzed. To quantify the complexity of the agents' behaviors, the "Approximate Entropy" (ApEn) of their trajectories was calculated as described by Peng et al. (p. 11 in [9]), with the only difference, that the trajectory segments compared are normalized by their initial positions to account for the translations during the course of the robot's motion. The parameters used for the calculation of the ApEn are $m = 2$ and $r = 5$.

Assessing the influence of morphology onto behavior, a measure for morphological complexity is required. Here, we approximate an agents' morphological complexity by its number of constituent modules. Putting the number of modules and approximate entropy of all built locomotion agents into relation, it can be seen that larger agents tend to exhibit more complex trajectories as plotted in Fig. 11.

As shown already by the simulation results plotted in Fig. 7, it is a challenging task to develop diverse structures from many components, which at the same time fulfill the building constraints. On the other hand, the real-world results show the

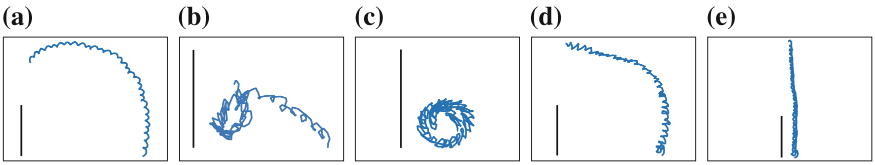


Fig. 12 Trajectories of an agent with fixed morphology and motor control in different environments. The tested grounds are: **a** plywood, **b** carpet, **c** soft foam, **d** fine sandpaper and **e** textile. The scale bar in all plots is 60 mm, i.e. the side length of an active module

benefits of larger structures to achieve complex behaviors. The fabrication process therefore has to be carefully implemented to take these considerations into account.

Apart from morphology and control (the parameters under control of the evolutionary algorithm), the testing environment has a large influence on an agent's behavior. In Fig. 12, the trajectories of one robotic agent with fixed control parameters in five different environments are shown. From these trajectories it can be seen that not only the performance varies, but also different behaviors emerge through the complex system-environment interactions.

Since the evaluation of the robotic agents is based on real-world tests, the results are not deterministic. The stochasticity can be introduced both in the fabrication and the evaluation steps. During the genotype-phenotype mapping, i.e. the fabrication of physical locomotion agents, small differences always occur, which influence the robot's morphology. The second source of uncertainty is the morphology-behavior mapping, i.e. the real-world evaluation of agents. The initial conditions and local details of the environment or internal parameters of the modules influence an agent's behavior in a stochastic way.

5 Conclusion

In this article, the design optimization of physical locomotion agents is presented. An evolutionary algorithm was applied directly onto the encoded fabrication process, which enabled the automatic implementation of candidate solutions in real-world for their performance evaluation in the task environment. For the successful optimization, a process is required which can generate diverse mechanical designs and autonomously generate new design based on the solution performance.

Over five experiments, 500 candidate solutions were built with about 96% success rate and subsequently tested. The evolutionary process led to a relevant increase of locomotion fitness over ten generations in all five experiments. After the fabrication the robot morphologies interact with the task environment. The emerging behavior determines the performance at the given task. It was shown that although many successful agents exhibit periodic motions, the automatic design can generate more complex working behaviors.

Analysis of these experiments emphasized the importance of the fabrication constraints for the physical implementation of the presented system. The fabrication constraints directly influence the diversity of designs which can successfully be constructed, especially if larger structures are considered. On the other hand, to achieve more complex and nontrivial behaviors, it is beneficial to fabricate larger structures—an ability which is directly influenced by the fabrication constraints.

References

1. Brodbeck, L., Iida, F.: An extendible reconfigurable robot based on hot melt adhesives. Auton. Robots **39**(1), 87–100 (2015). doi:10.1007/s10514-015-9428-1
2. Brodbeck, L., Hauser, S., Iida, F.: Morphological evolution of physical robots through model-free phenotype development. PLoS ONE **10**(6), e0128,444 (2015). doi:10.1371/journal.pone.0128444
3. Kuehn, T., Rieffel, J.: Automatically designing and printing 3-D objects with EvoFab 0.2. Artif. Life **13**, 372–378 (2012). doi:10.7551/978-0-262-31050-5-ch049
4. Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., Murata, S.: Distributed self-reconfiguration of M-TRAN III modular robotic system. Int. J. Robot. Res. **27**(3–4), 373–386 (2008). doi:10.1177/0278364907085560
5. Lichtensteiger, L., Eggenberger, P.: Evolving the morphology of a compound eye on a robot. In: Third European Workshop on Advanced Mobile Robots, pp. 127–134 (1999). doi:10.1109/EURBOT.1999.827631
6. Neubert, J., Lipson, H.: Soldercubes: a self-soldering self-reconfiguring modular robot system. Auton. Robots (2015). doi:10.1007/s10514-015-9441-4
7. Nurzaman, S.G., Culha, U., Brodbeck, L., Wang, L., Iida, F.: Active sensing system with in situ adjustable sensor morphology. PLoS ONE **8**(12), e84,090 (2013). doi:10.1371/journal.pone.0084090
8. Peet, R.K.: The measurement of species diversity. Annu. Rev. Ecol. Syst. **5**, 285–307 (1974)
9. Peng, Z., Genewein, T., Braun, D.A.: Assessing randomness and complexity in human motion trajectories through analysis of symbolic sequences. Front. Hum. Neurosci. **8**(168) (2014). doi:10.3389/fnhum.2014.00168
10. Pfeifer, R., Lungarella, M., Iida, F.: Self-organization, embodiment, and biologically inspired robotics. Science **318**(5853), 1088–1093 (2007). doi:10.1126/science.1145803
11. Revzen, S., Bhoite, M., Macasieb, A., Yim, M.: Structure synthesis on-the-fly in a modular robot. pp. 4797–4802 (2011). doi:10.1109/IROS.2011.6094575
12. Sproewitz, A., Moeckel, R., Vespignani, M., Bonardi, S., Ijspeert, A.: Roombots: a hardware perspective on 3D self-reconfiguration and locomotion with a homogeneous modular robot. Robot. Auton. Syst. **62**(7), 1016–1033 (2014). doi:10.1016/j.robot.2013.08.011
13. Wang, L., Iida, F.: Physical connection and disconnection control based on hot melt adhesives. IEEE/ASME Trans. Mechatron. **18**(4), 1397–1409 (2013). doi:10.1109/TMECH.2012.2202558
14. Wang, L., Brodbeck, L., Iida, F.: Mechanics and energetics in tool manufacture and use: a synthetic approach. J. R. Soc. Interface **11**(100) (2014). doi:10.1098/rsif.2014.0827
15. Weel, B., Crosato, E., Heinerman, J., Haasdijk, E., Eiben, A.: A robotic ecosystem with evolvable minds and bodies. pp. 165–172 (2014). doi:10.1109/ICES.2014.7008736
16. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot. Autom. Mag. **14**(1), 43–52 (2007). doi:10.1109/MRA.2007.339623

Part IV
Knowledge-Based Robotics and Knowledge

Session Summary

Bridging the gap between symbolic knowledge-based representations and control systems for robot systems acting in the real world is one of the major challenges for establishing truly autonomous systems that can reason about activities to be carried out and transform them into proper actions to be carried out. This session included two papers addressing the problem of using knowledge about objects or actions to infer trajectories or actions. It also included a paper studying the problem of building robots from functional specifications. In addition, authors presented an approach to a neuromorphic sense of touch for robotic fingertips. Two further papers are concerned with the identifiability analysis of frictional contacts as well as proper distance functions for belief-space planning.

The paper entitled "Robobarista: Object Part Based Transfer of Manipulation Trajectories from Crowd-Sourcing in 3D Pointclouds" by Jaeyong Sung, Seok Hyun Jin and Ashutosh Saxena provided an innovative approach to manipulation planning by exploiting the fact that many objects share similarly operated object parts. The authors provide an approach based on deep learning to determine manipulation trajectories based on past experience.

In the paper "Cloud-Based Probabilistic Knowledge Services for Instruction Interpretation" by Daniel Nyga and Michael Beetz, the authors present a framework for learning and reasoning about action-specific probabilistic knowledge bases. In this framework, the information about actions and objects is compactly represented by first-order probabilistic models, which are used to learn joint probability distributions for inferring the most probable executable instruction.

The paper "Neuromorphic Artificial Sense of Touch: Bridging Robotics and Neuroscience" by Udaya Bhaskar Rongala, Alberto Mazzoni, Domenico Camboni, Maria Chiara Carrozza and Calogero Maria Oddo presents a system that allows to code tactile information by means of a sequence of spikes, mimicking the neural dynamics of Merkel Slowly Adapting and Meissner Fast Adapting human mechanoreceptors. It furthermore presents a neuromorphic finger with a bio-inspired tactile sensor and describes a dedicated processing architecture to generate a neuromorphic output in the form of spikes. The experimental results show a high classification performance for the data obtained from sliding the finger over different types of textures.

The paper "Identifiability Analysis of Planar Rigid-Body Frictional Contact" presented by Nima Fazeli, Russ Tedrake and Alberto Rodriguez investigates the identifiability of the inertial parameters and the contact forces associated with an object making and breaking frictional contact with the environment. The paper demonstrates that given a time history of the kinematic measurements of the object without external force, the parameters ratio of mass moment of inertia to mass of the object as well as ratio of the tangential and normal forces to mass can be identified.

In the paper "Robot Creation from Functional Specifications", the authors Ankur M. Mehta, Joseph DelPreto, Kai Weng Wong, Scott Hamill, Hadas Kress-Gazit and Daniela Rus describe an integrated end-to-end system for rapidly creating printable robots from a Structured English description of desired behavior. The authors use linear temporal logic and a modular component library to allow the automatic synthesis of complete mechanical, electrical and software designs.

Finally, the paper entitled "The Importance of a Suitable Distance Function in Belief-Space Planning" by Zakary Littlefield, Dimitri Klimenko, Hanna Kurniawati and Kostas E. Bekris investigates the problem of finding a suitable distance function for belief-space planning. It considers the Wasserstein distance, the Hausdorff Distance, the KL-Divergence and the L1 Distance and compares them using a sampling-based framework on two popular problem classes. Based on the experimental data, the paper comes to the conclusion that the Wasserstein distance in belief-space planning provides significant improvements over the considered alternatives.

A Complete Algorithm for Generating Safe Trajectories for Multi-robot Teams

Sarah Tang and Vijay Kumar

1 Introduction

Multi-robot systems have become attractive solutions for a wide variety of tasks. One prominent initiative is the Amazon Prime Air project [5], which proposes using autonomous Unmanned Aircraft Systems (UASs) to deliver packages under five pounds to customers within a ten mile radius of a fulfillment center in less than 30 min. In this setting, hundreds to thousands of robots could be in the air simultaneously. Each robot is assigned a fixed and non-interchangeable goal, or *labeled*.

While it may seem promising to simply stagger the UASs' altitudes, recent Federal Aviation Administration (FAA) proposed guidelines [4] limit the maximum altitude of these small UASs to 400 ft, essentially confining vehicles to the horizontal plane. Thus, this work focuses on finding safe motion plans for robots operating in a two-dimensional space. This problem is harder than the three-dimensional version, because the latter provides an additional degree of freedom.

There are, broadly speaking, three guarantees of interest for planning algorithms: *safety* — robots will be collision-free with obstacles and each other, *optimality* — the solution is minimum cost, and *completeness* — the planner will always find a solution if one exists and indicate that there is none if one does not.

Approaches to the labeled multi-robot planning problem can be characterized as *coupled* or *decoupled*. *Coupled* planners search for optimal paths in the joint configuration space of all team members, either by directly applying planning algorithms such as A* [7] or with specialized variants [6]. These approaches typically guarantee optimality and completeness. However, as the search space grows exponentially with the number of robots, they quickly become computationally impractical.

S. Tang (✉) · V. Kumar
University of Pennsylvania, Philadelphia, PA, USA
e-mail: sytang@seas.upenn.edu

V. Kumar
e-mail: kumar@seas.upenn.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_34

Decoupled planners, on the other hand, plan for each robot separately. One approach is to plan each robot's motion in priority order. Lower priority robots must avoid higher priority ones [1, 3]. An alternative is to first find paths that avoid static obstacles, then design velocity profiles that avoid inter-robot collisions [9, 11]. These planners tend to be faster, but are typically not complete.

As a result, algorithms that combine both approaches have been proposed. van den Berg et al. [17] decouple the problem into smaller coupled subproblems, minimizing the dimensionality of the highest-dimensional subproblem. Subdimensional expansion [18] first plans in each robot's individual configuration space and searches a joint state space in regions where collisions occur. These approaches offer significant computational improvements, but can still perform poorly in the worst case.

Other planning approaches include rule-based [2] or network flow [19] algorithms. Alternatively, van den Berg et al. [16] frame the problem as a reciprocal collision avoidance problem. In air traffic control, Tomlin et al. [13] find safe conflict resolution maneuvers in the presence of uncertainties. However, this approach requires computation of solutions to the Hamilton–Jacobi–Isaacs PDE equation, which becomes computationally difficult for large teams.

Other settings allow for robots to be completely interchangeable. Proposed solutions to the *unlabeled* multi-robot planning problem must solve both the task assignment and trajectory generation problems [14]. In particular, Turpin et al. propose an $O(N^3)$ solution to the unlabeled planning problem in obstacle-free environments [14] for teams of N robots.

Our work proposes a centralized algorithm for finding collision-free trajectories for a team of labeled robots operating in an obstacle-free two-dimensional workspace. In essence, each robot pursues its own optimal motion plan until an impending collision is detected. This causes the affected robots to enter a holding pattern, similar to the racetrack patterns used in civilian aviation in congested airspace. Our approach is similar to subdimensional expansion, however, collisions are resolved through analytically constructed maneuvers as opposed to a high-dimensional graph search. While this is suboptimal, our algorithm offers completeness guarantees and allows for greater scalability to large teams.

The remainder of this paper will proceed as follows. Section 2 presents terminology and Sect. 3 discusses a known solution to the unlabeled multi-robot planning problem. Section 4 details our algorithm for the labeled problem and discusses its safety and completeness guarantees. Section 5 characterizes the algorithm's performance through simulation experiments and Sect. 6 presents conclusions and directions for future work.

2 Problem Definition

Let $\mathcal{I}_Z = \{1, 2, \ldots, Z\}$ denote the set of integers between 1 and Z, inclusive. Let N denote the number of robots in the team. We represent the start and goal positions of robot $i \in \mathcal{I}_N$ with $\mathbf{s}_i \in \mathbb{R}^2$ and $\mathbf{g}_i \in \mathbb{R}^2$, respectively, and the sets of all start and goal positions with S and G, respectively. $\mathbf{x}$ denotes a position in $\mathbb{R}^2$ and $\mathbf{x}_i$ denotes the state of robot i. Each robot has identical first-order dynamics:

$$\dot{\mathbf{x}}_i(t) = \mathbf{u}_i(t), \qquad \|\mathbf{u}_i(t)\|_2 \leq v_{max} \tag{1}$$

In the centralized paradigm, each robot knows the states and goals of all robots.

We define a trajectory as a piecewise smooth function of time, $\gamma(t) : [t_0, t_f] \rightarrow \mathbb{R}^2$. Let $\gamma^{(\mathbf{x}_0,\mathbf{x}_1)}(t)$ denote an optimal trajectory between $\mathbf{x}_0$ and $\mathbf{x}_1$ and $\gamma_i(t)$ denote robot i's trajectory between $\mathbf{s}_i$ and $\mathbf{g}_i$. Let $\gamma(t)$ denote the set of all trajectories $\gamma_i(t)$.

We model each robot as a disk of radius R. We use $\mathscr{B}(\mathbf{x}_i(t))$ to denote the area robot i occupies at $\mathbf{x}_i(t)$ and $\mathscr{B}(\gamma_i)$ to denote the area it sweeps out traversing $\gamma_i(t)$.

The goal of the labeled planning problem is to plan a trajectory $\gamma_i(t)$ for each robot such that $\gamma_i(0) = \mathbf{s}_i$, $\gamma_i(t_{f,i}) = \mathbf{g}_i$. All robots' trajectories start simultaneously but each robot can reach its goal at a unique $t_{f,i}$. We assume robots remain stationary at their goals for all $t > t_{f,i}$, and we require $\mathscr{B}(\mathbf{x}_i(t)) \cap \mathscr{B}(\mathbf{x}_j(t)) = \emptyset$ for all $t \in [0, \max_{i \in \mathscr{I}_N} t_{f,i}]$, $j \neq i \in \mathscr{I}_N$.

3 Concurrent Assignment and Planning of Trajectories (CAPT)

First, consider the unlabeled planning problem: given N robots and M goals, plan a trajectory $\gamma_i(t)$ for each robot such that each goal is visited by one robot. When $M > N$, some goals will remain unvisited while when $M < N$, some robots will not visit any goals. In this section, we outline the Concurrent Assignment and Planning of Trajectories (CAPT) algorithm [14] to solve this problem.

Suppose the start and goal locations are at least $2\sqrt{2}R$ away from each other:

$$\|\mathbf{s}_i - \mathbf{s}_j\|_2 > 2\sqrt{2}R \quad \forall i \neq j \in \mathscr{I}_N, \qquad \|\mathbf{g}_i - \mathbf{g}_j\|_2 > 2\sqrt{2}R \quad \forall i \neq j \in \mathscr{I}_M \tag{2}$$

Define the assignment mapping robots to goals as $\phi : \mathscr{I}_N \rightarrow \mathscr{I}_M \cup 0$, where $\phi_i = j$ indicates that robot i is assigned to goal j and $\phi_i = 0$ if robot i is unassigned. The CAPT algorithm finds the assignment and trajectories that solve:

$$\min_{\phi, \gamma(t)} \sum_{i=1}^{N} \int_0^{t_f} \dot{\mathbf{x}}_i(t)^T \dot{\mathbf{x}}_i(t) dt \tag{3}$$

The solution to this problem consists of straight-line trajectories that minimize the sum of the squares of the distances traveled. In other words, the optimal assignment is given by:

$$\phi^\star = \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{s}_i - \mathbf{g}_{\phi_i}\|_2^2 \tag{4}$$

This assignment can be found in $O(N^3)$ time using the Hungarian Algorithm [10].

Denote the assigned goal of robot i with $\mathbf{g}_i^\star$, where $\mathbf{g}_i^\star = \mathbf{s}_i$ if robot i is unassigned and $\mathbf{g}_i^\star = \mathbf{g}_{\phi_i^\star}$ otherwise. The optimal trajectories are the constant velocity straight-line trajectories from $\gamma_i(0) = \mathbf{s}_i$ to $\gamma_i(t_f) = \mathbf{g}_i^\star$. We want all robots to arrive at their goals simultaneously at t_f, which can be found with:

$$t_f = \max_{i \in \mathscr{I}_N} \frac{\|\mathbf{s}_i - \mathbf{g}_i^\star\|_2}{v_{max}} \tag{5}$$

We will refer to such trajectories as *synchronized*. Turpin et al. show these trajectories are collision-free [14].

4 Optimal Motion Plans + Circular HOlding Patterns (OMP+CHOP) for the Labeled Planning Problem

We now present our algorithm to solve the labeled planning problem. First, we discuss the best-case scenario, where all robots can move directly to their goals following an Optimal Motion Plan (OMP). Next, we consider the worst-case scenario, where all robots must enter a single Circular HOlding Pattern (CHOP). Finally, we describe the full algorithm, which combines these two strategies.

We again assume the start and goal positions satisfy the separation assumptions given in Eq. 2, however, in the labeled setting, $M = N$.

4.1 Optimal Motion Plans (OMPs)

Given any two waypoints and times of arrival at these points, we can design an optimal trajectory taking robot i from $\mathbf{x}_i(t_0) = \mathbf{x}_0$ to $\mathbf{x}_i(t_f) = \mathbf{x}_f$ by solving:

$$\begin{gathered} \gamma^{(\mathbf{x}_0, \mathbf{x}_f)}(t) = \underset{\gamma(t)}{\operatorname{argmin}} \int_{t_0}^{t_f} \dot{\mathbf{x}}_i(t)^T \dot{\mathbf{x}}_i(t) dt \\ \text{subject to: } \gamma(t_0) = \mathbf{x}_0, \;\; \gamma(t_f) = \mathbf{x}_f \end{gathered} \tag{6}$$

As before, the optimal trajectory is the constant-velocity straight-line path:

$$\gamma^{(\mathbf{x}_0, \mathbf{x}_f)}(t) = \left(\mathbf{x}_f - \mathbf{x}_0\right) \frac{t - t_0}{t_f - t_0} + \mathbf{x}_0 \tag{7}$$

The Optimal Motion Plan (OMP) for a robot is the optimal trajectory from its current position to its goal. In the best case, all robots' OMPs from their start positions are collision-free. Then, for each robot, $\gamma_i(t) = \gamma^{(\mathbf{s}_i, \mathbf{g}_i)}(t)$, $t_0 = 0$, and $t_{f,i} = \frac{\|\mathbf{s}_i - \mathbf{g}_i\|_2}{v_{max}}$. Trajectories are *unsynchronized*: all robots travel at v_{max} to arrive at different times.

4.2 Circular HOlding Patterns (CHOPs)

When their OMPs are not collision-free, robots enter a Circular HOlding Pattern (CHOP) to safely maneuver to their goals. Algorithm 1 presents the CHOP construction algorithm and Sects. 4.2.1–4.2.4 detail its key steps. Its inputs are the *CHOP start time*, τ_s, the index set of robots involved, $\mathscr{R}_m$, a set of *CHOP start positions*, X_s, from which robots enter the CHOP, and the set of goals $X_g = \{\mathbf{g}_i \mid i \in \mathscr{R}_m\}$. The equality sign denotes the assignment of a value, a left arrow indicates the addition of discrete elements to an existing set, and $X_{a,i}$ denotes element i of set X_a.

For now, assume all robots immediately enter a single CHOP. This represents the worst-case scenario, where robots are densely packed and smaller CHOPs cannot be created. In this case, the algorithm inputs are $\tau_s = 0$, $\mathscr{R}_m = \mathscr{I}_N$, $X_s = S$, $X_g = G$.

Algorithm 1 $(m, \mathbf{x}_c, r_c) =$ Create CHOP$(\tau_s, \mathscr{R}_m, X_s, X_g, R, v_{max})$

```
$N_m$ = number of robots in $\mathscr{R}_m$
$\mathbf{x}_c = \frac{\sum_{i \in \mathscr{I}_{N_m}} X_{s,i}}{N_m}$ // Define center of the CHOP
$n_w = 2N_m$ // Designate number of intermediate waypoints in the CHOP
$r_c$ = Find CHOP Radius$(n_w, \mathbf{x}_c, X_g, R)$ //Find minimum safe radius for the CHOP
// Find the set of intermediate waypoints
$X_m = \{\mathbf{x}_c + r_c[\cos(\theta_i) \;\; \sin(\theta_i)]^T \mid \theta_i = (i-1)\frac{2\pi}{n_w}, i \in \mathscr{I}_{n_w}\}$
// Assign entry waypoint for each robot
$X_w = \{X_{m,1}, X_{m,3}, ..., X_{m,n_w-1}\}$
$\phi^s$ = CAPT$(X_s, X_w)$
// Define Exit Condition for each robot
for all $i \in \mathscr{I}_{N_m}$ do
    $\phi^g_i = \text{argmin}_{j \in \mathscr{I}_{n_w}} \|X_{m,j} - X_{g,i}\|_2$ // Assign the exit waypoint
end for
$\mathscr{P}_i = \emptyset \;\; \forall i \in \mathscr{I}_{N_m}$ // Find priority sets
for all $i \in \mathscr{I}_{N_m}$ do
    for all $j \in \mathscr{I}_{N_m} \backslash i$ do
        if $\mathscr{B}(X_{g,i}) \cap \mathscr{B}(\gamma^{(X_{m,\phi^g_j}, X_{g,j})}) \neq \emptyset$ then
            $\mathscr{P}_i \leftarrow j$
        end if
    end for
end for
$(m)$ = Construct CHOP$(\tau_s, \mathscr{R}_m, X_s, X_m, X_g, \phi^s, \phi^g, \mathscr{P}, v_{max})$
```

4.2.1 Define Intermediate Waypoints

First, we find a set of n_w *intermediate waypoints*, X_m, distributed evenly about a circle with center $\mathbf{x}_c$ and radius r_c. These waypoints must satisfy *safety conditions*:

1. The distance between all points in the set X_w, defined in Line 8, is at least $2\sqrt{2}R$.
2. The distance of every goal in X_g from every waypoint in X_m is at least $2\sqrt{2}R$.
3. The distance of every goal in X_g from every path between a pair of consecutive intermediate waypoints in X_m is at least $2R$.

We designate n_w as twice the number of robots in $\mathscr{R}_m$ and $\mathbf{x}_c$ as the mean of the robots' start positions. r_c, the minimum radius that satisfies the safety criteria, can be found analytically. Note that robots' goals can be inside or outside the CHOP.

4.2.2 Define Entry Waypoints

To enter a CHOP, robots move synchronously from their CHOP start positions to an intermediate waypoint designated as their *entry waypoint*. Line 8 chooses every other waypoint from X_m to form the set of candidate entry waypoints, X_w. In Line 9, these waypoints are assigned to robots with the optimal assignment returned by the CAPT algorithm when considering X_s as start positions and X_w as goals.

4.2.3 Define Exit Conditions

Next, robots synchronously and sequentially visit intermediate waypoints in clockwise order until they satisfy their Exit Condition (EC). First, Lines 11–13 assigns the intermediate waypoint closest to each robot's goal as its *exit waypoint*. Robots can only exit the CHOP from this waypoint. Second, Lines 14–21 construct each robot's *priority set*, $\mathscr{P}_i$. A robot can exit via its exit waypoint only if all robots in $\mathscr{P}_i$ have exited. Line 17 ensures that if robot i remaining stationary at its goal will result in a collision with robot j moving towards its goal, robot i cannot exit before robot j.

4.2.4 Construct CHOP

To execute a CHOP, each robot follows optimal trajectories to sequentially visit its CHOP start position, its entry waypoint, a series of intermediate waypoints, and its exit waypoint at the appropriate times. Upon satisfying its EC, it returns to pursuing an OMP starting from its exit waypoint. Thus, we can fully represent the motion of all robots in a CHOP with $m = \{\{X_i \mid i \in \mathscr{R}_m\}, T, T_{goal}\}$. X_i is the series of waypoints robot i visits, starting from its CHOP start position and ending with its exit waypoint. Note that the sets X_i can be different lengths. $T = \{t_1, t_2, \ldots\}$ indicates arrival times at waypoints, where robot i must be at position $X_{i,j}$, if it exists, at time t_j. T is common to all robots, and $|T| = \max_{i \in \mathscr{R}_m} |X_i|$, where $|\cdot|$ denotes a set's cardinality. Finally, $T_{goal} = \{t_{goal,i} \mid i \in \mathscr{R}_m\}$ is a series of *goal arrival times*. Robot i must reach its goal at time $t_{goal,i}$ after exiting the CHOP.

We already know X_i for each robot. Line 22 additionally defines the series T and T_{goal}. Section 4.4 will show that to guarantee safety, trajectories between waypoints in the CHOP and the OMPs of robots that have exited must all be synchronized. To achieve this while respecting all robots' velocity constraints, we define $t_1 = \tau_s$ and:

$$t_j = t_{j-1} + \max_{i \in \mathscr{R}_{m_j}} \frac{\|\mathbf{x}_{next,i} - X_{i,j-1}\|_2}{v_{max}} \qquad j = 2, \ldots, j_{max} \tag{8}$$

Here, $j_{max} = \max_{i \in \mathscr{R}_m} |X_i|$. $\mathscr{R}_{m_j} \subseteq \mathscr{R}_m$ is the subset of indices for which $|X_i| \geq j - 1$, $\mathbf{x}_{next,i}$ refers to $X_{i,j}$ if $|X_i| \geq j$ and $X_{g,i}$ if $|X_i| = j - 1$. Then:

$$t_{goal,i} = \begin{cases} t_{|X_i|+1} & \text{if } |X_i| < j_{max} \\ t_{j_{max}} + \max_{i \in \mathscr{R}_{m_{j_{max}}}} \frac{\|G_i - X_{i,j_{max}}\|_2}{v_{max}} & \text{if } |X_i| = j_{max} \end{cases} \tag{9}$$

We further define the *CHOP exit time* for each robot, denoted $\tau_{f,i}$, as the time it leaves its exit waypoint, which we will use in later algorithms.

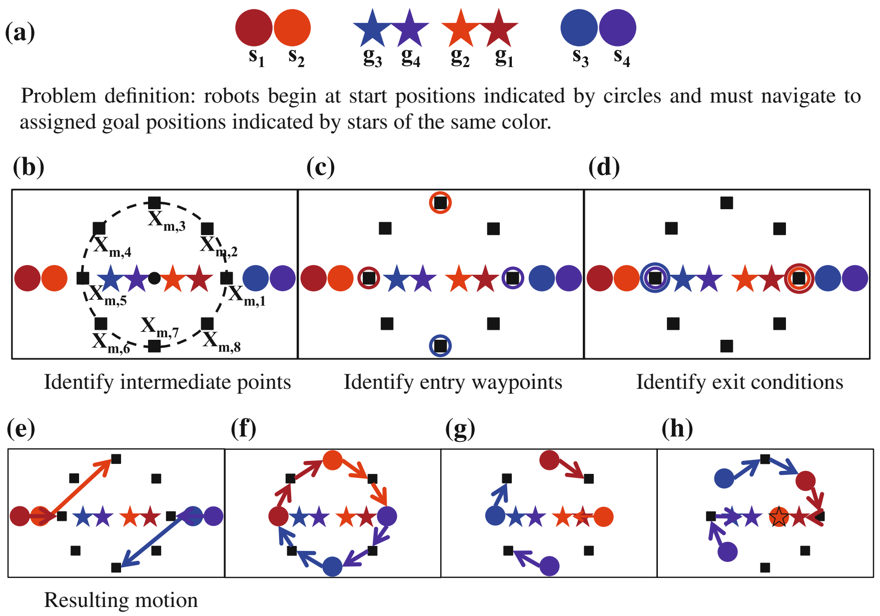
4.2.5 Example Problem

We illustrate Algorithm 1 using the example problem presented in Fig. 1a, b illustrates the placement of the intermediate waypoints, pictured as black squares. Figure 1c shows the assigned entry waypoint for each start position with a circle of the same color. Figure 1d shows each robot's assigned exit waypoint with a circle of the same color, with higher priority robots indicated by larger circles. Figure 1e–h illustrate the resulting motion plan. As an example, robot 2's planned trajectory is:

$$\gamma_2(t) = \begin{cases} \gamma^{(\mathbf{s}_2, X_{m,3})}(t) & t_0 \leq t \leq t_1 \\ \gamma^{(X_{m,3}, X_{m,2})}(t) & t_1 < t \leq t_2 \\ \gamma^{(X_{m,2}, X_{m,1})}(t) & t_2 < t \leq t_3 \\ \gamma^{(X_{m,1}, \mathbf{g}_2)}(t) & t_3 < t \leq t_{exit,2} \end{cases} \tag{10}$$

4.3 The Motion Planning Algorithm

Now, we combine the previous techniques into a single motion planning algorithm, referred to as OMP+CHOP, that allows robots to follow their OMPs when possible and directs them into appropriately designed CHOPs in congested areas. This algorithm is presented in Algorithm 2 and described in detail in Sects. 4.3.1–4.3.3.



Problem definition: robots begin at start positions indicated by circles and must navigate to assigned goal positions indicated by stars of the same color.

Identify intermediate points

Identify entry waypoints

Identify exit conditions

Resulting motion

Fig. 1 We design a Circular HOlding Pattern (CHOP) for the problem in (**a**). **b**–**d** show key steps of Algorithm 1. **e**–**h** illustrate the motion plan

Algorithm 2 γ = OMP_CHOP(S, G, R, v_{max})

```
1: $\mathscr{M} = \emptyset$
2: $\gamma$ = Compute Motion Plan ($S$, $G$, $\mathscr{M}$)
3: $\mathscr{C}$ = Find Imminent Collision($\gamma$, $R$)
4: **while** $\mathscr{C} \neq \emptyset$ **do**
5:     $(\tau_s, \mathscr{R}_{add}, X_s, X_g, \mathscr{M}_{add})$ = Compute CHOP Parameters($\gamma$, $\mathscr{M}$, $\mathscr{C}$, $G$)
6:     $(m_{new}, \mathbf{x}_c, r_c)$ = Create CHOP($\tau_s$, $\mathscr{R}_{add}$, $X_s$, $X_g$, $R$, $v_{max}$)
7:     $\mathscr{M} \leftarrow m_{new}$
8:     $\mathscr{M}$ = Remove CHOPs($\mathscr{M}_{add}$)
9:     $\gamma$ = Compute Motion Plan ($S$, $G$, $\mathscr{M}$)
10:    $\mathscr{C}$ = Find Imminent Collision($\gamma$, $R$)
11: **end while**
```

4.3.1 Compute Motion Plan

$\mathscr{M}$ contains the set of all CHOPs in the motion plan, from which the set of trajectories γ can be derived. Initially, in Line 2, $\mathscr{M}$ is empty and all robots follow their OMPs from their start positions. When $\mathscr{M}$ contains CHOPs, as in Line 9, each robot follows its OMP until its earliest CHOP's start time. It then follows optimal trajectories to each waypoint designated by the CHOP to its exit waypoint, when it again pursues an OMP until it encounters another CHOP or its goal. This process is pictured in Fig. 2. The choice of CHOP parameters, described in Sect. 4.3.3, will guarantee that CHOPs in $\mathscr{M}$ will always start along its robots' OMPs.

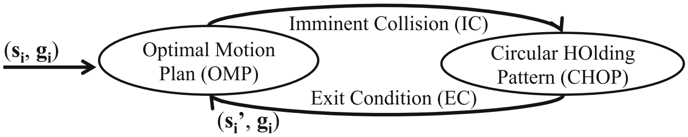


Fig. 2 Overview of our algorithm's motion plan: robots follow their OMP whenever possible, entering CHOPs to resolve collisions

We will use a subscripted variable, such as m_k, to denote a particular CHOP in $\mathscr{M}$ and $\mathscr{R}_{m_k}$, τ_{s,m_k}, τ_{f,i,m_k} to denote the indices of its robots, its start time, and the CHOP exit time of robot $i \in \mathscr{R}_{m_k}$, respectively.

4.3.2 Find Imminent Collisions (ICs)

Line 3 finds the first Imminent Collision (IC) amongst robots following trajectories γ. We characterize a collision with its time, t_c, the number of robots in collision, N_c, and the set of indices of the colliding robots, $\mathscr{C}$. For all robots $i \in \mathscr{C}$, there must be at least one $j \neq i \in \mathscr{C}$ for which $\mathscr{B}(\mathbf{x}_i(t_c)) \cap \mathscr{B}(\mathbf{x}_j(t_c)) \neq \emptyset$. We will use $\mathscr{C}$ to denote both the collision and the set of robots in the collision.

4.3.3 Create Local CHOP

Line 5 of Algorithm 2 finds the parameters of a new CHOP, denoted m_{new}, that will resolve the detected IC. This function is presented in Algorithm 3.

m_{new} is characterized by the set of indices of the robots it contains, R_{add}. As shown in Line 1 of Algorithm 3, $\mathscr{R}_{add}$ initially contains only robots in the IC. Additionally, existing CHOPs in $\mathscr{M}$ might need to be *merged* with m_{new}. These CHOPs are contained in $\mathscr{M}_{add}$, which is initially empty. $\mathscr{M}_{curr}$, also initially empty, contains only the CHOPs to be merged that were identified in the most recent iteration.

Algorithm 3 then grows $\mathscr{R}_{add}$ and $\mathscr{M}_{add}$ until a valid CHOP can be constructed. Line 2 indicates that if any robots in $\mathscr{C}$ are executing a CHOP when the IC occurs, their CHOPs must be merged with m_{new}. Lines 5–6 defines the CHOP start time and start positions for the current $\mathscr{R}_{add}$, ensuring that the start positions are always on robots' OMPs. Line 8 creates m_{curr}, the CHOP defined by the current $\mathscr{R}_{add}$. Additional robots and CHOPs are added based on three *merging conditions*:

1. Add robots and CHOPs whose paths intersect m_{curr}'s circle (Lines 10–13), so when moving between intermediate waypoints, robots in m_{curr} will be collision-free, even with robots not in the CHOP. Note we only consider robots' paths, which simplifies this condition to fast line segment-circle intersection tests.
2. Merge CHOPs that will cause conflicting motion plans for robots in $\mathscr{R}_{add}$ (Lines 15–19), so $\mathscr{M}$ will always translate to a valid motion plan.

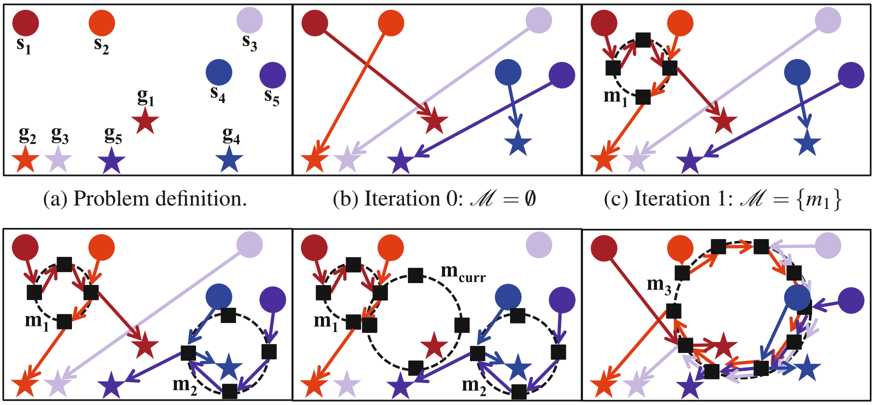
Algorithm 3 $(\tau_s, \mathscr{R}_{add}, X_s, X_g, \mathscr{M}_{add})$ = Compute CHOP Parameters$(\gamma, \mathscr{M}, \mathscr{C}, G)$

```
1: $\mathscr{M}_{add} = \emptyset, \mathscr{R}_{add} = \mathscr{C}, t_s = t_c, merge = 1$ // Initialize variables
2: $\mathscr{M}_{curr} = \{m_k \in \mathscr{M} \mid \exists\, r \in \mathscr{C} \cap \mathscr{R}_{m_k}$ for which $t_c \in [\tau_{s,m_k}, \tau_{f,r,m_k}]\}$
3: while $true$ do
4:   // Find valid starting conditions
5:   $\tau_s = \max_{t \le t_s} t$ such that $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|_2 \ge 2\sqrt{2}R \;\; \forall j \ne i \in \mathscr{R}_{add}$
6:   $X_s = \{\mathbf{x}_i(\tau_s) \mid i \in \mathscr{R}_{add}\}, X_g = \{\mathbf{g}_i \mid i \in \mathscr{R}_{add}\}$
7:   if $merge == 0$, $break$ end if
8:   $(m_{curr}, \mathbf{x}_c, r_c)$ = Create CHOP$(\tau_s, \mathscr{R}_{add}, X_s, X_g, R, v_{max})$
9:   // Merge robots and CHOPs whose paths intersect $m_{curr}$'s circle
10:  $t_a = t_{1,m_{curr}}, t_b = \max_{i \in \mathscr{R}_{add}} \tau_{f,i,m_{curr}}$
11:  $l$ = Set of paths that all robots $r \in \mathscr{I}_N \setminus \mathscr{R}_{add}$ traverse between $[t_a, t_b]$
12:  $\mathscr{R}_{OMP}$ = Robots whose OMP's paths are in $l$ and intersect a circle at $\mathbf{x}_c$, radius $r_c + 2R$
13:  $\mathscr{M}_{curr} \leftarrow$ CHOPs whose paths are in $l$ and intersect a circle at $\mathbf{x}_c$, radius $r_c + 2R$
14:  // Merge CHOPs that will cause conflicting motion plans for robots in $\mathscr{R}_{add}$
15:  $\mathscr{R}_{add} \leftarrow \mathscr{R}_{OMP} \cup \{\mathscr{R}_{m_j} \mid m_j \in \mathscr{M}_{curr}\}$
16:  for $r \in \mathscr{R}_{add}$ do
17:    $\tau_{min,r} = \min(\tau_s \cup \{\tau_{s,m_j} \mid m_j \in \mathscr{M}_{curr}$ and $r \in \mathscr{R}_{m_j}\})$
18:  end for
19:  $\mathscr{M}_{curr} \leftarrow \{m_k \in \mathscr{M} \mid \exists\, r \in \mathscr{R}_{m_k} \cap \mathscr{R}_{add}$ and $\tau_{f,r,m_k} \ge \tau_{min,r}\}$
20:  // Merge CHOPs that contain two or more common robots with $R_{add}$
21:  $\mathscr{R}_{add} \leftarrow \{\mathscr{R}_{m_j} \mid m_j \in \mathscr{M}_{curr}\}$
22:  $\mathscr{M}_{curr} \leftarrow \{m_k \in \mathscr{M} \mid |\mathscr{R}_{add} \cap \mathscr{R}_{m_k}| \ge 2\}$
23:  // If any additional robots or CHOPs were identified to be merged, iterate again
24:  if $\mathscr{R}_{OMP} \ne \emptyset$ or $\mathscr{M}_{curr} \setminus \mathscr{M}_{add} \ne \emptyset$ then
25:    $\mathscr{M}_{add} \leftarrow \mathscr{M}_{curr}, \mathscr{R}_{add} \leftarrow \{\mathscr{R}_{m_j} \mid m_j \in \mathscr{M}_{curr}\}, t_s = \min(\tau_s \cup \{\tau_{s,m_j} \mid m_j \in \mathscr{M}_{curr}\})$
26:    $merge = 1, \mathscr{M}_{curr} = \emptyset$
27:  else
28:    $merge = 0$
29:  end if
30: end while
```

3. Merge CHOPs that contain two or more common robots with R_{add} (Lines 21–22). This ensures that no two robots will be in the same CHOP more than once, which will help provide algorithm completeness in Sect. 4.5.

Line 21 adds any new robots to $\mathscr{R}_{add}$ and Line 25 merges any new CHOPs. To merge the CHOPs in $\mathscr{M}_{add}$, their constituent robots are added to $\mathscr{R}_{add}$. If any merged CHOPs occur before m_{curr}, m_{curr}'s start time is shifted to the earliest start time. We then reconstruct m_{curr} with the updated $\mathscr{R}_{add}$ and iterate again as necessary.

With the returned parameters, we use Algorithm 1 to create the new CHOP, m_{new}, which is added to $\mathscr{M}$. The merged CHOPs in $\mathscr{M}_{add}$ are removed. A new motion plan is computed and the next IC is resolved until the motion plan is collision-free.



(a) Problem definition. (b) Iteration 0: $\mathscr{M} = \emptyset$ (c) Iteration 1: $\mathscr{M} = \{m_1\}$

(d) Iteration 2: $\mathscr{M} = \{m_1, m_2\}$ (e) Iteration 3: m_{curr} created, and additional merging is needed. (f) Iteration 3: $\mathscr{M} = \{m_3\}$

Fig. 3 Illustration of Algorithm 2 on the example problem in (**a**). Robots start at *circles* and must navigate to stars of the same color

4.3.4 Example Problem

Figure 3 illustrates Algorithms 2 and 3 on the example problem in Fig. 3a.

Figure 3b shows the initial motion plan, where $\mathscr{M} = \emptyset$ and all robots follow their OMPs from their start positions. Figure 3c, d shows the motion plans after the first two ICs are resolved. Next, an IC between robots 1 and 3 is detected.

m_{curr} in Fig. 3e represents the initial CHOP created in Line 1 of Algorithm 3, where $\mathscr{R}_{add} = \mathscr{C} = \{1, 3\}$. In Lines 12–13, robot 5's OMP's path and robot 2's path in m_1 are found to intersect m_{curr}'s circle. Thus, $\mathscr{R}_{OMP} = \{5\}$, $\mathscr{M}_{curr} = \{m_1\}$. At Line 15, $\mathscr{R}_{add} = \{1, 2, 3, 5\}$. Evaluating Line 19, m_2 contains robot 5, which is in $\mathscr{R}_{add}$, and $\tau_{f,m_2,5} > \tau_{min,5} = \tau_{s,m_{curr}}$. Thus, m_2 is added to $\mathscr{M}_{curr}$. Lines 21–22 will not change $\mathscr{R}_{add}$ or $\mathscr{M}_{curr}$. Finally, from Line 25, $t_s = \tau_{m_1}$, $\mathscr{R}_{add} = \{1, 2, 3, 4, 5\}$, and $\mathscr{M}_{add} = \{m_1, m_2\}$. No further additions to $\mathscr{R}_{add}$ or $\mathscr{M}_{add}$ are needed.

We create $m_{new} = m_3$ and add it to $\mathscr{M}$, and m_1 and m_2 are removed from $\mathscr{M}$. No other ICs exist. Figure 3f illustrates the resulting motion plan.

Note Algorithm 2 can be modified to accommodate vehicles with a lower velocity bound, v_{min}, instead of v_{max}. With an additional constraint that a CHOP's intermediate waypoints must be at least $2\sqrt{2}R$ away from its start positions, the minimum length of any synchronized trajectory is $d_{min} = 2\sqrt{2}R$. The maximum length is $d_{max} = \sqrt{2}r_{c,max}$, where $r_{c,max}$ is the radius of a CHOP involving all N robots and contains all goals in G. Thus, running Algorithm 2 with $v_{max} = v_{min}\frac{d_{max}}{d_{min}}$ will ensure that robots will not travel slower than v_{min}.

4.4 Safety

Theorem 1 *Robots are collision-free when executing a CHOP from Algorithm 1.*

Proof Consider a CHOP $m = \{\{X_i \mid i \in \mathscr{R}_m\}, T, T_{goal}\}$ with final goals X_g. Let $X_s^k = \{X_{i,k-1} \mid i \in \mathscr{R}_{m_k}\}$ denote the positions of robots in $\mathscr{R}_{m_k}$ at t_{k-1} and X_g^k denote the set $\{\mathbf{x}_{next,i} \mid i \in \mathscr{R}_{m_k}\}$. Here, $\mathscr{R}_{m_k}$ and $\mathbf{x}_{next,i}$ are defined as in Eq. 8. We show that robots' trajectories are collision-free for all $k = 2, \ldots, \max_{i \in \mathscr{R}_m} |X_i| + 1$.

We use the CAPT algorithm to assign entry waypoints, so for $k = 2$, when robots move from their CHOP start positions to their entry waypoints, the assignment of goals X_g^k to robots at X_s^k minimizes the total distance squared.

In subsequent intervals, X_s^k contains only intermediate waypoints while X_g^k can contain both intermediate waypoints and goals. Suppose robot $i \in \mathscr{R}_{m_k}$ is moving between intermediate waypoints. Robots enter at every other intermediate waypoint and subsequent rotations are synchronized, so $X_{i,j} \neq X_{j,k-1} \forall j \neq i \in \mathscr{R}_{m_k}$. Thus:

$$\|X_{i,k} - X_{i,k-1}\|_2^2 \leq \|X_{i,k} - X_{j,k-1}\|_2^2 \quad \forall j \neq i \in \mathscr{R}_{m_k} \tag{11}$$

Now, suppose robot i is moving from its exit waypoint to its goal. By design, the exit waypoint is the closest intermediate waypoint to the goal. Thus:

$$\|X_{g,i} - X_{i,k-1}\|_2^2 \leq \|X_{g,i} - X_{j,k-1}\|_2^2 \quad \forall j \neq i \in \mathscr{R}_{m_k} \tag{12}$$

As a result, no alternate assignment of points in X_s^k to those in X_g^k will result in paths with a lower total distance squared than the CHOP's specified assignment. Thus, in each time interval, robots move from their positions in X_s^k to the one in X_k^g that coincides with the minimum total distance squared assignment.

Line 5 of Algorithm 3 and safety conditions 1 and 2 of Algorithm 1 guarantee positions in X_s^k and X_g^k for all k meet the separation conditions in Eq. 2. The CAPT algorithm guarantees all synchronized trajectories between waypoints are collision-free [14]. Finally, safety condition 3 and the priority sets in Algorithm 1 ensure robots stationary at their goals will not collide with moving robots.

By assigning inter-waypoint motions that match the optimal unlabeled allocation, we inherit the collision avoidance guarantees of the CAPT algorithm. In essence, we use a series of solutions to the unlabeled problem to move towards labeled goals.

4.5 Completeness

Theorem 2 *Algorithm 2 is complete.*

Proof To be complete, an algorithm must always find a collision-free motion plan in finite time if one exists and indicate that there is no solution when one does not.

From Theorem 1, a CHOP containing all N robots will always be a valid solution. We must additionally show that Algorithm 2 returns a solution in finite iterations.

First note that Algorithm 3 always returns in finite iterations, as there are finite numbers of robots and CHOPs that can be added to $\mathscr{R}_{add}$ and $\mathscr{M}_{add}$, and elements are never removed. Define $\mathscr{A}$ as the set of *interactions* in $\mathscr{M}$. An interaction is a pair of indices of robots, $\{i, j\}$, such that $i, j \in \mathscr{R}_m$ for some $m \in \mathscr{M}$. For example, in Fig. 3d, $\mathscr{A} = \{\{1, 2\}, \{4, 5\}\}$. When all robots are in a single CHOP, $\mathscr{A} = [\mathscr{I}_N]^2$.

In each iteration of Algorithm 2, either the algorithm terminates, or a new CHOP is added to $\mathscr{M}$. In the latter case, the set of interactions in $\mathscr{A}$ is strictly growing.

To see this, first note that at each iteration, all removed CHOPs have been merged into m_{new}, so interactions are never removed. Alternatively, $\mathscr{A}$ can remain unchanged. This can only occur if $\mathscr{M}_{add}$ contains a single CHOP, m_1, identical to m_{new}. Suppose m_{new} resolves the IC, $\mathscr{C}$. Then, $\mathscr{C} \subseteq \mathscr{R}_{m_{new}} = \mathscr{R}_{m_1}$. m_1 resolves the first IC between robots in $\mathscr{C}$ and guarantees they reach their goals collision-free. Thus, robots in $\mathscr{C}$ can only collide if they abandon their OMPs to enter other CHOPs. Let $\mathscr{M}_{after}$ be the set of CHOPs that robots in $\mathscr{C}$ enter after exiting m_1. CHOPs in $\mathscr{M}_{after}$ fulfill merging condition 2, so $\mathscr{M}_{after} \subset \mathscr{M}_{add}$, and $\mathscr{M}_{add} \neq \{m_1\}$. We have a contradiction, so $\mathscr{A}$ must contain at least one new interaction.

Merging condition 3 guarantees that robots will interact at most once. In finite iterations, $\mathscr{A}$ will contain all unique interactions. This represents the case where all robots are in a single CHOP, which is a collision-free motion plan.

5 Simulation Results

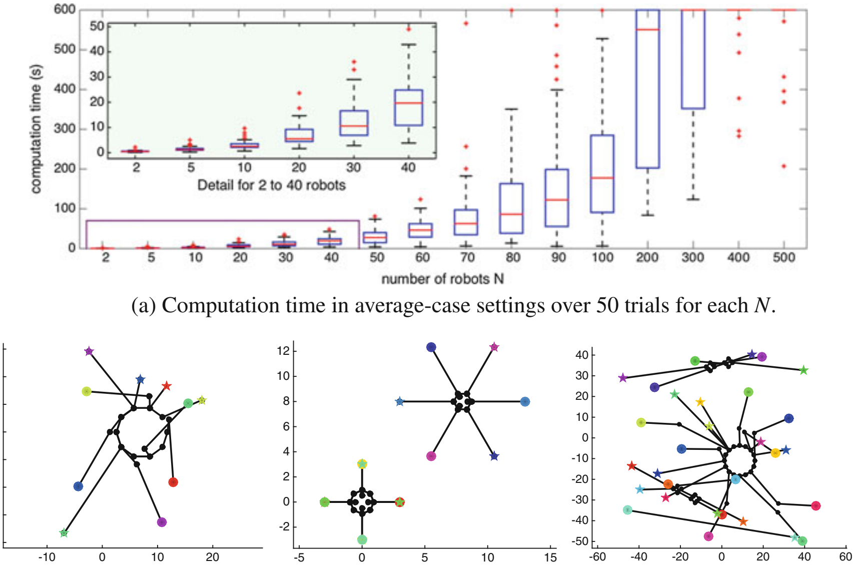
Finally, we examine the algorithm's performance in simulations. Experiments were done on a 2.5 GHz Macbook Pro in MATLAB and C++ Mex, with a maximum algorithm runtime of 10 min.

We define a solution's sub-optimality ratio using the total distance of its paths:

$$r_d = \frac{\sum_{i=0}^{N} \int_0^{t_{f,i}} \dot{\gamma}_i(t)dt}{\sum_{i=0}^{N} \|\mathbf{s}_i - \mathbf{g}_i\|_2} \tag{13}$$

The denominator is an underestimate of the optimal total distance, as for problems like Fig. 1a, the straight-line paths to goals have no collision-free velocity profile.

To detect ICs, we sample trajectories at $dt = \frac{R}{v_{max}}$, where $R = 1, v_{max} = 5$, to ensure no collisions occur between samples. We check for collisions using a spatial hashing algorithm [8] and further eliminate checks for robots moving between intermediate waypoints and between pairs of robots executing the same CHOP.



(a) Computation time in average-case settings over 50 trials for each N.

(b) 5 robots with 1 CHOP

(c) 7 robots with 2 CHOPs

(d) 15 robots with 5 CHOPs

Fig. 4 **a** Displays our algorithm's performance over 50 randomly generated case studies. **b**–**d** illustrates the final motion plans for example problems

5.1 Variations in Team Size

To examine the effect of the team's size on computation time, we randomly generate case students for 500 robots. We then subsample 400 start to goal assignments from the original set, 300 assignments from the remaining set of 400, and so on.

Figure 4a plots the algorithm computation time for various team sizes. All motion plans for $N \leq 100$ were found in less than 4 min. Figure 7 plots the suboptimality ratios of the solutions, r_d, which is below 7 for all solved problems. Figure 4b–d shows the paths of the final motion plan for three example problems.

5.2 Variations in Problem Density

Next, for a given team size N, we deterministically generate a set of start positions from Halton sequences. These positions are sorted by y-coordinate and stored in S_{init}. For each experiment, we choose a constant D_k and construct the sets $S = D_k S_{init}$ and $G = S + [2R \quad 0]^T$. Robot $i \in \mathscr{I}_N$ is assigned start position $\mathbf{s}_i = S_i$ and goal $\mathbf{g}_i = G_{\phi_i^{D_k}}$. $\phi_i^{D_k} = i$ for $i \leq \left\lceil \frac{D_k}{D_{k,max}} N \right\rceil$, and $\phi_i^{D_k}$ for other robots are a random permutation of each other's indices. We designate $D_{k,max} = 50$. When $D_k = D_{k,max}$, $\phi_i^{D_k} = i$ for

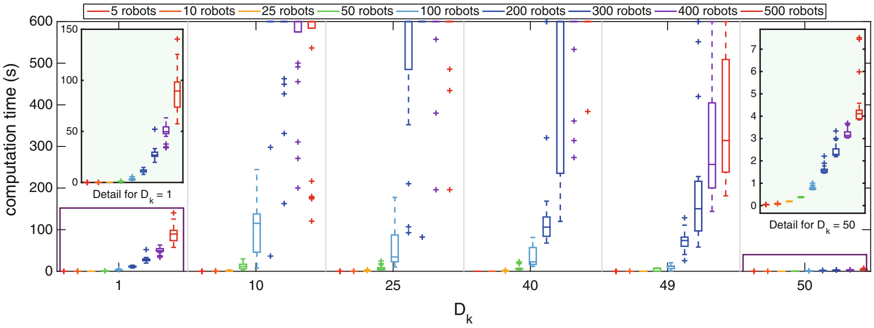


Fig. 5 Computation time over 25 trials for each combination of N and D_k

all robots, representing the best-case scenario: robots are sparsely located and their OMPs, a simple translation rightwards, are safe. As D_k decreases, the available free space decreases and the number of collisions increases.

Figure 5 shows the computation time and Fig. 7 shows the corresponding r_d values over 25 trials for each combination of N and D_k. For small D_k, robots are tightly packed and a single large CHOP will likely be created in a few iterations. Solutions are found quickly, but r_d values are high. As D_k increases, the available free space allows for formation of more local CHOPs, causing smaller deviations from robots' OMPs. This decreases r_d, but increases the computation time. This increase in computation time is more dramatic for larger values of N.

For large D_k, collisions become sparse and fewer CHOPs need to be constructed, decreasing both the computation time and r_d. When $D_k = D_{max}$, no CHOPs need to be created, so the computation time required is small. In short, our algorithm finds solutions quickly for both extremely sparse and dense settings, but requires more computation time when planning many local CHOPs for large teams.

5.3 *Worst-Case Distributions*

We also evaluate the algorithm's performance in the worst-case scenario. For a given N, we find the densest packing of N equally-sized circles in a square that satisfies the separation conditions [12]. We use these circles' centers as both the start and goal positions and generate 50 random assignments for each N. These problems pose the additional challenge that each robot's goal is the start position of another robot.

Figure 6 shows we can efficiently solve these problems for $N \leq 504$ in less than 3.5 min. Again, once the first collision is found, it is probable that a CHOP containing all N robots will be formed in a only a few iterations. As shown in Fig. 7, r_d becomes rather high for large teams. Nonetheless, we are able to find safe motion plans for teams of hundreds of robots in a challenging environment.

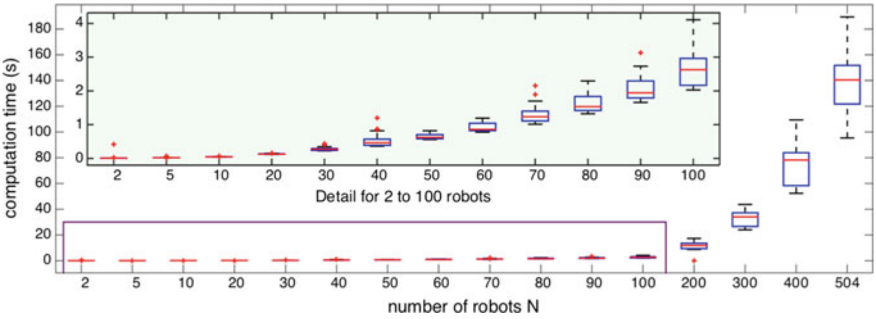


Fig. 6 Computation time in worst-case settings over 50 trials for each N

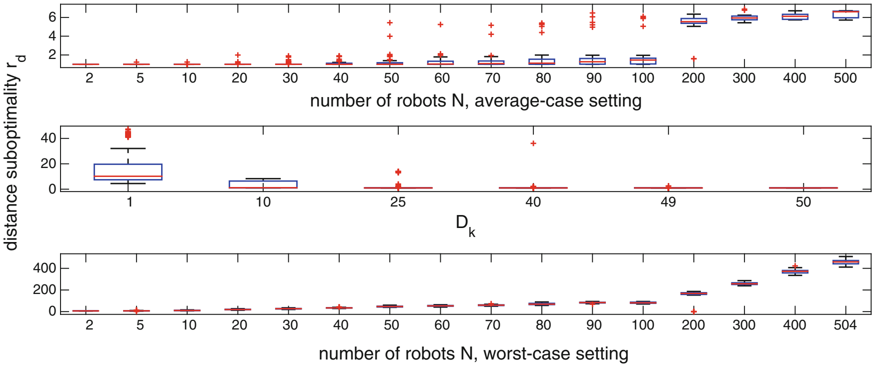


Fig. 7 Suboptimality over all experiments

5.4 Comparison with Other Multi-robot Planning Algorithms

Finally, we discuss the performance of our algorithm in comparison with M* with heuristic function inflated by ε [18] and Optimal Reciprocal Collision Avoidance (ORCA) [16]. Table 1 reports the algorithms' performances for a problem generated in Sect. 5.2 with $N = 10$, $D_k = 1$, 5, and 10.

The M* algorithm builds on A* as an underlying algorithm, searching for optimal paths to goals for each robot in its individual configuration space when robots are collision-free and in a higher-dimensional joint configuration space when they collide. M* retains the completeness and optimality guarantees of A*. In the best-case scenario, M* is extremely fast, as its search space remains low-dimensional. However, its computation time scales up quickly as robots become more densely packed, as the size of the search space grows exponentially with each additional robot in collision. The computation time of our OMP+CHOP algorithm does not scale up as quickly. We note that variants of M* can improve performance, but no results for M*-based algorithms have been reported for problems where $N > 200$ [18].

Table 1 Comparison of performances of multi-robot planning algorithms

		OMP+CHOP	M* ($\varepsilon = 1.5$)	ORCA [15]
Best case	Planning time (s)	2.78	0.0020	6.00
$D_k = 10$	Suboptimality ratio	1.00	1.00	1.00
Average case	Planning time (s)	2.59	0.027	70.25
$D_k = 5$	Suboptimality ratio	1.07	1.001	1.001
Worst case	Planning time (s)	2.65	16.09	23.00
$D_k = 1$	Suboptimality ratio	5.35	1.11	1.07

ORCA is a decentralized, real-time algorithm that, at each time step, assigns each robot a safe velocity based on the observed velocities of its neighbors. The assigned velocity is guaranteed to be collision-free for a known time horizon. We report the total time of the motion plan as the algorithm's planning time, but note that these are different measures. Compared to OMP+CHOP, ORCA's solutions are more optimal. However, in highly dense scenarios, it is possible that a guaranteed safe velocity cannot be found and robots are forced to choose a "best possible" velocity instead. While ORCA has been shown to perform well for large teams in dense settings in practice [16], there are no safety or completeness guarantees.

6 Conclusions and Future Work

We present the OMP+CHOP algorithm to solve the labeled multi-robot planning problem. This algorithm is scalable while still maintaining safety and completeness guarantees. CHOPs are designed analytically, and no high-dimensional graph searches are required to resolve imminent collisions between robots. This becomes particularly beneficial in densely packed regions or when many robots converge at a single collision point, where other motion planning algorithms reach bottlenecks.

However, we trade off optimality for safety and scalability. In particular, in densely packed problems, the motion plan can be very suboptimal and some robots might circle the CHOP many times before exiting. Immediate directions for future research are applying the algorithm to robots with higher order dynamics and developing a decentralized algorithm requiring only local communication. Future work will also work towards analytically characterizing the algorithm's suboptimality.

Acknowledgements We gratefully acknowledge the support of ONR grants N00014-09-1-1051 and N00014-09-1-103, NSF grant IIS-1426840, and Exyn Technologies. Sarah Tang is supported by NSF Research Fellowship Grant No. DGE-1321851. The authors would also like to thank Levi Cai from the University of Pennsylvania for his implementation of the M* algorithm.

References

1. Buckley, S.: Fast motion planning for multiple moving robots. In: Proceedings of the 1989 IEEE International Conference on Robotics and Automation (ICRA), pp. 322–326 (1989)
2. de Wilde, B., ter Mors, A.W., Witteveen, C.: Push and rotate: cooperative multi-agent path planning. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS), pp. 87–94 (2013)
3. Erdmann, M., Lozano-Perez, T.: On multiple moving objects. Algorithmica **2**, 1419–1424 (1986)
4. FAA: Overview of small uas notice of proposed rulemaking (2015)
5. Forbes: Meet amazon prime air, a delivery-by-aerial-drone project (2013)
6. Goldenberg, M., Felner, A., Stern, R., Sharon, G., Sturtevant, N., Holte, R.C., Schaeffer, J.: Enhanced partial expansion A*. J. Artif. Intell. Res. **50**(1), 141–187 (2014)
7. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
8. Hastings, E.J., Mesit, J., Guha, R.K.: Optimization of large-scale, real-time simulations by spatial hashing. In: Proceedings of the 2005 Summer Computer Simulation Conference, pp. 9–17 (2005)
9. Kant, K., Zucker, S.W.: Toward efficient trajectory planning: the path-velocity decomposition. Int. J. Robot. Res. (IJRR) **5**(3), 72–89 (1986)
10. Kuhn, H.: The hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (1955)
11. Peng, J., Akella, S.: Coordinating multiple robots with kinodynamic constraints along specified paths. Int. J. Robot. Res. (IJRR) **24**(4), 295–310 (2005)
12. Specht, E.: The best known packings of equal circles in a square (2013). [Online]. Available: http://hydra.nat.uni-magdeburg.de/packing/csq/csq.html
13. Tomlin, C., Pappas, G.J., Sastry, S.: Conflict resolution for air traffic management: a study in multi-agent hybrid systems. IEEE Trans. Autom. Control **43**, 509–521 (1998)
14. Turpin, M., Michael, N., Kumar, V.: CAPT: concurrent assignment and planning of trajectories for multiple robots. Int. J. Robot. Res. **33**(1), 98–112 (2014)
15. van den Berg, J.: RVO2 library documentation (2008). [Online]. Available: http://gamma.cs.unc.edu/RVO2/documentation/2.0/index.html
16. van den Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal *n*-body collision avoidance. In: The 14th International Symposium on Robotics Research (ISRR), pp. 3–19 (2009)
17. van den Berg, J., Snoeyink, J., Lin, M., Manocha, D.: Centralized path planning for multiple robots: optimal decoupling into sequential plans. In: Proceedings of Robotics: Science and Systems (RSS) (2009)
18. Wagner, G., Choset, H.: Subdimensional expansion for multirobot path planning. Artif. Intell. **219**, 1–24 (2015)
19. Yu, J., LaValle, S.M.: Planning optimal paths for multiple robots on graphs. In: Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 3612–3617 (2013)

Neuromorphic Artificial Sense of Touch: Bridging Robotics and Neuroscience

Udaya Bhaskar Rongala, Alberto Mazzoni, Domenico Camboni, Maria Chiara Carrozza and Calogero Maria Oddo

1 The Challenge of Understanding and Emulating the Somatosensory System

The scientific knowledge on human senses such as audition and vision is well grounded as recognized by the Nobel prizes awarded in 1961 for the explanatory findings about the physiology of the cochlea, and in 1911, 1967 and 1981 for the description of optical refraction, transduction mechanisms in the eye and perceptual processes in the vision cortex, respectively.[1] Conversely, a well-integrated theory of human tactile sensing is still lacking, i.e., the physical determinants of tactile coding and perception are not yet fully understood, and the detailed neuronal mechanisms and relative contributions of the different classes of mechanoreceptors remain to be identified [40]. For instance, agreement has not yet been reached on the most informative mechanoreceptor composition or the coding strategy (e.g., temporal, spatial, spatiotemporal, intensity) used by humans to map the different perceptual dimensions (force, roughness, softness and slipperiness) [17] of natural tactile interaction.

Another open debate in tactile processing concerns the coding of textural features: some recent studies supported Katz's duplex theory proposed in 1925, according to which tactile experience is supposed to be mediated by different classes of mechanoreceptors via the transition from spatial cues for coarse geometries

[1] Nobel Laureates in Physiology or Medicine (www.nobelprize.org/nobel_prizes/medicine/laureates).

Udaya Bhaskar Rongala and Alberto Mazzoni are equally contributed.

U.B. Rongala · A. Mazzoni · D. Camboni · M.C. Carrozza (✉) · C.M. Oddo (✉)
The BioRobotics Institute, Scuola Superiore Sant'Anna (SSSA), I-56025
Pontedera, Pisa, Italy
e-mail: m.c.carrozza@sssup.it

C.M. Oddo
e-mail: c.oddo@sssup.it

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_35

towards vibrational cues for fine textures [4, 12, 13]. On the other hand, Johnson and colleagues presented human psychophysical studies and complementary electrophysiological results with non human primates supporting a peripheral neural mechanism for roughness, based on the spatial variation in the firing rate of slowly adapting afferents [5, 8, 41]. Recently, a spatiotemporal model was proposed to account for integration of tactile information from neighbouring receptors distributed along the fingertip [16, 18], supporting an integrated approach rather than transition from spatial to temporal coding of the duplex theory. Such integrated hypothetical model proposes that: (i) the relative timing of neural spikes elicited in (neighboring) tactile units of the fingertip conveys significant information during manipulation activities; (ii) the spikes pass through neural afferents showing differentiated delays (due to dispersion of conduction velocity) in the pathways up to the second order (Cuneate) neurons; (iii) second order neurons propagate the firing events to the higher stage in case that the differential delay introduced by the afferent pathways compensates the relative spike timing at the level of mechanoreceptors in the fingerpad; (iv) the tactile stimulus is pre-cortically represented through the population patterns of second order neurons being activated during finger-surface mechanical interaction.

A deeper understanding of biological touch sensing would foster the development of artificial tactile systems: adopting a biorobotic approach [10] to systematically merge engineering of artificial touch and neurophysiology of human touch shows great promises and mutual benefits for both robotics and neuroscience [35]. Neuromorphic biorobotics thus offers the possibility to emulate a biological sense of touch, with variable characteristics and design features, to selectively evaluate their related effects via artificial spiking implementations [6, 7, 19, 22, 24, 30, 31, 36]. This approach fosters in turn the design of new architectures for artificial tactile sensory systems to enable various application scenarios (such dexterous manipulation in service, assistive, human-augmentation, rehabilitation and industrial robotics [9, 14, 21]), whilst also providing hints and suggestions as to how to further develop experimental protocols and models in neurophysiology [24, 26, 28, 34].

The research strategy that we present in this study capitalizes on the availability of a Micro Electro-Mechanical System (MEMS) piezoresistive tactile sensor developed via silicon microfabrication technologies [2]. The sensor was previously shown to guarantee excellent performances in the coding of normal and shear force [25], of softness and slippage events [3], of textural characteristics [26]. It is appropriate for integration in arrays targeting hand neuroprosthetics and for the implementation of machine learning algorithms enabling autonomous development of motor skills based on sensory information and artificial curiosity [29].

The most recent studies based on such MEMS tactile sensor investigate the neuromorphic conversion of raw sensor outputs to spike codes conveying tactile information (Fig. 1a). Such neuromorphic artificial tactile system [36] grounds on the natural coding observed with microneurographic human recordings [28] and was implemented reproducing mechanosensor dynamics via the Izhikevich spiking neuron model [15]. We assessed the ability of the neuromorphic approach in encoding information about naturalistic textures in a variety of sensing conditions [33]. We found that a decoding based on the fine temporal structure of the spike patterns over a

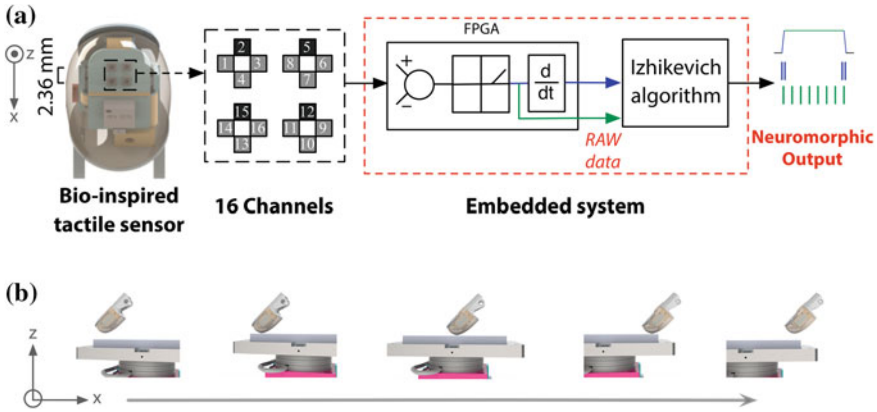


Fig. 1 **a** Processing of tactile information by the neuromorphic finger. From *left* to *right* fingertip structure, channels arrangements, preprocessing of sensors output and injection in the Izhikevich model, neuromorphic output. Green spikes are associated to SA artificial receptors and blue spikes to FA receptors. **b** Phases of passive touch protocol: indentation, sliding, retraction

single channel was able to correctly identify the presented naturalistic texture over a set of 10 stimuli, with a performance between 83 and 97% for indentation forces and tangential sliding velocities ranging between 200 and 600 mN and 10 and 25 mm/s (Fig. 1b). This suggests that our neuromorphic spike trains might contain enough information to convey an adequate sensory feedback to upper limb neuroprostheses [32, 37] and to be useful in robotic applications.

In the present work we will illustrate our methods to implement and evaluate a neuromorphic artificial sense of touch (Sect. 2.1). First we will show our state of the art approach [33], based on a spike metric technique, that results in spike patterns rich enough to allow for a correct off-line decoding of naturalistic stimuli (Sect. 2.2). Then, in Sect. 2.3 we will describe an extension of our approach with the aim to process stimuli in real-time via a neuro-bio-inspired architecture. This biomimetic approach makes possible to decode stimuli while the tactile data stream is gathered and not at the end of the process as in the algorithms used in Sect. 2.2 The second processing layer has a role similar to the Cuneate Nucleus in mammals [11, 18], since it receives the combined inputs of the biomimetic fingertip artificial mechanosensors. Following recent hypotheses on the way tactile information is processed pre-cortically [16, 18] we devised a temporal structure of the connectivity between the two layers such that the post-synaptic neurons are able to encode the angle of contact with the presented object. Finally, we will discuss (Sect. 3) how the results of our neuromorphic approach can contribute to both developing novel robotic artifacts, with particular reference to neuroprostheses, and understanding more about tactile information processing from periphery to cortical areas.

2 Neuromorphic Artificial Sense of Touch

2.1 Neuromorphic Mechanosensors Implementation via Artificial Neuron Spiking Model

In the implemented neuromorphic artificial touch system [36] we modeled with spiking neurons both Merkel Slowly Adapting (SA) type I mechanoreceptors, sensitive to the sustained level of mechanical interaction, and Meissner Fast Adapting (FA) type I mechanoreceptors, mostly sensitive to dynamic changes [1, 38]. The difference between the two artificial mechanoreceptor types relied in the input to the artificial neuron. Sensors outputs were processed and normalized to generate a virtual current as described by Eqs. 1 and 2, where $S_{x+}(t)$ and $S_{x-}(t)$ are the outputs of opponent tethers of a sensor along the x-axis (e.g., 2 and 4, 5 and 7, 12 and 10, 15 and 13, see Fig. 1a) that are subtracted in order to obtain a component related to local shear stress [25]. We introduced a parameter K, in Eq. 2 indicating the gain factor of the sensor voltage when injected as input current to the artificial neuron model. We tested a broad range of values for this parameter and found that high gain factors induced a strong firing rate independently from the stimulus presented, resulting in a less informative temporal structure of spikes. On the other hand, low gain factors led to low firing rate and consequently to a long latency in spike responses affecting the feature extraction. We found that a proper tradeoff between latency and information was achieved for $K = 10,000$, which induces informative responses with a short latency [33]. For dimensionality coherence, the gain factor K is divided per the fixed resistance R regulating the input current to the artificial neuron. In our implementation of SA receptor models the current defined in Eq. 2 corresponded to the external input I (Fig. 1a, Eq. 3), whereas for FA receptor models the external input was given by the time derivative of the current (Fig. 1a, Eq. 4). τ and C_m are introduced as an unitary fixed value of capacitance for dimensionality coherence.

To implement the artificial neuron we selected the Izhikevich model for its ability to exhibit adaptation, which is a key feature of mechanoreceptors. The model combines the biological plausibility of the Hodgkin-Huxley dynamics along with computational efficiency, and it is described by a system of Eqs. (from 3 to 6) that account for the sub-threshold evolution of the membrane potential v and adaptation variable u. Parameters A, B and C are the standard ones for the Izhikevich artificial neuron model. The value of the parameters a and b can be varied to reproduce different kinds of adaptation: a defines the characteristic time of recovery variable, b defines the sensitivity of recovery variable. In case that the membrane potential reached the threshold value (V_{th}) of 30 mV, one spike was released and the membrane potential v was reset to value c and the adaptation variable u was decreased by d as shown in Eq. 6. Parameters c and d contribute as well in defining the adaptation properties of the neuron. We have chosen the values of parameters a, b, c and d to obtain *regular spiking* dynamics [15], which is the case of human finger mechanoreceptors that we want to mimic. Computations were performed in Matlab® environment with a step $\Delta t = 50\,\mu s$.

Table 1 Parameters of Eq. 2–6

K	A	B	C	C_m	R	a	b	c	d	v_{th}
15,000	$\frac{0.04}{sV}$	$\frac{5}{s}$	$\frac{140V}{s}$	1F	1Ω	$\frac{0.02}{s}$	$\frac{0.2}{s}$	−65 mV	8 mV	30 mV

All the parameters of the implemented neuromorphic artificial tactile sensing system are summarized in (Table 1).

$$S_x(t) = S_{x+}(t) - S_{x-}(t) \tag{1}$$

$$I_x(t) = \begin{cases} \frac{K}{R} S_x(t), \, S_x(t) \geq S_{th} \\ 0, \, S_x(t) < S_{th} \end{cases} \tag{2}$$

$$\frac{dv(t)}{dt} = Av(t)^2 + Bv(t) + C - u(t) + \frac{1}{C_m} I_x(t) \tag{3}$$

$$\frac{dv(t)}{dt} = Av(t)^2 + Bv(t) + C - u(t) + \frac{\tau}{C_m} \frac{dI_x(t)}{dt} \tag{4}$$

$$\frac{du(t)}{dt} = a(bv(t) - u(t)) \tag{5}$$

if $(v(t^*) \geq v_{th})$, then a spike occurs at t^* and resulting

$$\begin{cases} v(t^* + \Delta t) \leftarrow c \\ u(t^* + \Delta t) \leftarrow u(t^*) + d \end{cases} \tag{6}$$

Tactile stimuli were presented to the biomimetic fingertip with a passive touch protocol implemented by means of a 2 DoFs mechatronic platform that could indent and slide the surfaces tangentially to the fingertip [27]: in the tactile stimulation sequence the platform was first moved vertically to cause indentation with the finger, then horizontally during the sliding phase and then vertically again to retract from indentation (Fig. 1b).

2.2 *Off-Line Categorization via Spike Distance Metrics*

We first evaluated our neuromorphic approach to artificial touch [36] with 10 different naturalistic tactile stimuli (Fig. 2) [33]. We tested five textile stimuli with different spatial patterns, glass and paper, and three kinds of polymers with different friction (high friction Latex, medium friction artificial skin Bioskin and the low friction cover of the Nintendo® console Wii).

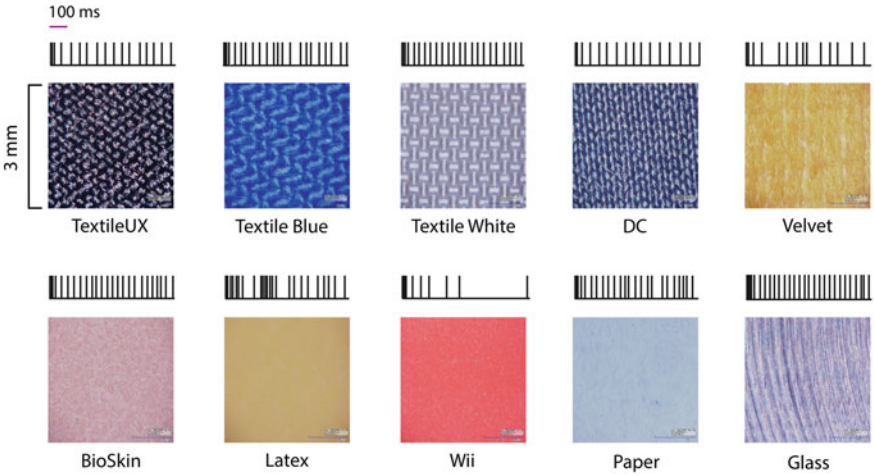


Fig. 2 Textures and representative response. Each panel shows microscope picture of a 3 mm square patch of the tactile stimuli presented to the finger, and, on *top*, a representative neuromorphic response to the stimulus by a SA artificial mechanosensor. Note that glass surface is smooth and the appearing ridges are in transparency from the surface below

We checked whether it was possible to disambiguate between the 10 presented stimuli based on the neuromorphic spike train responses produced during stimuli sliding phase (Fig. 1b) by the SA neuron associated to channel 2 (Fig. 1a). The channel was chosen since it was mostly sensitive to the tangential force along the sliding direction. All channels in similar positions (2, 5, 12 and 15) displayed similar behavior. We based our first decoding algorithm on the spike rate, i.e. on a very simple neural code to discriminate between stimuli, hypothesizing that different stimuli elicited a reliably different number of spikes over the window of interest. We found however that the firing rate of the neurons was not sufficient to discriminate between the inspected textures (decoding performance 68%). Since from visual inspection the temporal structure of spike patterns ranged from almost noisy to almost periodic (e.g., was more regular in textiles than in non-textile materials as shown in Fig. 2), we also performed a two-dimensional k-means clustering decoding (with leave one out validation) combining the mean and the coefficient of variation of single neurons firing rate. This two-dimensional decoding allow performance to raise to 78% (from the 68% of firing rate only) [33] showing that the knowledge of the temporal regularity of the spike response patterns contributes to the discrimination between stimuli.

Building on this result we performed a decoding of the textures evaluating the similarity between spike patterns by means of a spike train metrics [20] widely used in neuroscience: the Victor-Purpura distance [39].

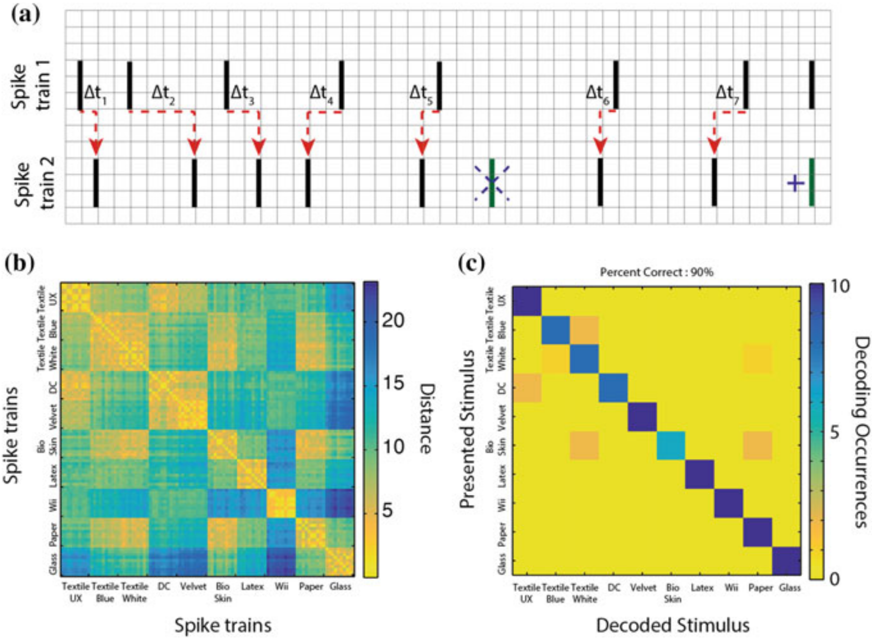


Fig. 3 Victor-Purpura (VP) distance based textures decoding. **a** Example computation of VP distance between two spike trains. The distance of the two example spike trains in the panel is equal to $cost = 2 + q\sum_i \Delta t_i$ where the first term is the cost of 1 spike deletion (*green*, on the *center*) and 1 spike addition (*green*, on the *right*), and the second is the total cost of all spike-time shifts (*red dashed arrows*). **b** VP distances between SA responses for all presentations of all stimuli. Shift cost coefficient $q = 5/s$. **c** Confusion matrix of k-means clustering texture decoding based on VP distances in (**b**) (see Sect. 2.2). Note that each stimulus was presented 10 times so 6/10 textures were identified 100% of the times. Overall performance was 90% for this specific stimulation condition

Briefly, this metrics computes the distance between two spike trains by measuring the minimum cost necessary to transform one spike pattern into the other by means of a sequence of the following two operations: adding/deleting a spike ($cost = 1$) and shifting a spike time of an interval Δt ($cost = q \cdot \Delta t$), as shown in Fig. 3a. The variable q defines the relevant time scale: if we have a spike at $t = A$ in the first train and a spike at $t = B$ in the second train and | A-B | is larger than 2/q then it is more convenient to delete the spike at $t = B$ and insert in the second train a new spike at $t = A$. Coherently, for $q = 0$ shifting spikes is costless hence the distance between two spike trains is equal to the difference between their spike counts, while for high values of q even small jitters in spike timing are associated to high costs. Previous studies of our group showed that the optimal q for this kind of textures and sensors is in the 5–10 s^{-1} range, corresponding to a timescale of 100–200 ms [33]. We measured the VP distance between each pair of responses in the dataset using a cost $q = 5/s$ (Fig. 3b) and then we performed a k-means clustering decoding with leave-one-out validation (Fig. 3c, that shows an example decoding with a 90% performance). Using this VP based approach, the decoding performance rose up to 97% (with a 10% chance level), indicating that the temporal structure of the neuromorphic spike patterns does contain enough information to allow a reliable disambiguation of real life textures even when considering data from a single sensor channel.

2.3 Two Layers Neurobioinspired Real-Time Architecture

The decoding approach presented in the previous section demonstrated the possibility to convey information about real life textures with neuromorphic sensors but contained one overestimation and one underestimation of the decoding power of the neuromorphic responses in real life robotics and neuroprosthetic applications. On one hand decoding performance was overestimated because we performed an offline processing that might not be straightforward to implement online. On the other hand performance was underestimated because we made use of only one channel of the tactile array. We present here an approach that addresses both limitations at once.

We introduced a second layer of artificial neurons collecting inputs from both SA-like and FA-like artificial mechanosensory neurons (Fig. 1). The encoding of shapes in the second layer, as we will see in the following, is based on precise relative spike latency rather than on the overall response pattern. This layer might play a role similar to the one of the Cuneate Nucleus (CN) in the tactile information pathway of mammals [16]. CN neurons were modeled as regular spiking Izhikevich neurons (as for artificial mechanosensors). The input to CN neurons was given by the sum of the excitatory inputs from mechanosensory neurons modeled as current-based post-synaptic potentials [23], such that the differential equations determining the evolution of their dynamics are:

$$\frac{dv(t)}{dt} = Av(t)^2 + Bv(t) + C - u(t) + J\sum\nolimits_{i\in PRE} PSP(t - t_i^+) \tag{7}$$

$$\frac{du(t)}{dt} = a(bv(t) - u(t)) \tag{8}$$

if $(v(t^*) \geq v_{th})$, then a spike occurs at t^* and resulting

$$\begin{cases} v(t^* + \Delta t) \leftarrow c \\ u(t^* + \Delta t) \leftarrow u(t^*) + d \end{cases} \tag{9}$$

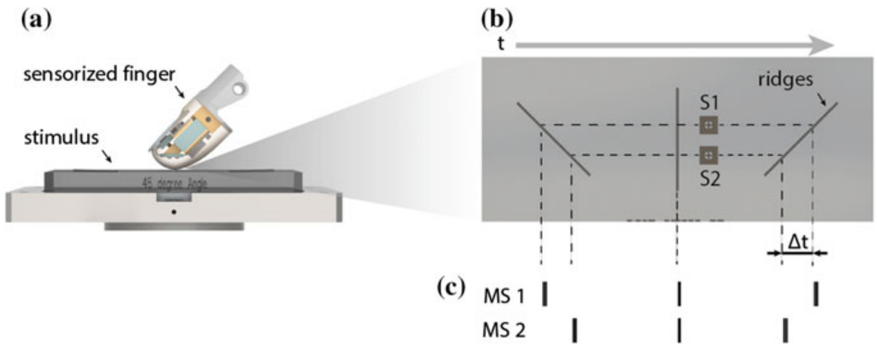


Fig. 4 Processing of stimulus orientation. **a** Contact between fingertip and stimulus during sliding. **b** contact times between the two vertically aligned sensors with ridges having an angle of −45°, 0° and 45° relatively to the direction orthogonal to stimulus-finger relative motion. **c** Illustration of neuromorphic mechanosensors responses to the three stimuli

where A, B, C, a, b, c, d have the values listed in Table 1, J is the fixed synaptic strength (8 mV/ms), t_i^+ are the times of the spikes fired by the pre-synaptic neuron i, PRE is the set of the pre-synaptic neurons for the neuron considered, and $PSP(t)$ is a bi-exponential function mimicking the temporal evolution of AMPA synapses Post-Spike Potentials, with rise time 0.5 ms and decay time 2 ms [23].

In our current simplified model, neurons in CN do not differ for the set of pre-synaptic neurons, but for the conduction delays of their inputs from peripheral neurons. We want to implement in this way a condition similar to the one hypothesized by Johansson and Flanagan [16] in which differential delays between peripheral spike trains play a role in the encoding of information.

If response latency and delay difference approximately compensate the post-synaptic delay, the CN neuron will receive two superimposed excitatory stimuli, increasing the probability that the membrane potential will cross spike threshold and fire. In this way, differential delays make possible to have angle-sensitive CN neurons, i.e., neurons firing only when the contact with the sensors occur with a given latency, which for constant sliding speed corresponds to a specific presented edge angle (as illustrated in Fig. 4b). This computational architecture, exploiting the details of the convergent connectivity between mechanosensors and Cuneate Nucleus, will be very easy to decode and straightforward to implement online while gathering the tactile data stream.

In order to preliminarily test the properties of this architecture we devised a set of stimuli consisting on 3 ridges forming an angle of −45°, 0° and 45° with the direction orthogonal to the stimulus sliding (Fig. 4), 3D printed on the same surface. The surfaces were presented over the course of a single sliding with the same stimulation conditions (20 mm/s tangential sliding, 500 mN indentation force) used in the naturalistic textures (decoding analysis in Sect. 2.2). Our proof-of-principle test involved only two mechanosensory channels (Fig. 4), but we considered both SA and FA outputs from them. The relative latency of the neuromorphic responses

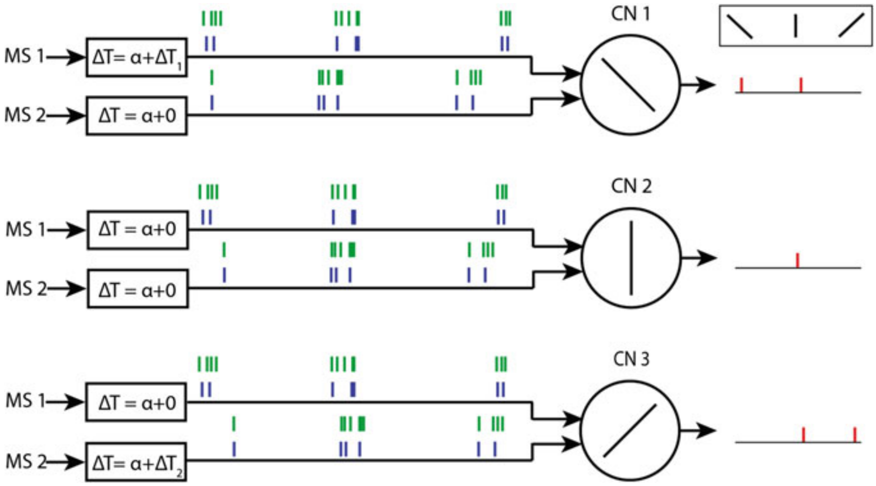


Fig. 5 Cuneate processing of mechanoreceptors inputs. Every row shows, from *left* to *right*, SA-like (*green*) and FA-like (*blue*) spike train responses for a single stimulation sequence as sensed by two channels converging to the same Cuneate nucleus neuron. In each row the two channels impinge on a different CN neuron with a specific relative delay. Due to these delays, CN1 responds mainly to the first two stimuli, CN2 mainly to the vertical one, and CN3 to the last two

of the two sensors shifted (also changing sign) during the sliding as a function of stimulus angle (Fig. 4b). Following the principles illustrated by Fig. 3, thanks to a suited delays distribution, these latencies will result into the activation of a restricted set of CN neurons (Fig. 5). This set being specific and reliable (Fig. 6), it allowed to reconstruct the angle of the detected edge via the lookup strategy shown in Table 2. Note that although the 0° angle ridge induces a response in both CN2 and CN3 the intensity of the response is different.

3 Perspectives

The sense of touch is currently the least understood among the human senses and the neuronal mechanisms of the somatosensory system are still under hypothetical modeling. For instance, little is known about whether and how the Cuneate Nucleus contributes to signal processing and integration, although recent results show sparse encoding of stimulus shape in Cuneate neurons [18]. Due also to the lack of experimental results, the only artificial computational model of Cuneate Nucleus processing of tactile sensory stimuli that we are aware of is the one proposed by Arleo and colleagues [6, 7]. Here we move forward from this model: first, we used spiking neuron models instead of spike-response model, second (and more important) because we used the model to test the feasibility of the encoding-though-delay model accounted

Fig. 6 Full path of ridge angle information processing. **a** Presented ridges. **b** Raster plots of responses elicited over 10 trials by different ridges in the FA and SA mechanosensors associated to the two channels. **c** Raster plots of responses elicited over 10 trials in the three CN neurons illustrated in Fig. 5

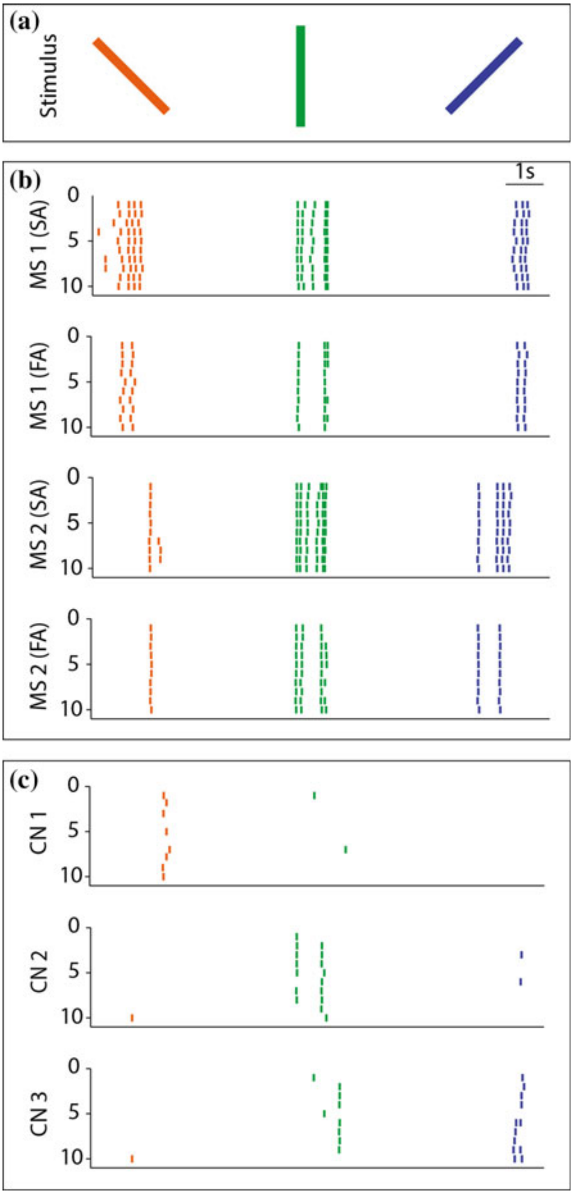


Table 2 Stimulus decoding table. For each column X marks the activated CN neurons. Note that each spatial pattern of activation is univocally associated to a single stimulus, leading to a feasible decoding

		NO stimulus	\	|	|	|	|	ND	/
Active cuneate neurons	CN 1		X	X			X	X	
	CN 2			X	X	X	X		
	CN 3					X	X	X	X

by Johansson and Flanagan and by Jörntell and Hayward [16, 18]. Our results suggest that this process could indeed be used by the brain to generate the sparse encoding of shapes in the Cuneate Nucleus. This is an example of how the neuromorphic approach can generate a virtuous circle of interactions between neurophysiological results and applications in robotics.

The main advantage of a neuromorphic implementation is the leanness, since the burden of data computation and storage is minimal: information is conveyed by means of the timestamp of the binary spike events whereas traditional implementations require a continuous sampling of amplitude-modulated data. As shown in the case-study examples presented in this work, relevant information is preserved to allow the decoding of textural and geometrical features. Such results pave the way towards possible future applications of the developed neuromorphic touch system in limb neurosthetics or in industrial applications requiring the artificial categorization of tactile qualities.

Acknowledgements This work was supported in part by the NEBIAS European project (NEurocontrolled BIdirectional Artificial upper limb and hand prosthesis; EU-FP7-ICT-611687), by the PRIN/HandBot Italian project (Biomechatronic hand prostheses endowed with bio-inspired tactile perception, bi-directional neural interfaces and distributed sensori-motor control; CUP: B81J12002680008; prot.: 20102YF2RY), by the NanoBioTouch European project (Nano-resolved multi-scale investigations of human tactile sensations and tissue engineered nanobiosensors; EU-FP7-NMP-228844), and by the Italian Ministry of Foreign Affairs and International Cooperation, Directorate General for Country Promotion (Economy, Culture and Science) - Unit for Scientific and Technological Cooperation, via the Italy-Sweden bilateral research project on “Brain network mechanisms for integration of natural tactile input patterns”.
The authors gratefully thank Mr. Giacomo Spigler and Dr. Lucia Beccai for previous collaboration in this project, Dr. Simon Johnson for providing part of the experimented tactile stimuli, and Prof. Silvestro Micera, Prof. Henrik Jörntell, Prof. Johan Wessberg and Prof. Paolo Dario for thoughtful scientific discussions.

References

1. Abraira, V.E., Ginty, D.D.: The sensory neurons of touch. Neuron **79**(4), 618–639 (2013)
2. Beccai, L., Roccella, S., Arena, A., Valvo, F., Valdastri, P., Menciassi, A., Carrozza, M.C., Dario, P.: Design and fabrication of a hybrid silicon three-axial force sensor for biomechanical applications. Sens. Actuators A Phys. **120**(2), 370–382 (2005)
3. Beccai, L., Roccella, S., Ascari, L., Valdastri, P., Sieber, A., Carrozza, M.C., Dario, P.: Development and experimental analysis of a soft compliant tactile microsensor for anthropomorphic artificial hand. IEEE/ASME Trans. Mech. **13**(2), 158–168 (2008)
4. Bensmaïa, S.J., Hollins, M.: The vibrations of texture. Somatosens. Mot. Res. **20**(1), 33–43 (2003)
5. Blake, D.T., Hsiao, S.S., Johnson, K.O.: Neural coding mechanisms in tactile pattern recognition: the relative contributions of slowly and rapidly adapting mechanoreceptors to perceived roughness. J. Neurosci. **17**(19), 7480–7489 (1997)
6. Bologna, L.L., Pinoteau, J., Passot, J.B., Garrido, J.A., Vogel, J., Vidal, E.R., Arleo, A.: A closed-loop neurobotic system for fine touch sensing. J. Neural Eng. **10**(4), 046019 (2013)
7. Bologna, L.L., Pinoteau, J., Brasselet, R., Maggiali, M., Arleo, A.: Encoding/decoding of first and second order tactile afferents in a neurorobotic application. J. Phys. Paris **105**(1), 25–35 (2011)
8. Connor, C.E., Johnson, K.O.: Neural coding of tactile texture: comparison of spatial and temporal mechanisms for roughness perception. J. Neurosci. **12**(9), 3414–3426 (1992)
9. Dario, P.: Tactile sensing: technology and applications. Sens. Actuators A Phys. **26**(1), 251–256 (1991)
10. Dario, P., Hannaford, B., Takanishi, A.: Guest editorial special issue on biorobotics. IEEE Trans. Robot. **24**(1), 3–4 (2008)
11. Hayward, V., Terekhov, A.V., Wong, S.-C., Geborek, P., Bengtsson, F., Jörntell, H.: Spatiotemporal skin strain distributions evoke low variability spike responses in cuneate neurons. J. R. Soc. Interface **11**(93), 20131015 (2014)
12. Hollins, M., Risner, S.R.: Evidence for the duplex theory of tactile texture perception. Percept. Psychophys. **62**(4), 695–705 (2000)
13. Hollins, M., Bensmaïa, S.J., Sean, W.: Vibrotactile adaptation impairs discrimination of fine, but not coarse, textures. Somatosens. Motor Res. **18**(4), 253–262 (2001)
14. Howe, R.D.: Tactile sensing and control of robotic manipulation. Adv. Robot. **8**(3), 245–261 (1993)
15. Izhikevich, E.M., et al.: Simple model of spiking neurons. IEEE Trans. Neural Net. **14**(6), 1569–1572 (2003)
16. Johansson, R.S., Flanagan, J.R.: Coding and use of tactile signals from the fingertips in object manipulation tasks. Nat. Rev. Neurosci. **10**(5), 345–359 (2009)
17. Jones, L.A., Lederman, S.J.: Human Hand Function. Oxford University Press, Oxford (2006)
18. Jörntell, H., Bengtsson, F., Geborek, P., Spanne, A., Terekhov, A.V., Hayward, V.: Segregation of tactile input features in neurons of the cuneate nucleus. Neuron **83**(6), 1444–1452 (2014)
19. Kim, E.K., Sugg, K.B., Langhals, N.B., Lightbody, S.M., Baltrusaitis, M.E., Urbanchek, M.G., Cedema, P.S., Gerling, G.J.: An engineered tactile afferent modulation platform to elicit compound sensory nerve action potentials in response to force magnitude. In: World Haptics Conference (WHC), 2013, pp. 241–246. IEEE (2013)
20. Kreuz, T.: Measures of spike train synchrony. Scholarpedia **6**(10), 11934 (2011)
21. Lee, M.H., Nicholls, H.R.: Review article tactile sensing for mechatronics–a state of the art survey. Mechatronics **9**(1), 1–31 (1999)
22. Lee, W.W., Cabibihan, J., Thakor, N.V.: Bio-mimetic strategies for tactile sensing. In: IEEE SENSORS, 2013, pp. 1–4. IEEE (2013)
23. Mazzoni, A., Panzeri, S., Logothetis, N.K., Brunel, N.: Encoding of naturalistic stimuli by local field potential spectra in networks of excitatory and inhibitory neurons. PLoS Comput. Biol. **4**(12), e1000239 (2008)

24. Mitchinson, B., Pearson, M., Pipe, T., Prescott, T.J.: Biomimetic robots as scientific models: a view from the whisker tip. Neuromorphic and Brain-Based Robots, pp. 23–57. Cambridge University Press, Cambridge (2011)
25. Oddo, C.M., Valdastri, P., Beccai, L., Roccella, S., Carrozza, M.C., Dario, P.: Investigation on calibration methods for multi-axis, linear and redundant force sensors. Meas. Sci. Technol. **18**(3), 623 (2007)
26. Oddo, C.M., Controzzi, M., Beccai, L., Cipriani, C., Carrozza, M.C.: Roughness encoding for discrimination of surfaces in artificial active-touch. IEEE Trans. Robot. **27**(3), 522–533 (2011)
27. Oddo, C.M., Beccai, L., Vitiello, N., Wasling, H.B., Wessberg, J., Carrozza, M.C.: A mechatronic platform for human touch studies. Mechatronics **21**(3), 604–613 (2011)
28. Oddo, C.M., Beccai, L., Wessberg, J., Wasling, H.B., Mattioli, F., Carrozza, M.C.: Roughness encoding in human and biomimetic artificial touch: spatiotemporal frequency modulation and structural anisotropy of fingerprints. Sensors **11**(6), 5596–5615 (2011)
29. Pape, L., Oddo, C.M., Controzzi, M., Cipriani, C., Förster, A., Carrozza, M.C., Schmidhuber, J.: Learning tactile skills through curious exploration. Front. Neurorobot. **6**, (2012)
30. Pearson, M.J., Mitchinson, B., Sullivan, J.C., Pipe, A.G., Prescott, T.J.: Biomimetic vibrissal sensing for robots. Philos. Trans. R. Soc. Lond. B Biol. Sci. **366**(1581), 3085–3096 (2011)
31. Rager, D.M., Alvares, D., Birznieks, I., Redmond, S.J., Morley, J.W., Lovell, N.H., Vickery, R.M.: Generating tactile afferent stimulation patterns for slip and touch feedback in neural prosthetics. In: 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2013, pp. 5922–5925. IEEE (2013)
32. Raspopovic, S., Capogrosso, M., Petrini, F.M., Bonizzato, M., Rigosa, J., Di Pino, G., Carpaneto, J., Controzzi, M., Boretius, T., Fernandez, E., et al.: Restoring natural sensory feedback in real-time bidirectional hand prostheses. Sci. Transl. Med. **6**(222), 222ra19–222ra19 (2014)
33. Rongala, U.B., Mazzoni, A., Oddo, C.M.: Neuromorphic artificial touch for categorization of naturalistic textures. IEEE Trans. Neural Net. Learn. Syst. **PP**(99), 1 (2015). doi:10.1109/TNNLS.2015.2472477. ISSN 2162-237X
34. Scheibert, J., Leurent, S., Prevost, A., Debrégeas, G.: The role of fingerprints in the coding of tactile information probed with a biomimetic sensor. Science **323**(5920), 1503–1506 (2009)
35. Service, R.F.: Minds of their own. Science **346**(6206), 182–183 (2014). doi:10.1126/science.346.6206.182. http://www.sciencemag.org/content/346/6206/182.short
36. Spigler, G., Oddo, C.M., Carrozza, M.C.: Soft-neuromorphic artificial touch for applications in neuro-robotics. In: 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), 2012, pp. 1913–1918. IEEE (2012)
37. Tan, D.W., Schiefer, M.A., Keith, M.W., Anderson, J.R., Tyler, J., Tyler, D.J.: A neural interface provides long-term stable natural touch perception. Sci. Transl. Med. **6**(257), 257ra138–257ra138 (2014)
38. Vallbo, Å.B., Johansson, R.S., et al.: Properties of cutaneous mechanoreceptors in the human hand related to touch sensation. Hum. Neurobiol. **3**(1), 3–14 (1984)
39. Victor, J.D., Purpura, K.P.: Nature and precision of temporal coding in visual cortex: a metric-space analysis. J. Neurophys. **76**(2), 1310–1326 (1996)
40. Yoshioka, T., Bensmaia, S.J., Craig, J.C., Hsiao, S.S.: Texture perception through direct and indirect touch: an analysis of perceptual space for tactile textures in two modes of exploration. Somatosens. Mot. Res. **24**(1–2), 53–70 (2007)
41. Yoshioka, T., Gibb, B., Dorsch, A.K., Hsiao, S.S., Johnson, K.O.: Neural coding mechanisms underlying perceived roughness of finely textured surfaces. J. Neurosci. **21**(17), 6905–6916 (2001)

Robot Creation from Functional Specifications

Ankur M. Mehta, Joseph DelPreto, Kai Weng Wong, Scott Hamill, Hadas Kress-Gazit and Daniela Rus

1 Introduction

Although robots have become prevalent in academic and industrial applications, there is a knowledge barrier which prevents them from fully integrating into daily life. The creation of robots typically requires deep understanding of the available tools as well as the expertise to combine parts in a way that will achieve some desired behavior. Due to this intensive methodology, a discrepancy often arises between the end users and the robot creators, leading to general-purpose robots being created before a task is fully specified.

The long-term vision is to instead enable the creation of custom personal robots on-demand by encapsulating the needed low-level knowledge into computational tools that can automatically address a user's high-level robotic needs. Typical end users will have a task that they want the robot to perform and an understanding of the task requirements, but may not be able to construct or even assemble integrated programmed electromechanical mechanisms to realize a solution. This paper

A.M. Mehta (✉)
University of California, Los Angeles, Los Angeles, CA, USA
e-mail: mehtank@ucla.edu

J. DelPreto · D. Rus
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: delpreto@csail.mit.edu

D. Rus
e-mail: rus@csail.mit.edu

K.W. Wong · S. Hamill · H. Kress-Gazit
Cornell University, Ithaca, NY, USA
e-mail: kw358@cornell.edu

S. Hamill
e-mail: sbh92@cornell.edu

H. Kress-Gazit
e-mail: hadaskg@cornell.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_36

therefore moves towards a system that can compile a high level behavioral description into a completely fabricable robot design including software.

This work improves and expands the robot compiler system presented in [18]. Whereas the earlier system required a user to provide the complete structural specification for a robot design, from the selection of components to their connectivity and subsequent geometric layout, the system presented herein accepts a more intuitive functional specification as initial input – a description of the relationships between atomic robot action primitives. The selection of components from a library is then aided by automated filtering, and a geometric layout is generated from basic positional constraints. Connectivity and parameter relationships are automatically derived from the generated geometric layout and robot controller. Furthermore, the system presented here enhances design iteration by allowing users to simulate the robot controller prior to fabrication.

2 Problem Formulation and Contributions

The goal of this work is to be able to generate a complete integrated design and controller for a custom robot on demand. To minimize the requirements on an end user, the inputs to the system must be as high level as possible, with the automation of low-level decisions. The process begins with a functional description of the desired behavior, and ends with a programmed printed electromechanical machine that executes the described task. The presented approach decomposes the robot design process into a series of stages that facilitates rapid prototyping and design iteration.

To define the desired behavior, the user first writes a functional specification, or task specification, in Structured English [14], capturing the requirements and goals of the robot. Though not to the level of natural language programming, this allows a casual user to describe rather than command how the robot should operate through the use of primitive elements called propositions. The ability to decompose a desired task solution into this functional specification is the only technical requirement on the user; no other mechanical, electrical, or computer engineering skills are assumed. The input specification maps directly into Linear Temporal Logic (LTL) formulas, which are the input to a controller synthesis algorithm [4]. If there exists a finite state machine capable of achieving the goals given an adversarial environment, a controller will be generated.

The propositions of this specification are then used to create a structural specification – a specification used to build the custom robot. The structural specification is constructed by mapping the propositions to parameterized robotic building blocks drawn from a robot component library. The system filters the library to recommend components appropriate to each proposition, aiding the user in grounding the specification. In addition, the system assesses the mapped propositions for possible behavioral conflicts, correcting the functional specification as needed. Depending on the components chosen by the user, a single functional specification may generate varied robot configurations that accomplish the same goal. The selected components are then automatically configured and connected, though advanced users can edit and create custom configurations. The result is a parameterized robot design capable of accomplishing the task.

Upon setting desired parameters, the structural specification can be compiled to synthesize printable mechanical fabrication files, electrical wiring instructions, and code for the custom robot. The robot controller generated from the functional specification can be analyzed in simulation, then converted to microcontroller code and directly programmed onto the robot. Once the user has built the robot with the given instructions, the desired behavior will be carried out by the created robot.

The full algorithm for creating a robot from a Structured English specification is described in pseudocode listing 1.

Algorithm 1 Robot creation from functional specifications

```
Input Structured English specification file
Convert Structured English into LTL
$FSM \leftarrow$ Compile LTL into controller automaton

$L \leftarrow$ Load Component library
$G \leftarrow \emptyset$                                        ▷ Grounding list
$A \leftarrow$ Filter($L$, dataConsumers)                       ▷ Actuators
for all $a \in FSM$[actuators] do
    if $a$ requires physical output then
        $A* \leftarrow$ Filter($A$, hasMechanicalPorts)
    else
        $A* \leftarrow$ Filter($A$, $\neg$hasMechanicalPorts)
    end if
    $c \leftarrow$ User select from $A*$
    $G := G \cup \{$Ground$(a, c)\}$
end for
$S \leftarrow$ Filter($L$, dataGenerators)                      ▷ Sensors
for all $s \in FSM$[sensors] do
    $c \leftarrow$ User select from $S$
    $G := G \cup \{$Ground$(s, c)\}$
end for

Component $\leftarrow$ Core($FSM$)
for all $g \in G$ do
    if $g$.component has MechanicalPort then
        Attach(Component, $g$.component.structure)
    end if
    Connect(Component, $g$.component.signal)
end for

Save Component as structural specification
Output fabrication files for Component
```

The particular contributions of this work are:

- a process for grounding the propositions of an LTL specification to components from a design library to generate a user-guided robot configuration,

- automatic generation of a complete structural specification, including user-guided physical layout and automated controller synthesis, for the compilation of integrated robot designs,
- a process to detect potential behavioral conflicts during the proposition grounding process, and to automatically correct the LTL specification accordingly,
- filtering algorithms to simplify user interactions with the robot compiler API,
- an implementation of all of the above into an integrated end-to-end system generating robots from a Structured English task description, and
- sample robots generated using the system.

3 Related Work

3.1 Functional Specification

There has been an increasing interest in the automatic construction of provably-correct robot controllers from high-level or temporal logic task specifications in the robotics community. These controllers, if successfully synthesized, will behave as specified in the mission statements.

The synthesis and execution of these controllers from temporal logic specifications have been shown by [6, 12]. Groups have since tackled a variety of challenges using these controllers, such as the problem of a changing workspace during controller execution [1, 16], conducting motion planning for robots given temporal goals [2], or generating optimized robot trajectories from temporal logic task specifications [24]. These controllers are also used to control multiple robots [11].

The AI and planning community also has planning languages, from STRIPS [7] to PDDL [17] and more extensions, to create functional specification for machines. Using the planning languages, a problem, or functional specification, is defined and solved with the composition of different actions that each consist of preconditions or post-conditions. The generated plans are similar to the controllers generated from high-level task specifications. In this paper, we use high-level task specifications to generate provably-correct robot controllers out of preference.

The functional specification system for this work stems mostly from [13]. Given a robot model and its environment, controllers that satisfy high-level task specifications are composed automatically. The synthesized controllers also respond to different environment behaviors during controller execution.

3.2 Robot Creation

There has previously been substantial work regarding processes to fabricate robots. For example, robots have been developed from 2D processes [3, 19, 22] using a range of materials at many different size scales. Some of these methods have also been applied to rapid prototyping [9], although the design phase typically still requires significant time and expertise. Efforts have been made to automate the decomposition of 3D shapes into 2D fold patterns [5, 15, 23], but these often do not address compliant or kinematic structures. Furthermore, these works remain within the realm of designing desired structures rather than abstracting the design input to a task-based level.

There has also been significant work on modular robotic systems and behavior [10, 21, 25, 26]. While these systems provide substantial configurational flexibility, modular robots often lack the specialized physical components to address specific behavioral tasks.

This work builds most directly upon the robot compiler presented in [18], which describes a system for generating integrated mechanical, electrical, and software designs for custom robots using a modular component library. The system is extended here by abstracting the user input to a higher level: instead of starting with structural specifications, the user can now begin with a desired behavioral task. Once this high-level task has been processed to create functional requirements, the robot compiler's component library and computational tools are used to generate fully functional origami-inspired foldable robots.

4 Design Flow

To facilitate the rapid prototyping of custom robots from a description of desired behavior, the proposed approach here creates a user-friendly environment by providing a suite of integrated tools that break the process described in Algorithm 1 into a series of well-defined, computer-aided stages, illustrated in Fig. 1. To detail this process, we will consider the following example:

Example 1 The user wants to build a robot that can conduct a pick-and-place grasper task. When the robot receives a request from the user, it will move to a pick-up location and wait for an object to be presented. The robot will then pick up the object, return to the original position, release the object, and inform the user that it has completed the task.

4.1 Behavioral Description to Functional Specification

To create a custom robot from a description of desired behavior, the user starts by writing a mission specification for the task to be conducted. The specification of Example 1 is shown in Fig. 2. Using the Linear Temporal Logic MissiOn Planning toolkit (LTLMoP) [8], the user can write a specification in Structured English by first defining different types of binary propositions. These propositions are abstracted

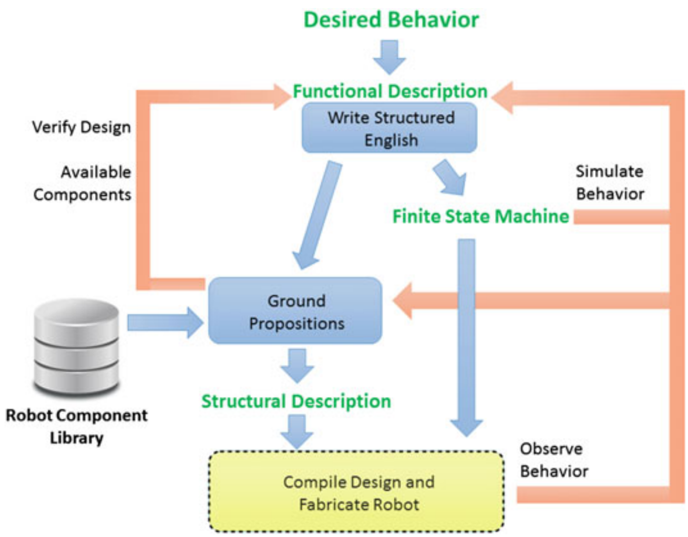
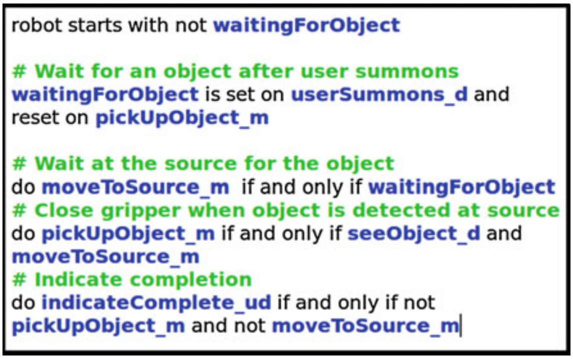


Fig. 1 The system aids robot design by decomposing the procedure into a series of manageable stages. The process can also become iterative at each stage, using feedback from simulation or fabricated devices to encourage rapid prototyping. The *dotted yellow box* contains the previous work from [18], employed by the system in this paper

Fig. 2 The desired behavior of a robotic grasper can be cast into Structured English, from which a finite state machine is automatically generated and a functional description is naturally extracted



```
robot starts with not waitingForObject

# Wait for an object after user summons
waitingForObject is set on userSummons_d and
reset on pickUpObject_m

# Wait at the source for the object
do moveToSource_m  if and only if waitingForObject
# Close gripper when object is detected at source
do pickUpObject_m if and only if seeObject_d and
moveToSource_m
# Indicate completion
do indicateComplete_ud if and only if not
pickUpObject_m and not moveToSource_m
```

from the robot location and actions and its environment. The propositions can be divided into four different types:

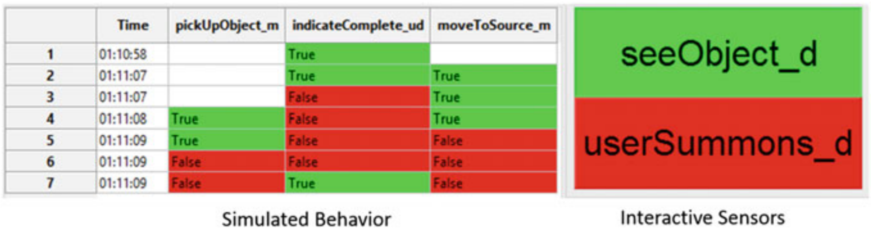
- Region propositions: If a map is given to the robot, the map can be decomposed into different regions, with each region being one proposition. During the controller execution, only one of the region propositions is true at any given time, representing the current location of the robot. In Example 1, there are no region propositions for simplicity.
- Sensor propositions: These are propositions abstracted from the robot's surrounding environment. In Example 1, the presence of an object is abstracted into a

proposition called “seeObject_d” that is true when an object is observed and false otherwise. Note that even though these propositions describe different sensing capabilities of the robot, they are independent of how this capability is implemented on the custom robot; the propositions specify the functionality of the component rather than the actual structural component. In Example 1, “userSummons_d” is also a sensor proposition.

- Actuator propositions: These are propositions abstracted from the possible actions of the robot. In Example 1, activating an actuator proposition “moveToSource_m” specifies that the robot should move to the pick-up location (and if the proposition is deactivated it implies the robot should be at the drop-off location). As with the sensor propositions, these actions are functional rather than structural and therefore independent of implementation; for example, a robot with legs may move differently than a robot with wheels. In Example 1, “pickUpObject_m” and “indicateComplete_ud” are also actuator propositions.
- Custom propositions: These are propositions that are directly linked to neither robot sensing nor actions but that are necessary for specifying more complex behaviors. In Example 1, once “userSummons_d” becomes true, the proposition “waitingForObject” becomes and remains true until “pickUpObject_m” is true.

Using these propositions, the user can follow the grammar outlined in [14] to write a specification, and a robot controller can be generated with the synthesis algorithm in [4]. This correct-by-construction robot controller, in the form of a finite-state machine, will be automatically generated using the toolkit if the mission statement from the user is feasible regardless of how the environment behaves. The finite state machine generated for Example 1 can be found in the supplementary material at http://web.mit.edu/mehtank/www/isrr2015/.

Once a controller is created, the finite state machine can be evaluated with an integrated simulation engine in which sensors can be interactively triggered and the proposition states can be visualized. A sample visualization of the engine is shown in Fig. 3 for Example 1. This facilitates an iterative process in which the user can immediately see how the robot would behave and adjust the specification or functional requirements accordingly.



	Time	pickUpObject_m	indicateComplete_ud	moveToSource_m
1	01:10:58		True	
2	01:11:07		True	True
3	01:11:07		False	True
4	01:11:08	True	False	True
5	01:11:09	True	False	False
6	01:11:09	False	False	False
7	01:11:09	False	True	False

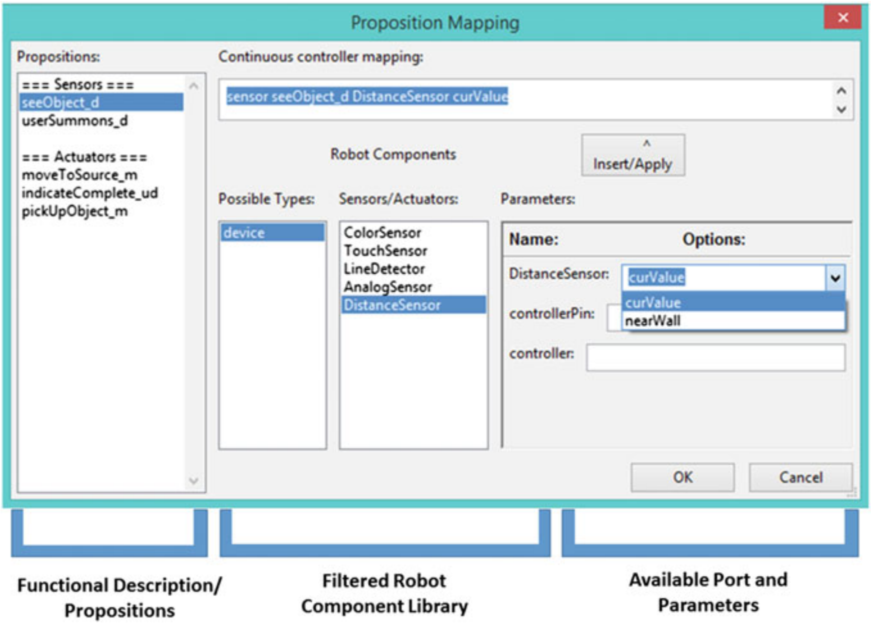
Fig. 3 The generated finite state machine can be simulated to ensure desired behavior and encourage iterative design. Here, the behavior of a pick-and-place grasper is being simulated. The simulation displays the number of state changes, the time changes occur, and the status of the propositions

4.2 *Functional Description to Structural Specifications*

4.2.1 Grounding

Once a functional description, or specification, of the robot has been obtained from the behavioral description, a physical instantiation of the robot that achieves the target task must be determined. In particular, the action and sensing tasks to be performed by the robot can now be grounded to available robot components to generate a structural description of the robot.

To ground the functional propositions to structural components, the grounding editor in the toolkit, modified from [8] for robot creation as shown in Fig. 4, first retrieves the library of possible modular robotic components from the robot compiler [18]. These components include basic building blocks as well as previously constructed assemblies, each designed to implement a specific behavior. Each component in the library encapsulates the design and fabrication information relevant to the component, including the mechanical structure, the electrical properties and the software of the component, and is parameterized to allow customizability.



Propositions:

=== Sensors ===
seeObject_d
userSummons_d

=== Actuators ===
moveToSource_m
indicateComplete_ud
pickUpObject_m

Continuous controller mapping: sensor seeObject_d DistanceSensor curValue

Possible Types: device

Sensors/Actuators: ColorSensor, TouchSensor, LineDetector, AnalogSensor, DistanceSensor

Parameters:

Name:	Options:
DistanceSensor:	curValue
controllerPin:	
controller:	

Proposition Mapping
×
Robot Components
Insert/Apply
curValue
nearWall
OK
Cancel
Functional Description/ Propositions
Filtered Robot Component Library
Available Port and Parameters

Fig. 4 The functional description can be converted to a structural description by selecting modular components from a robot library for each action and sensor proposition. Filtered lists of possibilities are automatically provided, and the user can choose to customize them by setting parameters or simply accept the default values

The components are currently divided into three types: mechanical components, which require constructed structural elements to interface an electromechanical transducer with the environment; device components, which are discrete devices with a completely self-contained action; and UI components, which are purely virtual components that include smartphone interface elements such as sliders or toggle switches. A user can specify desired possible component type(s) for proposition by suffixing its name; based on that suffix, the grounding editor displays a filtered list of allowable components. For the case of Example 1, a list of mechanical actuator components will be shown for the proposition `pickUpObject_m`, while a list of device and UI actuator components will be shown for the actuator proposition `indicateComplete_ud`.

The user can then ground each proposition to one of the available components on the filtered list to obtain the desired actions and sensing capabilities. It is possible that no library component can adequately satisfy a particular proposition, indicating that the proposition is too complex given the existing contents of the library. In this case, the user can either modify the original specification to decompose that proposition into simpler constructs, or create a new component to satisfy the needed behavior. In the latter case, the user would write a new functional specification to define the needed component in terms of simpler propositions, and adding the successful design back into the library.

When a component is chosen from the library, a list of the component parameters is also presented so the user can customize the component if needed. Through this process, a mapping is created between the propositions and the robot components. More specifically, each actuator and sensor proposition is assigned to a port on a component.

It is possible to map more than one proposition to a given component. To ensure conflicting behaviors do not occur, the mapping interface evaluates each mapped proposition and modifies the original functional specification to include mutual exclusions of propositions mapped to the same component. The user is then informed of this modification so they can ensure that the desired behavior is still achieved. The grounding editor creates a close-loop design process by providing feedback to the user through amendments to the functional specification based on the structural specification. With the grounding editor, not only the functional specification affects the structural specification, but the structural specification set by the user also changes the functional specification.

When the compiler processes the design, it will automatically insert multiplexers as appropriate to ensure that the correct command is sent to the component. It should also be noted that some of the components employ analog signals; to meet the boolean requirements of the generated controller, analog sensors get thresholded before becoming inputs to the finite state machine, while the binary actuator commands from the controller are scaled to a user-specified analog value before being applied to the device.

Some possible groundings of propositions in Example 1 are shown in Fig. 5. The conversion of functional description to structural specification is aided by the toolkit but is ultimately chosen by the user; the user asserts control over the design according to personal preference and task-specific requirements, such as environmental consid-

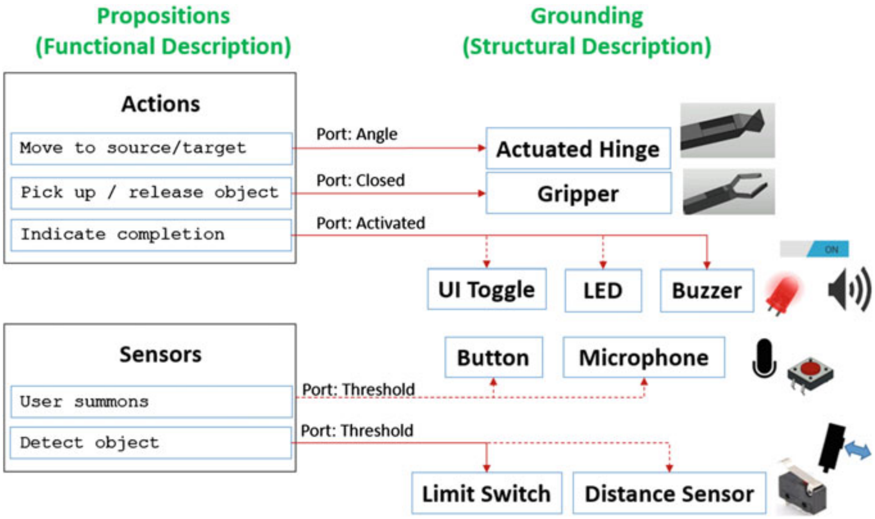


Fig. 5 For each functional proposition needed for the robotic grasper, there are numerous possible robot components in the library that can be used for implementation. Here, a few such options are shown and the *solid lines* indicate those chosen for the current experiment

eration and component availability. Since there are often many components which can be grounded to the same proposition, many different robots can result from the same functional description. For example, a human-generated input may be mapped to a button, a microphone, or a UI element, while an indicator action may be mapped to a light, a buzzer, or a flag waver. This approach simplifies and guides the robot design process for novice users without restricting expert users.

4.2.2 Mechanical Connections

A structural specification also requires the geometric layout of the physical components into a single integrated electromechanical device. Though the design space of geometric configurations can get intractably large, the system once again aids a novice user by presenting a reduced set of options to handle general cases. An expert user can bypass the filter and create arbitrary mechanical connections constrained only by available interface points designed to limit component collision.

The mechanical-type components preferentially presented for grounding are designed to mostly fit into a rectangular prism bounding box. This allows for physical composition by tiling the selected components into orthogonal regions. The user can select whether a particular component belongs in the front, back, left, right, or center of the robot; the system then iterates through the full list of mechanical components and appends them onto the core controller module, growing the robot as it goes. Components with parameterized dimensions get scaled to fit the entire collection.

In a similar manner, the remaining non-mechanical device-type components then get mounted on any exposed face of the robot. The user can specify whether the device should be facing forwards, backwards, left, right, up, or down, and the system will mount the device onto the respective structures assembled in the previous step.

4.3 Integrated Robot Fabrication

Once the complete structural specifications have been generated, the robot compiler processes the modular design into design files for the complete robot [18], producing mechanical fabrication files, electrical wiring instructions, and microcontroller code.

The mechanical structure is fabricated using an origami-inspired cut-and-fold process: the generated fabrication file gets sent to a desktop vinyl cutter to be cut from a 2D sheet of plastic, and the user then follows the folding instructions and the generated wiring instructions to fabricate the robot. Finally, the automatically generated robot software can be loaded onto the main controller, ranging from low level drivers to the implementation of the finite state machine created from the LTL specification.

The robot can then simply be powered on to achieve the task specifications initially provided by the user.

5 Assumptions, Generalizations and Guarantees

5.1 Functional Specification

We consider functional specifications where the lexicon, or the set of words that can be used, corresponds to physical and computational components of a possible robot. In this paper, specifications are written in Structured English which has a deterministic and well defined grammar [14]. This grammar allows for the specification of safety constraints, goals and conditional expressions. The design flow described in this paper easily generalizes to functional specifications given in natural language as long as the natural language utterance can be represented formally using propositions that can be grounded, such as the work in [20]. In future work, we will leverage natural language processing tools to enrich the expressivity and ease of use of the design tools.

5.2 Grounding Propositions to Computational and Physical Components

The current process allows a user to map more than one proposition to a single component. This may be done by the user if different propositions are intended to dictate different behaviors for the same component. As an example, the user may ground the propositions `secureObject` and `releaseObject` to the same physical component `Gripper`, with the intent of having the gripper open when `releaseObject` is true and having the gripper close when `secureObject` is true. However, a problem may arise if the structured English specification allows both `secureObject` and `releaseObject` to be true simultaneously, resulting in contradictory commands to the same physical component, which will prevent the robot from achieving the desired behavior and may damage the robot.

The grounding interface addresses this issue by evaluating the grounded propositions and assessing which propositions (if any) have been grounded to the same component. The system then appends mutual exclusion clauses to the structured English specification so that the propositions may not both be true simultaneously, alerting the user to the change. This process is demonstrated with the *Fetch Robot* case study described in Sect. 6. When parsing the specification into complete robot designs, the compiler will automatically insert multiplexers into the data flow in order to ensure that the component receives the correct command.

5.3 Robot Behavior Guarantees

The controller generated for the robot is correct-by-construction, which means that provided the assumptions made by the user hold during controller execution and the user chose an appropriate grounding scheme, the robot will behave as expected.

6 Case Studies

6.1 Pick-and-Place Grasper

A sample robot made using this system is a robotic grasper. In this case a user desires a robot which, when prompted by the user, moves to a starting location and waits for an object; when an object is detected, it grasps it, moves to a target location, and notifies the user. This behavior can be written in a Structured English description as shown in Fig. 2, and the generated finite state machine can be examined via simulation as shown previously in Fig. 3. There are a variety of ways in which this can be grounded to generate a structural description, and a few such possibilities along with the one chosen here are depicted in Fig. 5; custom propositions represent internal state that

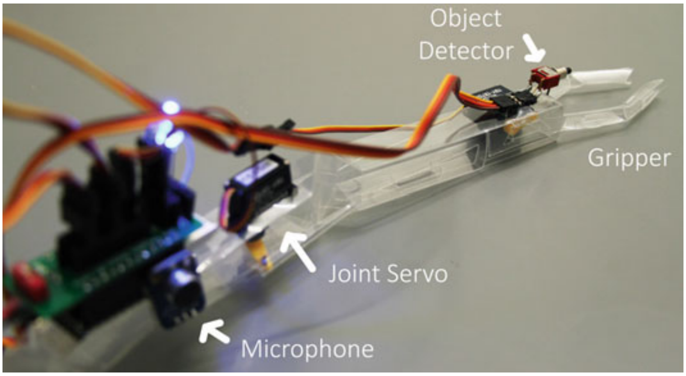


Fig. 6 A pick-and-place robotic grasper was designed using the presented system, starting with a desired behavior and ending with an inexpensive, rapidly manufactured, functional prototype

do not become grounded in robot components. During the process of choosing robot components from the library, various parameters such as arm length or gripper size can be set by the user according to their task's environment and restrictions.

Once the grounding is complete, the robot compiler generates a fold pattern along with electrical instructions and Arduino code. The resulting robot is shown in Fig. 6. After uploading the generated code, the arm demonstrates the desired behavior. When the user claps, the robot moves to the source location and waits for an object. It grasps the object upon detection, moves to the target location, releases the object, and indicates completion using a buzzer. Various metrics regarding the robot's performance as well as its design process are summarized in Table 1.

Specification 1 Linear Temporal Logic Specification for Example 1

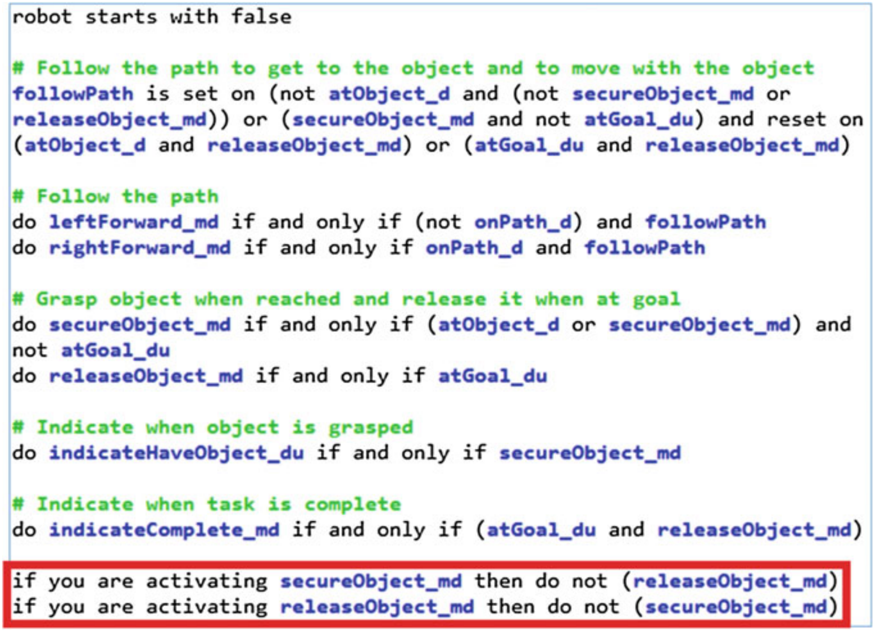
$$\begin{aligned}
&\neg\pi_{waitingForObject}\wedge\\
&\Box((\bigcirc\pi_{userSummons_d}\wedge\neg\pi_{pickUpObject_m})\rightarrow\bigcirc\pi_{waitingForObject})\wedge\\
&\Box(\pi_{pickUpObject_m}\rightarrow\neg\bigcirc\pi_{waitingForObject})\wedge\\
&\Box((\pi_{waitingForObject}\wedge\neg\pi_{pickUpObject_m})\rightarrow\bigcirc\pi_{waitingForObject})\wedge\\
&\Box((\neg\pi_{waitingForObject}\wedge\neg\bigcirc\pi_{userSummons_d})\rightarrow\neg\bigcirc\pi_{waitingForObject})\wedge\\
&\Box(\pi_{waitingForObject}\leftrightarrow\bigcirc\pi_{moveToSource_m})\wedge\\
&\Box((\pi_{seeObject_d}\wedge\pi_{moveToSource_m})\leftrightarrow\bigcirc\pi_{pickUpObject_m})\wedge\\
&\Box((\neg\pi_{pickUpObject_m}\wedge\neg\pi_{moveToSource_m})\leftrightarrow\bigcirc\pi_{indicateComplete_ud})
\end{aligned}$$

6.2 *Fetch Robot*

A second example robot is a mobile robot with an attached manipulator for retrieving an object placed along a path. The desired behavior is to follow a path until the object

Table 1 Performance of robotic grasper

Metric	Result
Approximate design time	30 min
Approximate fabrication time	30 min
Approximate cost	25 USD
Mass	49.4 g
Maximum actuated joint angle	±35 deg
Gripper strength (on 1.5 cm object)	100 mN
Maximum gripper opening	110 mm



```
robot starts with false

# Follow the path to get to the object and to move with the object
followPath is set on (not atObject_d and (not secureObject_md or
releaseObject_md)) or (secureObject_md and not atGoal_du) and reset on
(atObject_d and releaseObject_md) or (atGoal_du and releaseObject_md)

# Follow the path
do leftForward_md if and only if (not onPath_d) and followPath
do rightForward_md if and only if onPath_d and followPath

# Grasp object when reached and release it when at goal
do secureObject_md if and only if (atObject_d or secureObject_md) and
not atGoal_du
do releaseObject_md if and only if atGoal_du

# Indicate when object is grasped
do indicateHaveObject_du if and only if secureObject_md

# Indicate when task is complete
do indicateComplete_md if and only if (atGoal_du and releaseObject_md)

if you are activating secureObject_md then do not (releaseObject_md)
if you are activating releaseObject_md then do not (secureObject_md)
```

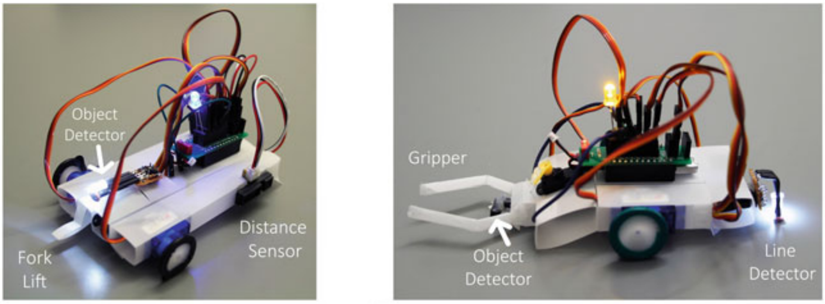
Fig. 7 The desired behavior of a path-following object fetcher can be defined using Structured English. The highlighted statements are necessary to enforce a mutual exclusion condition on propositions grounded to the same physical component, and are automatically generated and added to the behavioral specification

is reached, secure the object, continue following the path until the goal is reached, release the object, and indicate completion. This behavior can be written using the Structured English of LTLMoP as shown in Fig. 7.

To demonstrate the versatility and potential for rapid prototyping, two different sets of groundings were implemented. The chosen components are enumerated in Table 2, and the completed robots can be seen in Fig. 8. Some metrics regarding the performance as well as design of these robots are summarized in Table 3. In both cases

Table 2 Two separate sets of groundings implemented for a path following fetch robot

Functional proposition	Line follower	Wall follower
Move forward and left	Wheel 1	Wheel 1
Move forward and right	Wheel 2	Wheel 2
Detect path	Line detector	Distance sensor
Detect object	Touch sensor	Light sensor
Detect goal	UI toggle switch 1	Microphone
Secure object, release object	Gripper	Forklift
Indicate object secured	UI toggle switch 2	LED
Indicate complete	Wheel 1	Buzzer



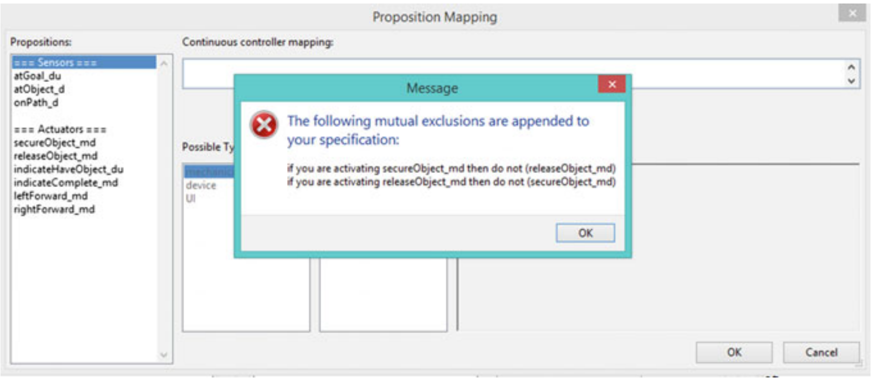
(a) This robot is a line follower and detects the object using a distance sensor.

(b) This robot is a wall follower and detects the object using a touch sensor.

Fig. 8 Two different robots made with the system which both achieve the desired task of following a path to retrieve an object

Table 3 Performance of path following fetch robots

Metric	Result	
	Line follower	Wall follower
Approximate design time	30	5 min
Approximate fabrication time	60	45 min
Approximate cost	30	45 USD
Mass	64.1	72.1 g
Speed	11.1	11.0 cm/sec
Maximum gripper opening	45	N/A mm



The following mutual exclusions are appended to your specification:

if you are activating secureObject_md then do not (releaseObject_md)
if you are activating releaseObject_md then do not (secureObject_md)

Fig. 9 The mapping process alerts the user when multiple propositions are mapped to the same component. In this case, the propositions `releaseObject_md` and `secureObject_md` have been mapped to the same component port

there are two propositions, `releaseObject_md` and `secureObject_md`, that are both mapped to the same port of the same component (`Gripper` or `Forklift` depending on the robot). In addition, the line follower instantiation maps both `leftForward_md` and `indicateComplete_md` to the same wheel servo in order to indicate completion with a "victory dance" behavior, spinning in a circle. LTLMoP detects these potential conflicts and notifies the user while automatically generating additional statements necessary to enforce a mutual exclusion on the relevant propositions. A sample notification is shown in Fig. 9. The generated statements are then automatically appended to the functional specification as shown in Fig. 7.

Once programmed with the generated code, both robot configurations performed the desired task. In addition, the generated code included calibration routines for the sensors when the robot first begins; it prompts the user to provide the minimum and maximum values for each sensor in turn and thereby determines a suitable threshold value for each sensor for converting the analog readings to boolean variables expected by the state machine. For example, the line follower will be placed over white and then over black, and the wall follower will be placed near the wall and then far from the wall. This also grants the user some runtime control over the robot behavior; for example, they can adjust how close the robot stays to the wall by adjusting the positions provided during calibration. Note that the line following robot design also includes UI elements; in this case, the user can use the provided Android app, which will automatically communicates with the generated robot via Bluetooth and display the appropriate user interface (in this case, two toggle switches).

7 Conclusion

In this paper, we present an approach to building and controlling a custom on-demand printable robot from a Structured English functional description, with an end-to-end integrated system implementing the above. This addresses a number of problems often faced by robot designers. Previously, despite the synthesis of a verified robot controller from a task specification, a mission may fail if existing robots are not suitable for the task. On the other hand, constructing custom robots to accomplish a desired behavior requires experience, expertise, tools, and resources.

The work presented here now allows users to start with a vision and follow system-generated recommendations to create a robot to execute that task. The user need not be experienced with robot creation or engineering principles, thus allowing even casual users access to these custom robots; advanced creaters still benefit from design automation. The correct-by-construction controller extends guarantees to the created robot, ensuring a successful mission provided that certain constraints are met. These guarantees coupled with online simulation simplify the design-build-test iteration loop and could easily allow for sophisticated design requirements such as building safety constraints into the robot. With the system demonstrated herein, functional and structural specifications can be matched to each other, allowing for the creation of on-demand robotic solutions for physical tasks.

This paper inspires a number of further research avenues addressing relaxing and avoiding such constraints, while expanding the autonomy provided by the compiler system. The system can be extended to integrate analog signals in a more automated manner for a richer behavioral design space. In addition, more complex functionality can be implemented by enabling a many-to-many mapping between propositions and components; though the compiler supports such a topology, determining mutual exclusion conditions is necessary to ensure provably correct constructions. Finally, a natural language input parser can allow greater flexibility in task specifications, potentially allowing more fine-grained recommendations of components for grounding through an analysis of the circumstances in which the proposition appears.

Acknowledgements This work was funded in part by NSF ExCAPE and grants #1240383 and #1138967 and NSF Graduate Research Fellowship 1122374, for which the authors express thanks.

References

1. Ayala, A.I.M, Andersson, S.B, Belta, C.: Probabilistic control from time-bounded temporal logic specifications in dynamic environments. In: Robotics and Automation (ICRA), pp. 4705–4710 (2012)
2. Bhatia, A., Kavraki, L.E., Vardi, M.Y.: Sampling-based motion planning with temporal goals. In: Robotics and Automation (ICRA), pp. 2689–2696 (2010)
3. Birkmeyer, P., Peterson, K., Fearing, R.S.: Dash: a dynamic 16g hexapedal robot. In: Intelligent Robots and Systems (IROS), pp. 2683–2689. IEEE (2009)

4. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012)
5. Demaine, E.D., Tachi, T.: Origamizer: a practical algorithm for folding any polyhedron (2009)
6. Fainekos, G.E., Kress-Gazit, H., Pappas, G.J.: Temporal logic motion planning for mobile robots. In: Robotics and Automation (ICRA), pp. 2020–2025 (2005)
7. Fikes, R.E., Nilsson, N.J.: Strips: a new approach to the application of theorem proving to problem solving. In: Proceedings of the 2nd IJCAI, London, UK, pp. 608–620 (1971)
8. Finucane, C., Jing, G., Kress-Gazit, H.: LTLMoP: experimenting with language, temporal Logic and robot control. In: IROS, pp. 1988–1993 (2010)
9. Hoover, A.M., Fearing, R.S.: Fast scale prototyping for folded millirobots. In: Robotics and Automation (ICRA), 2008, pp. 886–892. IEEE (2008)
10. Hornby, G., Lipson, H., Pollack, J.: Generative representations for the automated design of modular physical robots. IEEE Trans. Robot. Autom. **19**(4), 703–719 (2003)
11. Karaman, S., Frazzoli, E.: Complex mission optimization for multiple-UAVs using linear temporal logic. In: American Control Conference, Seattle, WA, pp. 2003–2009 (2008)
12. Kloetzer, M., Belta, C.: A fully automated framework for control of linear systems from temporal logic specifications. IEEE Trans. Autom. Control **53**(1), 287–297 (2008)
13. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Where's Waldo? Sensor-based temporal logic motion planning. In: Robotics and Automation (ICRA), pp. 3116–3121 (2007)
14. Kress-Gazit, H., Fainekos, G.E., Pappas, G.J.: Translating structured english to robot controllers. Adv. Robot. **22**(12), 1343–1359 (2008)
15. Lang, R.: Origami Design Secrets: Mathematical Methods for an Ancient Art. A K Peters/CRC Press, Boca Raton (2012)
16. Livingston, S.C., Prabhakar, P., Jose, A.B., Murray, R.M.: Patching task-level robot controllers based on a local mu-calculus formula. In: Robotics and Automation (ICRA), pp. 4588–4595 (2013)
17. McDermott, D., et al.: PDDL – the planning domain definition language – version 1.2. Technical report, Yale Center for Computational Vision and Control (1998)
18. Mehta, A.M., DelPreto, J., Shaya, B., Rus, D.: Cogeneration of mechanical, electrical, and software designs for printable robots from structural specifications. In: Intelligent Robots and Systems (IROS) (2014)
19. Onal, C., Wood, R., Rus, D.: An origami-inspired approach to worm robots. IEEE/ASME Trans. Mechatronics **18**(2), 430–438 (2013)
20. Raman, V., et al.: Sorry Dave, I'm afraid I can't do that: explaining unachievable robot tasks using natural language. In: Robotics: Science and Systems IX, Technische Universität Berlin, Berlin, Germany, 24 June–28 June 2013 (2013)
21. Romanishin, J., Gilpin, K., Rus, D.: M-blocks: momentum-driven, magnetic modular robots. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4288–4295 (2013)
22. Shimoyama, I., Miura, H., Suzuki, K., Ezura, Y.: Insect-like microrobots with external skeletons. IEEE Control Syst. **13**(1), 37–41 (1993)
23. Tama Software Ltd. Pepakura designer (2015). http://www.tamasoft.co.jp/pepakura-en/. Accessed 01 Apr 2015
24. Wolff, E.M., Topcu, U., Murray, R.M.: Optimization-based trajectory generation with linear temporal logic specifications. In: Robotics and Automation (ICRA), pp. 5319–5325 (2014)
25. Yim, M., Duff, D., Roufas, K.: PolyBot: a modular reconfigurable robot. In: Robotics and Automation (ICRA), vol. 1, pp. 514–520 (2000)
26. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robot. Autom. Mag. **14**(1), 43–52 (2007)

Cloud-Based Probabilistic Knowledge Services for Instruction Interpretation

Daniel Nyga and Michael Beetz

1 Introduction

In artificial intelligence, the problem of interpreting instructions is mostly approached by applying automated action planning methods in order to generate plans for tasks [21]. However, the plans generated by such systems are too abstract for competent execution by robots since they merely contain what is *given* but not what is *necessary*. A promising alternative is to generate plans for tasks from natural-language instructions that humans write for humans. Such instructions are available in abundance in the world-wide web at websites like wikihow.com, ehow.com, and many others. Instructions written for humans are more informative than automatically generated action plans because they often describe *how* actions have to be performed to bring about the desired effects, they talk about what can go wrong, and give additional hints. For robotic agents it makes sense to consider instruction understanding to be the computational problem of inferring how the agent could (successfully) perform the instruction. This problem formulation is substantially different to the problem of text understanding for question answering or machine translation. In those reasoning tasks, the vagueness and ambiguity of natural-language expressions can often be kept and translated into other languages. In contrast, robotic agents have to infer missing information pieces and disambiguate the meaning of the instruction in order to perform the instruction successfully.

Thus, if a robotic agent is tasked with the instruction *"neutralize 75ml of hydrochloric acid"*, for instance, the robot has to infer that neutralization requires to add a some amount of base substance to the hydrochloric acid. It also has to infer that this means that some amount of the base substance has to be transferred from the

D. Nyga (✉) · M. Beetz
Institute for Artificial Intelligence, University of Bremen, Bremen, Germany
e-mail: nyga@cs.uni-bremen.de

M. Beetz
e-mail: beetz@cs.uni-bremen.de

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_37

container which it is contained in into the container that holds the acid substance. Finally, because the amount is small and accurately specified the adding step should be performed through a pipetting action. As another example, consider the two instructions “fill the kettle with water” and “fill a cup with coffee.” Though the syntactic structure of the sentences as well as their semantics are identical, they fundamentally differ with respect to execution. Filling a kettle with water can be achieved by using the tap, whereas filling a mug with coffee implies a pouring motion from a coffee pot into a cup. In other words, understanding a natural-language instruction for robot execution requires *appropriate interpretation and completion*.

For the purpose of this paper we consider robotic agents that are equipped with a plan library that contains parameterizable plans for action verbs, which have to be refined according to a given instruction. In this case instruction understanding can be realized by retrieving the plan corresponding to the action required by the instruction and by constraining its parameterization according to the instruction.

To deal with the incomplete and ambiguous nature of natural-language instructions, we phrase the problem as a probabilistic reasoning problem, namely that of finding the most probable ‘executable’ refinement of the respective general plan given the natural-language instruction as evidence:

$$\arg\max_{\text{plan}} P\left(\text{intended(plan)} \,\middle|\, \begin{array}{l}\text{“neutralize 75ml}\\ \text{of hydrochloric acid”}\end{array}\right).$$

To perform this inference task we equip the robot with a joint probability distribution over the source, destination, the object acted on, the tool to be used, and other action roles for each action verb. To this end, we introduce the notion of *action cores*, which are conceptualizations of action verbs that represent formal specifications of actions and their parameters that are capable of interfacing the plans on a symbolic, linguistic level. The resulting probabilistic first-order knowledge base of action cores and their respective action roles is called *probabilistic action cores* (PRAC). PRACs can be used to perform disambiguation and completion of vague, underspecified natural-language (NL) sentences and thus are suitable for NL instruction interpretation.

An example of such an inference process is depicted in Fig. 1 which will also be the running example for our paper. We have equipped the robot with a plan library including plans for pouring and pipetting, among many others. The parameters of the plans for pouring and pipetting are the *theme* of the action, meaning the stuff to be transported from one place to another one, the *source* of the stuff, and the *destination* of the stuff. The problem of instruction interpretation for robot execution can now be formulated as the reasoning task of inferring the most probable plan (*action_core(a,c)*) and the most probable refinement of the formal plan parameters (source, destination, theme) given the natural language instruction *“neutralize 75ml of hydrochloric acid”*.

This is a very elegant and general formulation of instruction interpretation because by doing the inference task on a joint probability distribution over action instructions we can at the same time infer the plan that is most appropriate for performing the

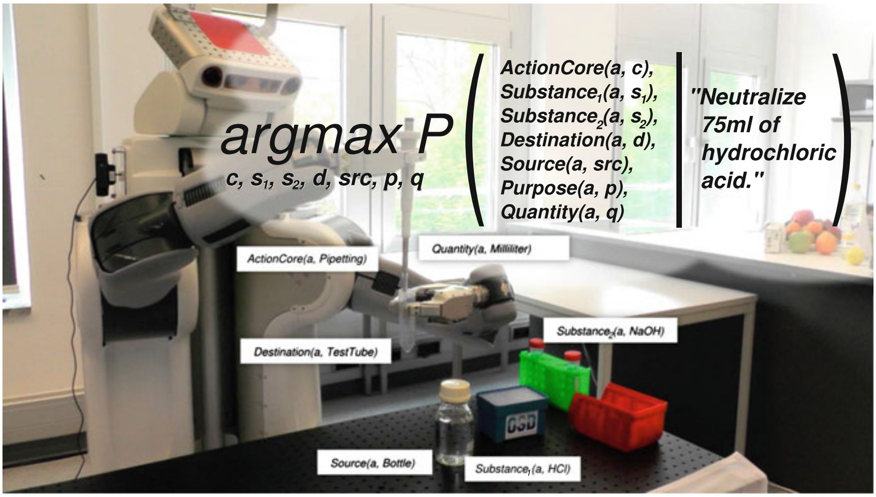


Fig. 1 Exemplary reasoning task in PRAC for interpreting a NL instruction

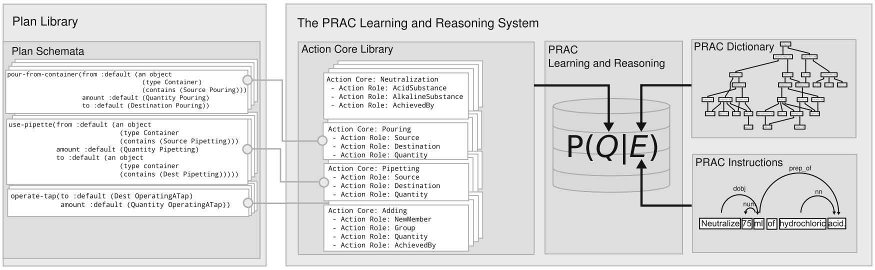
instruction, the refinement of the parameters of the plan schema on the basis of the information given in the instruction, and automatically fill in missing parameters by inferring their most probable value from the distribution.
The key contributions of this paper are the following:

- We formalize PRAC and the computational problem of inferring the most probable executable action instruction.
- We show how PRACs can be realized as Markov logic knowledge bases and learned from few examples.
- We show how the problem of inferring the most probable executable action instruction can be implemented to yield effective solutions as full size first-order probabilistic reasoning problems.

In the remainder of this paper we proceed as follows. We start with an introduction to the PRAC framework and detail its conceptual components. Subsequently, we give a formal definition of the reasoning tasks that PRAC addresses and describe our approach for tackling them. Then we discuss the state-of-the-art in instruction interpretation for robot commanding and conclude our work.

2 Conceptual Framework

The PRAC framework for translating natural-language instructions into abstractly parameterized robot action plans is depicted in Fig. 2. Its main components are the **PRAC plan library**, the **PRAC knowledge base**, and the **PRAC dictionary**. In a nutshell, the roles of these components are the following ones.



```
pour-from-container(from :default (an object
                                    (type Container)
                                    (contains (Source Pouring)))
                     amount :default (Quantity Pouring)
                     to :default (Destination Pouring))

use-pipette(from :default (an object
                           (type Container
                           (contains (Source Pipetting)))
            amount :default (Quantity Pipetting)
            to :default (an object
                           (type container
                           (contains (Dest Pipetting)))))

operate-tap(to :default (Dest OperatingATap)
             amount :default (Quantity OperatingATap))
```

```
Action Core: Neutralization
 - Action Role: AcidSubstance
 - Action Role: AlkalineSubstance
 - Action Role: AchievedBy

Action Core: Pouring
 - Action Role: Source
 - Action Role: Destination
 - Action Role: Quantity

Action Core: Pipetting
 - Action Role: Source
 - Action Role: Destination
 - Action Role: Quantity

Action Core: Adding
 - Action Role: NewMember
 - Action Role: Group
 - Action Role: Quantity
 - Action Role: AchievedBy
```

Fig. 2 Key concepts of the framework and their role in inferring the most probable executable instruction

predicate	meaning
$det(w_1, w_2)$	w_2 is the determiner of w_1
$dobj(w_1, w_2)$	w_1 is the direct object of w_2
$advmod(w_1, w_2)$	w_1 is an adverbal modifier of w_2
$prep\_with(w_1, w_2)$	w_1 and w_2 are connected by 'with'
$nn(w_1, w_2)$	w_1 and w_2 are a compound noun
$has\_pos(w, p)$	w has the part-of-speech p

concept	definition
cup.n.01	a small open container usually used for drinking; usually has a handle
cup.n.02	cupful, the quantity a cup will hold
cup.n.08	a large metal vessel with two handles that is awarded as a trophy to the winner of a competition

Fig. 3 *Left* Selection of logical predicates representing syntactic structure of words in a sentence. A comprehensive list can be found in [9] *Right* Selection of different meanings of the word 'cup' obtained from WordNet

The **PRAC dictionary** provides all possible meanings of all the words that can occur in a NL instruction to be executed by a robot. The meanings are concepts in an ontological knowledge base defined in the dictionary *WordNet* [12], which comprises more than 117,000 concepts. For example, the possible meanings of 'cup' in the PRAC dictionary include a specialization of a physical object and, more specifically, a container object, an amount specification, and a trophy (see also Fig. 3).

The **PRAC knowledge base** contains a collection of action verb-specific knowledge bases, called **action cores**, that represent how possible action instructions for a given action verb can be constructed on a conceptual level. For example, we can formalize a pouring action on a concept level in terms of conjunctions of logical assertions over the predicates *action_core(a, Pouring)*, *theme(a, t)*, *source(a, s)*, *destination(a, d)*, etc. The assertion *theme(a, t)* states that the theme of action *a* is of the type *t*, i.e. the entity which is poured. The parameters *t*, *s* and *d* are concepts in the PRAC dictionary.

Action-specific knowledge bases are then trained with a set of instructions stated in first-order logic in order to learn a joint probability distribution over predicate instantiations, which is induced by the given set of instructions. These distributions are called the **probabilistic action core (PRAC)**. The learned distribution represents correlations between the concept restrictions of the parameters in instructions with respect to an action verb. For example, the PRAC of Pouring could entail that if the Theme of a pouring action is the concept wine then it is more likely that the source for the pouring action will be an instance of the concept bottle and the destination an instance of the concept glass. Conversely, if the Theme is of the concept water, then the source is more likely to be a tap.

Finally, the **PRAC plan library** contains action specific plans. PRAC plans are equipped with plan signatures following the 'design-by-contract' principle: The plan signature specifies the formal parameters of the plan, the concept restrictions for each parameter and how the respective plan parameter can be computed from the PRAC knowledge base. For example, the signature of the plan for a pouring action looks as follows:

```
pour-from-container(from :default (an object
                                      (type container.n.01)
                                      (contains (Pouring Theme))
                    amount :default (Pouring Quantity)
                    to :default (an object
                                      (type container.n.01)
                                      (contains (Pouring Goal))))
```

The plan schema specifies that the from parameter has to be a specialization of the concept container.n.01 and that the from parameter can be retrieved from the PRAC knowledge base of Pouring by retrieving the value of the role Theme of Pouring. Likewise, the amount parameter can be obtained by querying for the Quantity predicate.

It is required that all formal parameters of the plan are linked to roles in the respective action core. By providing a plan signature, the designer of the plan guarantees that for all plan refinements that satisfy the concept restrictions of the individual parameters, executing the plan generates meaningful behavior. 'Meaningful' here means that the plan generates behavior that makes sense but is not required to succeed. For a pouring action, for instance, the plan tells the robot to grasp the source container, to hold it above the destination and to tilt it. However, the execution of the parameterized plan hazards failures caused by inappropriate motor control or inaccurate perception, such as spilling the liquid because the container is held too high, off center, or the pouring angle is too steep. This requires that all parameters needed to call sub-plans are computed by the plan and no call to a sub-plan contains undefined parameters, which would cause the control system to crash.

The plans themselves are considered as black boxes in PRAC reasoning. Plan execution systems that can handle such qualitative, symbolic constraints on parameters include RAP [13] and PRS [14]. If deeper reasoning about the ramifications of actions is necessary, the CRAM [6, 19] executive provides reasoning methods that translate qualitative constraints into PROLOG queries that use sampling and backtracking to find parameter instantiations satisfying these constraints. Kinds of such parameters include e.g. action effects, visibility, reachability and the like.

Using the components of the PRAC system introduced above, the computational process for computing the most probable executable instruction operates as follows: In a first step, a given natural-language instruction ι is translated by a natural-language parser into a logical representation of the instruction's syntactic structure I, which we call a PRAC instruction. The PRACinstruction I is then interpreted by inferring the meaning and semantic role of the individual syntactic structures and missing information pieces using the PRAC *dictionary* and the action core itself. This interpretation process results in the *most probable executable instruction* of ι. In the remainder of this section we will describe in more detail the concepts and components that learning and reasoning about action cores is built upon.

2.1 PRAC *Instructions*

A PRAC instruction I is a set of assertions about the grammatical relations referring to the constituents of a natural-language instruction and their syntactic structure. These grammatical relations are represented by predicates including the small selection listed in Fig. 3. They are obtained for any sentence in natural language by a parser like the Stanford parser [8]. Using these predicates, a natural-language instruction such as $\iota =$ "neutralize the hydrochloric acid with sodium hydroxide", for example, is transformed into the logical assertions I,

$$\begin{array}{ll} \mathit{dobj}(\mathit{neutralize\text{-}1}, \mathit{acid\text{-}4}) & \mathit{has\_pos}(\mathit{neutralize\text{-}1}, \mathit{VB}) \\ \mathrm{det}(\mathit{acid\text{-}4}, \mathit{the\text{-}2}) & \mathit{has\_pos}(\mathit{acid\text{-}4}, \mathit{NN}) \\ \mathit{nn}(\mathit{acid\text{-}4}, \mathit{hydrochloric\text{-}3}) & \mathit{has\_pos}(\mathit{hydroxide\text{-}7}, \mathit{NN}) \\ \mathit{nn}(\mathit{hydroxide\text{-}7}, \mathit{sodium\text{-}5}) & \mathit{has\_pos}(\mathit{sodium\text{-}6}, \mathit{NN}) \\ \mathit{prep\_with}(\mathit{neutralize\text{-}1}, \mathit{hydroxide\text{-}7}), & \end{array} \tag{1}$$

which we denote by $\mathscr{I}(\iota)$. These syntactic dependencies indicate that the second word 'the' depends on the fourth word 'acid' as a determiner, 'hydrochloric' and 'acid' represent a compound noun, which forms the direct object of the word 'neutralize', which is connected to the word hydroxide via the preposition 'with'. The syntactic structure of the instruction thus forms a relational database that serves as evidence in a probabilistic relational model. For a more detailed and exhaustive description of the syntactic dependencies, we refer to [9].

2.2 PRAC *Dictionary*

The PRAC dictionary is a set of logical assertions that assign meaning (word senses) to words. It is filled with word senses ('synsets') from the online dictionary WordNet[1] (see Fig. 5). The word senses are organized in a taxonomy given by a directed acyclic graph, which we denote by the relation $\sqsubseteq$, i.e. $c_1 \sqsubseteq c_2$ denotes that the concept c_1 is a specialization of concept c_2.

[1] All concept names refer to concept names provided by the NLTK toolbox (http://www.nltk.org).

In the PRAC dictionary, a word w is assigned a particular meaning m by means of a set of logical assertions $has\_sense(w, m)$ and $is\_a(m, c)$ $\forall c\ m \sqsubseteq c$, where $has\_sense(w, m)$ states that the word w has the sense m in the WordNet dictionary and the is_a predicate is the transitive closure of m in $\sqsubseteq$. The PRAC dictionary also provides a function μ: $W \times P \mapsto \mathscr{P}(\top)$, which returns the set of all possible meanings of a word given its part of speech, where W denotes the set of all words, P is the set of all parts of speech, $\top$ is the set of all concepts in $\sqsubseteq$ and $\mathscr{P}(\cdot)$ denotes the power set.

Words can have multiple meanings causing ambiguity in NL instructions. Consider, for example, the terms ‘cup’ and ‘milk’ and their meaning in the two instructions “fill a cup with milk” and “add a cup of milk.” In the former case, ‘cup’ refers to a drinking mug, a physical object that can hold milk. In the latter case, it rather refers to a measurement unit specifying the amount of milk to be added. Though this semantic difference may seem subtle, correctly distinguishing between word meanings is crucial for successfully performing the actions. Thus, in finding an appropriate interpretation of an instruction, selecting the most appropriate word meanings is a necessity.

2.3 PRAC *Knowledge Base*

The PRAC knowledge base is the central component of the PRAC system. It contains a library of data structures, which we call **action cores**. An **action core** is the conceptualization of an action which constitutes an abstract event type and assigns an action role to each entity that is needed in order to successfully perform the respective action.

More formally, an action core AC is defined as a tuple $\langle A, R \rangle$, where A is the globally unique name of the action core and $R = \{r_{A_i}\}_{i=1}^{n_A}$ is an indexed set of its associated action roles. For an interpretation x of a PRAC instruction I, $interpretation(x, I)$, the following holds:

$$action\_core(x, A) \rightarrow \exists c_1, \ldots, c_{n_A} \bigwedge_{i=1}^{n_A} r_{A_i}(x, c_i),\ c_i \in \top \tag{2}$$

The right side of the implication in (2) ensures that every instantiation of an action core must have a complete assignment of its action roles to concepts in $\top$, otherwise it is not an instance of the action core. However, being able to assign all roles of an action core does not imply that it must have an instance in x. Equation 2 defines the space $\mathscr{X}$ of possible interpretations of an instruction. A graphical representation of one particular interpretation of the instruction “neutralize 75 ml of hydrochloric acid” is shown in Fig. 4: There are instances of the three action cores Neutralization, Adding and Pipetting with their respective roles assigned a concept. The set of action

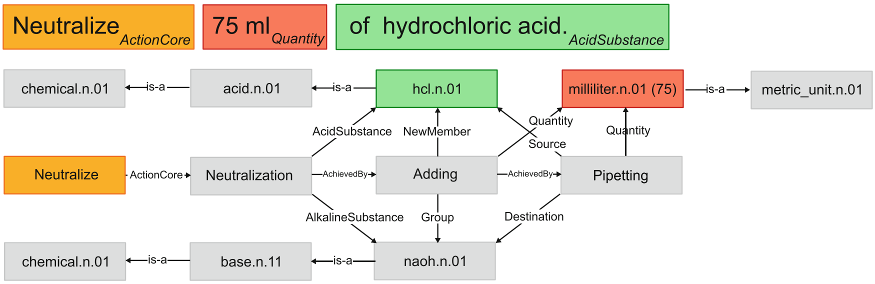


Fig. 4 Exemplary action instantiation for the instruction "Neutralize 75 ml of hydrochloric acid." Taxonomy paths (*is-a*) are truncated for better readability

cores and our definition of an interpretation can thus be regarded as a template for constructing a graphical model of interpretations like the one in Fig. 4.

An example of the action core Pouring and its action roles was already given above. In that context, the action core has a direct mapping to a plan schema in the PRAC plan library and its action roles Source, Destination and Theme interface the formal parameters of the plan schema. As another example, consider the action core Neutralization, representing the process of causing a chemical substance to take a neutral pH-value by combining it with some other substance. For the chemical reaction itself, there must always be two components reacting, an acid and a base. The corresponding action core Neutralization thus is attached two action roles, AcidSubstance and AlkalineSubstance. It is important to note that within PRAC, the domains of action roles and their corresponding parameter slots in the plan schemata are given by the set $\top$ of all concepts from the ontological knowledge base in the PRAC dictionary. This ensures that all symbols have the same semantics across the different components of PRAC, the syntactic representation in the PRAC instructions, the semantic action representation of action cores as well as the plan schemata.

There is an action core for every verb in the PRAC dictionary that represents a meaningful action. However, there are action cores that do not have a direct correspondence to a plan schema because they do not represent actions that are directly executable but are subject to further reasoning. Neutralization is an example of such an action core: It is not an executable action as such, but rather describes a chemical process that is triggered by Adding one substance to the other. Adding itself is an action core representing the process of making a new member part of an existing group. It has three action roles, namely the Group, the NewMember and the Quantity. The Adding action core, however, also represents a process that can be achieved in very different ways depending on the context and the objects involved. For example, "add one liter of water" could be achieved by using the tap or pouring from one container to the other, whereas "add one milliliter of water" should be performed by using a pipette. Conversely, "add a pinch of salt" can be done by using a salt cellar. Such an action core A that does not have a direct mapping to an executable plan schema has attached a designated action role AchievedBy(A, A'), which is assigned another action core A' that represents the most likely refinement of the action represented by A.

The goal in natural-language instruction understanding is now to find the *most probable interpretation under the instruction given as evidence*. Therefore, the PRAC knowledge base has a conditional probability distribution over all action cores and their action roles, conditioned on the PRAC dictionary and the PRAC instructions, as depicted in Fig. 2,

$$P\left(\begin{array}{c} action\_core(x, A) \rightarrow \\ \exists c_1, \ldots, c_{n_A} \bigwedge_{i=1}^{n_A} r_{A_i}(x, c_i) \end{array} \middle| \top, \mathscr{I} \right). \tag{3}$$

We call (3) the **probabilistic action core** (PRAC). The probabilistic action core is a first-order probabilistic knowledge base about actions and their parameterizations that is used to disambiguate, interpret, complete and refine NL instructions.

2.4 Examples

The probabilistic action core can be used to resolve ambiguity and to complete an instruction to the most plausible action specification, based on what is given by the instruction. In the following, we will illustrate the usage of the PRAC distribution by means of three simple exemplary queries.[2]

Action role assignment. PRAC can be queried for the most likely assignment of roles for a given set of objects with respect to a particular action core. Consider an instruction, such as "add 3 drops of sodium hydroxide." There are two objects o_1 and o_2 in the instruction given by the concepts *naoh.n.01* and *drop.n.02* in the PRAC dictionary. In context of the Adding action core, one can solve for the most probable assignment of the action roles attached to Adding, i.e.

$$\underset{o_1', o_2', o_3' \in \{o_1, o_2, \bot\}}{\arg\max} \; P\left(\begin{array}{l} Quantity(o_1'), \\ NewMember(o_2'), \\ Group(o_3') \end{array} \middle| \begin{array}{l} is\_a(o_1, drop.n.02), \\ is\_a(o_2, naoh.n.01) \end{array} \right) = \left\{\begin{array}{c} o_1' = o_1, \\ o_2' = o_2, \\ o_3' = \bot \end{array}\right\},$$

where $\bot$ denotes the *null* assignment. In this example, the Quantity role has been assigned the object *drop.n.01*, the NewMember role the object *naoh.n.01* and the Group could not have been assigned any of the objects mentioned in the instruction. Note that this arg max solution is not a proper interpretation of the instruction in the notion from above, because there is no Group specified.

[2] We are using a slightly modified notation, which technically does not precisely fit the previous formulations. We think this simplified notation better supports the understanding of reasoning considered in this paper.

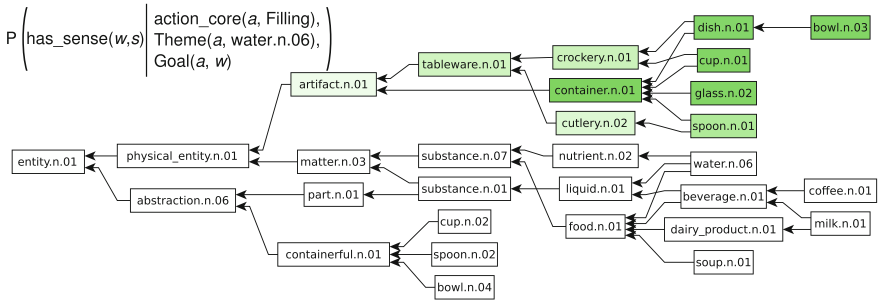


Fig. 5 Conditional distribution over an excerpt of the PRAC taxonomy structure for containers, substances and measuring units for an entity w taking the Goal role of the Filling action core

Action role completion. In order to fill missing role assignments such as the Group in the previous example, one can solve for a different arg max query. Consider the instruction "neutralize the hydrochloric acid" and suppose we have already assigned the object *hcl.n.01* the role AcidSubstance of the Neutralization action core. According to its definition, there must be the role AlkalineSubstance assigned to some concept, which is not given in the instruction. In order to infer its role assignment, we introduce a Skolem constant s that hypothetically fills the missing role slot of AlkalineSubstance. Since the taxonomy relation of the PRAC dictionary is included in the PRAC distribution, one can query for the most probable type of s:

$$\arg\max_{c \in \top} P\left(is\_a(s, c) \;\middle|\; \begin{array}{l} is\_a(hcl, hcl.n.01), \\ AcidSubstance(hcl), \\ AlkalineSubstance(s) \end{array} \right) = naoh.n.01,$$

which means that sodium hydroxide (NaOH) is the most probable alkaline counterpart for the Neutralization of hydrochloric acid (HCl).

Joint distributions over taxonomies. One of the key features of PRAC is the ability to perform reasoning about unmodeled concepts, i.e. concepts that have not been seen during learning. This enables (1) a compact representation of knowledge, (2) efficient transfer of the learnt knowledge to new situations and (3) filling missing information pieces in underdetermined action specifications. Figure 5 shows an example of a conditional distribution over concepts in the PRAC taxonomy for potential Goals of a filling action: From all concepts, specializations of containers gain highest probability, which reasonably reflects our intuitions about a typical filling action. Since PRAC maintains joint distributions over the action roles and the concepts in the PRAC dictionary, we can compute any conditional distribution given any evidence, which enables context-sensitive completion of actions like in the previous example. Such out-of-domain inference tasks are implemented using the FUZZY-MLN reasoning framework [22].

3 The PRAC Learning and Reasoning System

In this section, we describe in more detail how reasoning is implemented in the PRAC framework. We first depict the basic ideas of learning and inference in PRAC, address the issues of learning and present the processing pipeline for performing inference about interpretations and completions of natural-language instructions. There are two key paradigms in the PRAC reasoning system.

Learning by generalization. Humans are capable of learning rapidly and flexibly how to use different words in different situations by only having seen very few examples. They have available an efficient apparatus for generalization, which allows them to abstract away from a very small set of specific instantiations to more generic patterns of everyday situations that we often encounter in their 'typical' form. Consider the example of a 'filling' action. From hearing just a few specific instances of that action verb, e.g., "fill a pot with water" and "fill a cup with milk", humans are capable of generalizing to a stereotyped pattern like "fill a container with a liquid." This kind of generalization is both powerful and efficient since, on the one hand, it enables compact representation of knowledge and on the other hand, it allows to treat new, unseen examples in a meaningful way.

Inference by specialization. Reasoning about new, unseen situations is done by selecting one or more generic patterns that best fit the new situation and by adapting them to reality as necessary in order to come up with an instantiated representation which is as specific and unambiguous as possible. In the example from above, an instruction like "fill a glass with juice," for instance, is matched against the generic 'filling' action and is adapted accordingly by inspecting the conceptual subsumption of the terms 'juice', which corresponds to the liquid being poured and 'glass', which constitutes the goal container of the filling action.

PRAC implements these two paradigms in a coherent probabilistic framework, which automatically finds abstractions of common situations as illustrated in the above examples by exploiting the semantic similarities of concepts in the taxonomy graph. These abstractions reasonably reflect human intuitions of how specific terms are to be used in certain situations. These principles of abstraction and generalization from examples also constitute cornerstones of human cognition [3, 17, 26]. As an implementational framework, we use Markov logic networks (MLN) [23] to encode the knowledge about action cores, their action roles, the PRAC dictionary and the PRAC instructions, which is a powerful knowledge representation formalism that combines first-order logic with probability theory.

3.1 Reasoning

The probabilistic first-order knowledge base in (3) for solving inference problems of the form $\arg\max_Q P(Q \mid E)$ has an enormous size. It contains at least the cross product of all possible word meanings squared and roles where the set of the possible

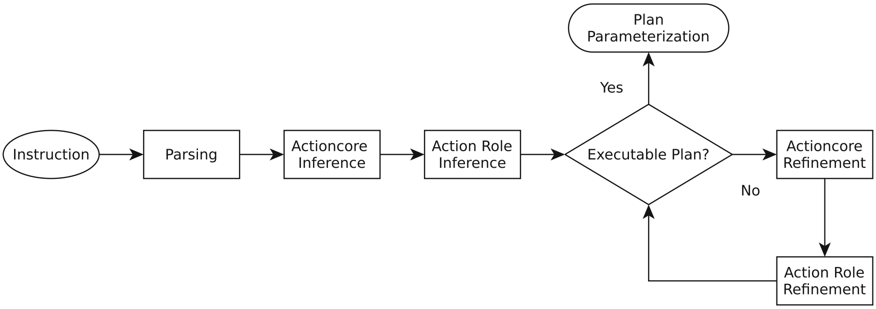


Fig. 6 Flow diagram of the PRAC reasoning pipeline with the steps (1) NL parsing (2) action core identification (3) action role identification (4) checking for an executable plan schema attached to an action core and (5) action core refinement

word meanings include all possible meanings of the words that occur in the training data plus the number of their superconcepts in the taxonomy. To make the reasoning problem feasible we decompose it into three weakly connected subproblems and generate the probabilistic knowledge bases for each substep independently to keep the knowledge bases as small as possible: (1) inferring the relevant PRAC, (2) disambiguation and role assignment and (3) inferring missing information pieces and refinements of action cores and their associated roles.

Reasoning in PRAC about the most probable interpretation of a natural-language instruction ι is implemented by the following multi-step composition of database transformations by means of probabilistic relational inference:

$$\underset{R_{A_{missing}}}{\arg\max}\, P\left(R_{A_{missing}} \middle| \arg\max_{R_{A_{given}}} P\left(R_{A_{given}} \middle| \arg\max_A P\left(A \middle| I\right)\right)\right),$$

where $I = \mathscr{I}(\iota)$ is a PRAC instruction representing the syntactic structure of the NL instruction, A is the action core referred to by the instruction, $R_{A_{given}}$ are the action role assignments of A given in I, and $R_{A_{missing}}$ are the action roles of A which do not have a correspondence in I. Figure 6 depicts the reasoning pipeline of PRAC, which we will describe in the following in more detail.

1. Parsing: The first step in PRAC reasoning is to analyze the syntactic structure of the instruction at hand, which yields a PRAC instruction database I according to (1) containing syntactic relations and the part of speech for each word.
2. Given the words and their part of speech, the possible word meanings are obtained from the PRAC dictionary and the actioncore is identified that is ‘activated’ by I with highest probability:

$$\widehat{A} = \arg\max P\left(action\_core(a, ac) \middle| I, \mu(I)\right)$$

3. Given $\widehat{A}$ and its associated roles $\widehat{R}$ from the actioncore library, PRAC performs simultaneous word sense disambiguation and action role assignment taking into account the concept taxonomy of the PRAC dictionary:

$$\widehat{A}' = \arg\max P\left(has\_sense(\cdot,\cdot), \widehat{R} \,\middle|\, \widehat{A} \cup I \cup \sqsubseteq\right)$$

4. Subsequently, having assigned the action roles for the identified actioncore, PRAC checks if there is a plan schema in the plan library attached to the actioncore, which can be parameterized with the inferred roles. If so, the schema is instantiated with its parameters and sent to the plan executive.
5. If there is no plan schema attached, PRAC incrementally computes refinements of $\widehat{A}'$ by alternately solving for the most probable actioncore ac' ac can be AchievedBy and its action roles:

$$\widehat{A}'' = \arg\max_{a'} P\left(achievedBy(a, a') \,\middle|\, \widehat{A}' \cup \sqsubseteq\right)$$

A visualization of an exemplary inference process in PRAC and the execution of an instantiated plan schema can be found in the video accompanying this paper.[3]

4 Related Work

In recent years, much work has been done in order to make knowledge sources available to robots, which are indented for human use [24, 27], and to generate robot plans out of natural-language instructions [10, 16, 20, 25, 27]. Dzifcak et al. [10] use a combinatorial categorial grammar for deriving a goal formulation in temporal logics in order to find an action sequence that achieves this goal. Matuszek et al. [16] use statistical machine translation techniques to match natural-language navigation directives against a formal path description language. Others [24, 25] use probabilistic models to derive plans to be executed by a robot. Misra et al. [18] take into account the context of the environment for grounding objects in an instruction to objects in the environment. They solve the ambiguity in instructions using an energy function corresponding to a conditional random field. What these approaches have in common is that they do not take into account that natural-language instructions typically are severely underspecified, ambiguous and often not directly executable. They make what is commonly referred to as the *closed-world assumption* postulating that all knowledge about the world is given and complete. Additionally, most approaches to teach robots by means of natural language are designed to capture and execute what is specified by an instruction using 'shallow' mappings to robot control, but they are not intended to accumulate more semantic action knowledge that can be recalled in and adapted to different situations. Artzi et al. [2] and Kim et al. [15] learn probabilistic context-free grammars for robot navigation tasks. Their approach is inspired

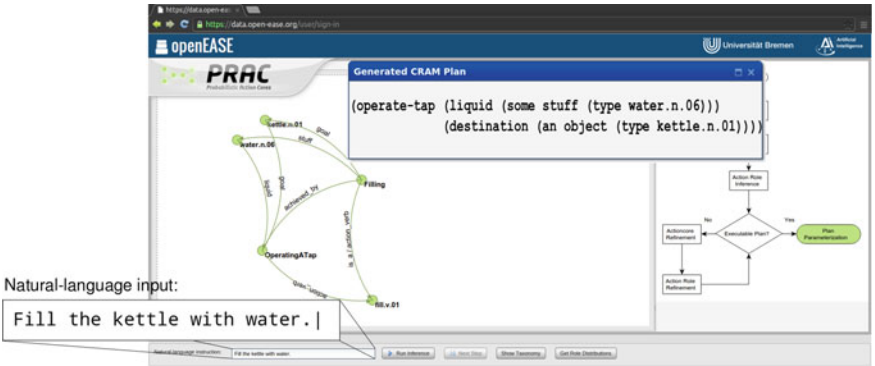
[3] https://youtu.be/iA6s7IGqubs.

from a more linguistic point of view, where such grammars are are typically induced from large corpora of text consisting of sequences of navigational directives.

We take a different approach accounting for the variational complexity and richness of human-scale manipulation tasks with everyday objects. In PRAC, the result of a linguistic analysis of an instruction is taken only as evidence in a probabilistic first-order knowledge base which allows us on the one hand to include any syntactic characteristics of a sentence as evidence in a query, and on the other hand it enables tight integration with the robot's belief state, high-level knowledge base, executive and perception system, which can provide comprehensive context information, such as the objects perceived in a scene, for instance. In addition, PRAC makes use of a rich taxonomy of concepts which allows to transfer the lernt knowledge to new, unseen concepts. Our work is not about finding action sequences given a particular goal, but about *how* to perform complex everyday activities in presence of partial and incomplete information. It is inspired by and closely related to Minsky's [17] frame representation and partially adapted from the FrameNet [4] specifications of action verbs but extended and adapted for including knowledge necessary for robot action execution. Our ultimate goal is a complete robotic agent that is able to successfully *perform* complex manipulation tasks formulated in NL (cmp. [1, 5]). Commonly used linguistic corpora of instructions do not account for the behavior that the analyzed instruction produces. This makes a large-scale corpus-based evaluation of PRAC nearly impossible. Our future work therefore focuses on thoroughly evaluating PRAC with respect to these points, such as the executability of an action or whether or not it produces the desired effects and avoids undesired effects by executing the generated robot plans in a simulated environment (cmp. [11]).

5 Conclusions

In this paper we have shown how interpretation of ambiguous and underdetermined natural-language instructions can be formulated as the problem of computing the most probable complete and unique instruction in an action specific knowledge base called probabilistic action core (PRAC). Within the PRAC framework, the most probable complete and unique instruction enables robots to find the most appropriate plan and with the most general refinement of the formal plan parameters given the NL instruction. To perform this inference the PRAC framework learns a joint probability distribution over all possible ways in which instructions for a given action verb can be formulated. Our PRAC framework provides an attractive alternative to other instruction interpretation approaches, in particular for the interpretation of complex manipulation tasks. One important advantage is that PRACs are not limited to inferring which sequences of actions should be executed but also *how* the individual actions are to be executed. A second advantage is that the use of taxonomic reasoning in the PRAC inference results in the inference of the most general concept refinements of the plan parameters. This generates least commitment calls of plans that keep maximal flexibility at execution time and avoids the necessarity of grounding symbolic names



Generated CRAM Plan

```
(operate-tap (liquid (some stuff (type water.n.06)))
             (destination (an object (type kettle.n.01))))
```

Natural-language input:

```
Fill the kettle with water.|
```

Fig. 7 The browser-based webinterface to PRAC on the openEASE cloud robotics platform

that are generated in the interpretation process (symbol grounding problem). Our current implementation comprises a set of 12 PRACs and plan schemata from two application domains, the household/cooking domain and the domain of conducting chemical experiments, which we are continuously extending. We implemented PRAC as an open-source software framework which is accessible as a web service on the cloud robotics platform openEASE [7] (http://www.open-ease.org), shown in Fig. 7.

Acknowledgements This work is supported by the EU FP7 Projects *RoboHow* (grant number 288533) and *ACAT* (grant number 600578).

References

1. Anderson, J.: Constraint-directed improvisation for everyday activities. Ph.D. thesis (1995)
2. Artzi, Y., Zettlemoyer, L.: Weakly supervised learning of semantic parsers for mapping instructions to actions. Trans. Assoc. Comput. Linguist. **1**(1), 49–62 (2013)
3. Bailey, D.: When push comes to shove: A computational model of the role of motor control in the acquisition of action verbs. Ph.D. thesis, University of California (1997)
4. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics -. ACL '98, vol. 1, pp. 86–90. Association for Computational Linguistics, Stroudsburg, PA, USA (1998)
5. Barker, R.: Ecological Psychology: Concepts and Methods for Studying the Environment of Human Behavior. Stanford University Press, Stanford (1968)
6. Beetz, M., Jain, D., Mösenlechner, L., Tenorth, M., Kunze, L., Blodow, N., Pangercic, D.: Cognition-enabled autonomous robot control for the realization of home chore task intelligence. Proc. IEEE **100**(8), 2454–2471 (2012)
7. Beetz, M., Tenorth, M., Winkler, J.: Open-EASE – a knowledge processing service for robots and robotics/ai researchers. In: IEEE International Conference on Robotics and Automation (ICRA), Seattle, Washington, USA (2015). (Finalist for the Best Cognitive Robotics Paper Award)
8. De Marneffe, M., MacCartney, B., Manning, C.: Generating typed dependency parses from phrase structure parses. Proc. LREC **6**, 449–454 (2006)

9. de Marneffe, M.-C., Manning, C.D.: The stanford typed dependencies representation. In: Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, CrossParser '08, pp. 1–8. Stroudsburg, PA, USA (2008) (Association for Computational Linguistics)
10. Dzifcak, J., Scheutz, M., Baral, C., Schermerhorn, P.: What to do and how to do it: translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In: IEEE International Conference on Robotics and Automation, 2009. ICRA'09, pp. 4163–4168. IEEE (2009)
11. Feldman, J., Narayanan, S.: Embodied meaning in a neural theory of language. Brain Lang. **89**(2), 385–392 (2004) (Language and MotorIntegration)
12. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
13. Firby, J.: Adaptive execution in complex dynamic Worlds. Technical report 672, Yale University, Department of Computer Science (1989)
14. Georgeff, M., Ingrand, F.: Decision making in an embedded reasing system. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 972–978. Detroit, MI (1989)
15. Kim, J., Mooney, R.J.: Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 433–444. Association for Computational Linguistics (2012)
16. Matuszek, C., Fox, D., Koscher, K.: Following directions using statistical machine translation. In: Proceeding of the 5th ACM/IEEE International Conference on Human-robot Interaction, pp. 251–258. ACM (2010)
17. Minsky, M.: A framework for representing knowledge. Technical Report Memo 306, MIT-AI Laboratory (1974)
18. Misra, D.K., Sung, J., Lee, K., Saxena, A.: Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In: Proceedings of Robotics Science and Systems, Berkeley, USA (2014)
19. Mösenlechner, L., Beetz, M.: Parameterizing actions to have the appropriate effects. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, USA (2011). Accessed 25–30 Sept 2011
20. Neo, E., Sakaguchi, T., Yokoi, K.: A natural language instruction system for humanoid robots integrating situated speech recognition, visual recognition and on-line whole-body motion generation. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2008. AIM 2008, pp. 1176–1182. IEEE (2008)
21. Nyga, D., Beetz, M.: Everything robots always wanted to know about housework (but were afraid to ask). In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal (2012). Accessed 7–12 Oct 2012
22. Nyga, D., Beetz, M.: Reasoning about unmodelled concepts – incorporating class taxonomies in probabilistic relational models (2015). http://arxiv.org/abs/1504.05411
23. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. **62**(1–2), 107–136 (2006)
24. Ryu, J., Jung, Y., Kim, K., Myaeng, S.: Automatic extraction of human activity knowledge from method-describing web articles. In: Proceedings of the 1st Workshop on Automated Knowledge Base Construction, p. 16 (2010)
25. Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., Roy, N.: Understanding natural language commands for robotic navigation and mobile manipulation. In: Proceedings of the National Conference on Artificial Intelligence (AAAI) (2011)
26. Tenenbaum, J.B., Kemp, C., Griffiths, T.L., Goodman, N.D.: How to grow a mind: statistics, structure, and abstraction. Science **331**(6022), 1279–1285 (2011)
27. Tenorth,M., Nyga, D., Beetz, M.: Understanding and executing instructions for everyday manipulation tasks from the world wide web. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1486–1491. Anchorage, AK, USA (2010). Accessed 3–8 May 2010

Identifiability Analysis of Planar Rigid-Body Frictional Contact

Nima Fazeli, Russ Tedrake and Alberto Rodriguez

1 Introduction

Autonomous manipulation in an uncertain environment requires an autonomous understanding of contact. A priori models of objects and their environment are routinely deficient or defective: In some cases it is not cost-effective to build accurate models; others the complex and ever-transforming nature of nature renders it impossible. This understanding of contact is often implicit in the design of a manipulator. By carefully choosing materials and geometries we can passively deal with uncertainty. However, when we want to monitor or actively control the execution of a manipulation task, an explicit understanding of the algebra between motions, forces, and inertias at contact is principal.

We are inspired by human's unconscious but effective ability to make sense of contact to understand its environment. It only takes us a small push to a cup of coffee to estimate how full it is, and a quick glance to a bouncing ball to gauge its stiffness. This work builds on the conviction that, similarly, robots can harness known laws of physical interaction to make sense of observed motions and/or forces, and as a result gain a better understanding of their environment and themselves.

In particular, in this initial study we explore the identifiability of inertial parameters and contact forces associated with planar frictional contact interactions. We exploit the linear complementarity formulation (LCP) of contact resolution [1, 18] to relate inertial parameters, contact forces, and observed motions. Section 3 reviews the

N. Fazeli (✉) · A. Rodriguez
Mechanical Engineering Department, MIT, Cambridge, MA, USA
e-mail: nfazeli@mit.edu

A. Rodriguez
e-mail: albertor@mit.edu

R. Tedrake
Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA
e-mail: russt@csail.mit.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_38

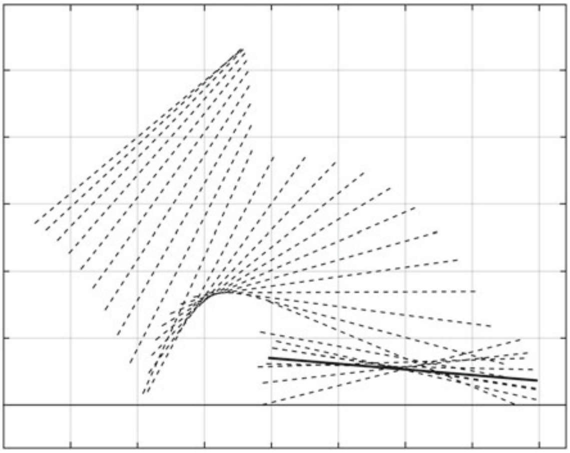


Fig. 1 Stewart and Trinkle [18] used the example of a falling rod to introduce a time-stepping complementarity scheme for contact resolution that has become one of the standard techniques for simulating frictional contact. In this paper we look at the same formulation and similar examples from the perspective of identification. Is the trajectory of the rod indicative enough of the dynamic system that governs its motion?

structure of an LCP problem and describes the mathematical framework necessary to outline the identifiability analysis.

The specific system we consider is a single planar rigid body undergoing impact after a period of free fall, as in Fig. 1. What can we say about an object from observing its motions and/or forces? The falling trajectory is a simple ballistic motion, which can be fitted to the dynamics of free fall. The key challenge, and focus of this paper, is in finding a formulation suitable for system identification, that can handle the complexity of unknown and spurious reaction forces due to frictional contact. Such a formulation might yield a systematic approach for a broader set of contact interactions including sliding, pushing or grasping.

Our main contribution is a systematic analysis of the question of the identifiability of the mass, the moment of inertia, and contact forces from kinematic observations of frictional contact interactions. Section 4 details the analysis both for cases when contacts stick or slip, as well as when known external forces are applied during contact.

In this paper we use a batch approach to system identification, where we extract the best possible inertial parameters and contact forces that explain a series of observations. A potential benefit over more traditional calibration methods for parameter fitting, is that equivalent on-line techniques are well understood and readily available. Section 5 evaluates the validity of the approach analysis with simulated and real experiments with a planar block and a planar ellipse falling on a flat ground, which are captured with a high-speed camera.

2 Background and Motivation

System identification studies the problem of fitting a model (i.e., inertial parameters) to a series of inputs (i.e., forces and torques) and responses (i.e., displacements/velocities/accelerations) of a dynamic system. The most basic idea behind system identification is that, although the response of a dynamic system tends to be complex, the governing dynamics are often linear in a set of observable parameters. For example, while $\sum \mathbf{f} = m \cdot \mathbf{a}$ can lead to complex trajectories, forces and accelerations are still linearly related by m. This allows closed-form least-squares formulations for the estimation of those parameters.

System ID is the process of identifying what parameters are instrumental and what observations are informative, and then make estimates of the parameters from measured data. This idea has been applied in robotics to the identification of serial and parallel link manipulators [7, 8], and to identify inertial parameters sufficient for control purposes [17].

In this paper we show that system identification has the potential to provide a formal approach to observe rigid-body contact interactions, which, in turn, opens with a wide set of possible applications: Contact-aware state estimation (Erdmann [6], Atkeson [3], Koval et al. [9], Zhang and Trinkle [22], Trinkle [20], Yu et al. [21]); Contact-aware planning and control (Lynch and Mason [10], Platt and Kaelbling [12], Posa et al. [13], Chavan Dafle and Rodriguez [5]); Fault detection or task monitoring (Rodriguez et al. [15], Salawu [16]).

One of the main assumptions in our approach is the selection of a time-stepping Linear Complementarity Problem (LCP) scheme for the resolution of forces and accelerations during frictional contact. Why LCP? Brogliato et al. [4] identifies 3 classes of methods for rigid body simulation:

i. *Penalty methods* model interaction as a reaction force proportional to the amount of interpenetration. Although easier to solve, they lack in realism.
ii. *Event-driven methods* rely on a listing, resolution, and selection of all possible contact/impact events. They typically require some knowledge of contact time. Müller and Pöschel [11] showed that they can lead to exceedingly high velocities in situations with multiple contacts.
iii. *Time-stepping methods* integrate the equations of motion during a finite time interval. Should a contact (or multiple) be detected during the interval, the algorithm resolves the collisions and continues to integrate the equations of motion.

The time-stepping approach, in conjunction with the velocity-impulse resolution of contact, which results in a Complementarity Problem (CP), has been advocated by Stewart and Trinkle [18] and Anitescu and Potra [2] among others, and has been shown to be robust to phenomena such as Painleve's problem [19], and always to have a solution, with linear approximations of the friction cone and for a positive definite mass matrix.

3 Complementarity Problems for Collision Resolution

The standard approach to resolve motion, is the following simple iterative scheme:

$$\text{Current state} \longrightarrow \begin{array}{c}\text{Compute resultant}\\ \text{of applied forces}\end{array} \longrightarrow \begin{array}{c}\text{Integrate forward}\\ \text{to next state}\end{array} \tag{1}$$

One of the core difficulties in dealing with contact is that it breaks that basic scheme. The motion of the system depends on the resultant of applied forces, but at the same time these applied forces (friction and contact normal) depend on the motion of the system. As a consequence, both contact forces and resulting motions must be determined (searched for) simultaneously, instead of sequentially as in (1).

This section reviews the complementarity formulation for contact resolution which solves simultaneously for contact forces and velocities. For further details we refer the reader to Stewart [19].

A Linear Complementarity Problem

A general (i.e. nonlinear) complementarity problem is defined as:

$$\text{Find: } z \qquad \text{s.t.} \qquad 0 \leq g(z), \qquad 0 \leq z, \qquad 0 = z \cdot g(z) \tag{2}$$

A linear complementarity problem is formed when g is of the form $g(z) = Mz + q$. The benefit of a complementarity formulation is that it allows us to write the equations of motion of a dynamic system with contact defined as unilateral constraints (and that mathematicians have devised solvers for that kind of problem). Force balance looks like:

$$M(q)\frac{dv}{dt} = J_n^T c_n + D(q)c_t + k(q, v) - \nabla V(q) + F_{ext}(t) \tag{3}$$

where:

- $J_n^T c_n(t)$ and $D(q)c_t(t)$ represent the normal and tangential contact forces;
- $J_n = \nabla\phi_n(q)$, is the gradient of a function $\phi_n(q)$ that determines the boundary between no contact ($\phi_n(q) > 0$) and penetration ($\phi_n(q) < 0$);
- $D(q)$ is a set of column vectors that linearly span the tangent space at contact, and the product $D(q)c_t$ represents the actual frictional force at a contact;
- $V(q)$ represents conservative forces, such as gravity;
- $k(q, v)$ represents the centrifugal and Coriolis velocity components;
- and F_{ext} represents all external non-conservative forces excluding contact.

The motion at contact, and the tangential contact forces due to friction are related by the principle of maximal dissipation [1] which states that during contact the selection of both has to maximize dissipation, i.e., generally that friction tends to oppose motion:

$$\min_{c_t} (v^+)^T D(q)c_t \qquad \text{such that: } \psi(c_t) \leq \mu c_n \tag{4}$$

Contact resolution then will need to search for the components of c_t such that the frictional force $D(q)c_t$ opposes velocity, from within a valid domain of frictional forces $\psi(c_t) \leq \mu c_n$. If we follow Coulomb's law, all possible frictional forces must lie inside a cone $FC(q) = \{D(q)c_t \text{ s.t. } ||c_t||^2 \leq \mu c_n\}$, where now $\psi(c_t) = ||c_t||^2$. In general, ψ can be shown to be convex, coercive and positively homogeneous which implies that $D(q)c_t \in c_n FC(q)$ is equivalent to $\psi(c_t) \leq \mu c_n$.

We can convert (4) into a CP constraint by noting that the inequality constraint can be incorporated in the minimization by using a Lagrange multiplier $h(c_t, \lambda) = (v^+)^T D(q)c_t - \lambda(\mu c_n - \psi(c_t))$ where now the condition for minimum is:

$$\frac{\partial h}{\partial c_t} = \mu D(q)^T v^+ + \lambda \frac{\partial \psi(c_t)}{\partial c_t} = 0 \tag{5}$$

Furthermore we can write:

$$\begin{aligned} &0 \in \mu D(q)^T v^+ + \lambda \frac{\partial \psi(c_t)}{\partial \psi} \\ &0 \leq \lambda, \qquad 0 \leq \mu c_n - \psi(c_t), \qquad 0 = \lambda(\mu c_n - \psi(c_t)) \end{aligned} \tag{6}$$

which completes the CP formulation for contact resolution:

$$\begin{aligned} \frac{dq}{dt} = vM(q)\frac{dv}{dt} &= J_n^T c_n + D(q)c_t - \nabla V(q) + k(q, v) + F_{ext} \\ \text{s.t. } &0 \leq c_n \perp 0 \leq \phi_n, \;\; 0 \in \mu D(q)^T v^+ + \lambda \frac{\partial \psi(c_t)}{\partial \psi} \\ \text{s.t. } &0 \leq \lambda \perp 0 \leq \mu c_n - \psi(c_t), \;\; 0 = J_n v^+ \text{ if } \phi_n(q) = 0 \end{aligned} \tag{7}$$

Note that the formulation so far is nonlinear with respect to the friction surface constraint (ψ). For the sake of resolution, it is common linearize the CP by approximating ψ as a polyhedral convex cone. We construct it by using a finer discretization of the tangent plane at contact with a set of vectors $\{J_n + \mu d_i(q) | i = 1, 2, ..., m\}$ that positively span it. It is convenient to chose these vectors equiangular with respect to each other, and are paired as $d_i = -d_j$, which we stack in a new matrix $\tilde{D}(q)$. Now we can express the friction force as $\tilde{D}(q)\tilde{c_t}$ where $\tilde{c_t} \geq 0$ and $\Sigma \tilde{c}_{ti} \leq \mu c_n$. Note that we will drop the tilde from the notation but in the rest of the paper, we will assume that the polyhedral approximation holds for all further analysis.

Finally, the CP formulation for contact resolution is:

$$
\begin{aligned}
q_{k+1} &= q_k + h \cdot v_{k+1} \\
M(q_k)(v_{k+1} - v_k) &= c_n J_n^T(q_k) + D(q_k)c_t - h \cdot k(q_k, v_k) - h \cdot \nabla V(q_k) + h \cdot F_{\text{ext}} \\
0 \le c_n &\perp 0 \le J_n(q_k)(v_{k+1} + \varepsilon v_k) \\
0 \le c_t &\perp 0 \le \lambda e + D(q_k)^T v_{k+1} \\
0 \le \lambda &\perp 0 \le \mu c_n - e^T c_t
\end{aligned} \tag{8}
$$

A Time-Stepping Approach

Next, to be able to simulate the evolution of the dynamics we convert the formulation to its time-stepping equivalent, for which we integrate the contact forces over a time step. The resulting equations of motion and constraints follow Stewart [19]. We integrate the expressions in (8) forward in time using an Euler scheme. Distance to contact is captured by a measure of the closest distance between boundary of two rigid bodies:

$$
\Phi = \left[\phi_n \ \phi_t \right]^T \tag{9}
$$

where $\phi_n > 0$ signals free space, $\phi_n = 0$ contact, and $\phi_n < 0$ interpenetration. Figure 2 depicts an arbitrary planar rigid body with active contact constraints:

$$
\Phi = \begin{bmatrix} \phi_n \\ \phi_t \end{bmatrix} = \begin{bmatrix} y - l(\beta)\cos(\beta - \theta) \\ x - l(\beta)\sin(\beta - \theta) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{10}
$$

where angle β parameterizes the object boundary and localizes the contact point. It is a function of θ such that $0 \le \beta(\theta) < 2\pi$. We compute the Jacobian of the constraint for the equation of motion (3) as:

$$
\frac{\partial \Phi}{\partial q} = \begin{bmatrix} J_n \\ J_t \end{bmatrix} = \begin{bmatrix} 0 \ 1 \ J_y(\theta) \\ 1 \ 0 \ J_x(\theta) \end{bmatrix} \tag{11}
$$

where the rows of $\frac{\partial \Phi}{\partial q}$ can be seen as contact forces, and:

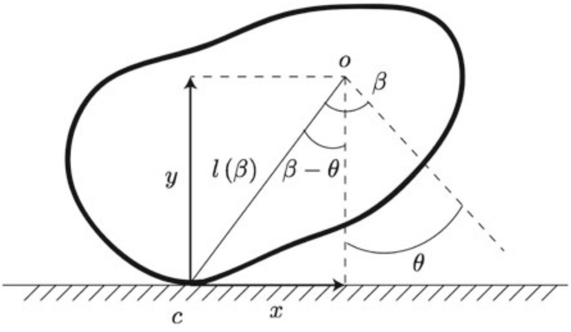


Fig. 2 2D rigid body in contact

$$
\begin{aligned}
J_y &= -\frac{\partial \beta}{\partial \theta}\left(\frac{\partial l}{\partial \beta}\cos(\beta-\theta) - l\sin(\beta-\theta)\right) - l\sin(\beta-\theta)\\
J_x &= -\frac{\partial \beta}{\partial \theta}\left(\frac{\partial l}{\partial \beta}\sin(\beta-\theta) + l\cos(\beta-\theta)\right) + l\cos(\beta-\theta)
\end{aligned}
\tag{12}
$$

In general, this formulation gives us the contact mode post-impact given pre-impact kinematic measurements, and externally applied forces. On the other hand, if the contact-mode is known, then there is no need to solve the CP. Impulses during impact and velocities/positions post-impact can be solved strictly as functions of these states and external influences pre-impact. We use this fact in Sects. 4.1 and 4.2 to find close form relations between forces, accelerations, and inertial parameters.

4 Identifiability Analysis

In this section we study the identifiability of the inertial parameters (mass and second moment of inertia) and contact forces of a rigid body as it comes into contact with a rigid and fixed flat surface. We assume that the positions, orientations and velocities (linear and angular) of the object are given and the external forces acting on the object are known. The second moment of inertia of the object is expressed with respect to a reference frame attached to the center of mass. Given that kinematic measurements of the trajectory are available, we can derive the direction of friction by invoking the principle of maximum dissipation as outlined in Sect. 3. We will denote the direction by J_t. We consider sticking and sliding contact modes separately in the following subsections.

4.1 Sliding Contact Mode

During sliding the complementarity constraints from (8) become:

$$0 < c_n, \qquad 0 < c_t, \qquad 0 < \lambda \tag{13}$$

The first inequality derives from the fact that at contact the distance constraint is equal to zero so its dual must be greater than zero. To understand the second and third inequalities we point out that since we have sliding contact then v_{k+1} must have at least one component that is not perpendicular to the tangent plane spanned by D and so $\max -d_i v_{k+1} \leq \lambda$ where d_i are the columns of D. This directly implies that $0 < \lambda$ and the second inequality comes from the fact that since $0 < \lambda$ then $\mu c_n = e^T c_t$ which implies that $0 < c_t$.

Utilizing the inequalities from (13) we revisit (8) and write it with c_n, c_t and λ as variables:

$$\begin{bmatrix} J_n^T M^{-1} J_n & J_n^T M^{-1} J_t & 0 \\ J_t^T M^{-1} J_n & J_t^T M^{-1} J_t & 1 \\ \mu & -1 & 0 \end{bmatrix} \begin{bmatrix} c_n \\ c_t \\ \lambda \end{bmatrix} = - \begin{bmatrix} J_n^T b \\ J_t^T b \\ 0 \end{bmatrix} \tag{14}$$

where:

$$b = v_k + hM^{-1}(-\nabla V - k(q, v) + F_{ext}) \tag{15}$$

$$M^{-1} = \text{diag}\{\frac{1}{m}, \frac{1}{m}, \frac{1}{I}\}, \qquad \nabla V = \text{diag}\{0, mg, 0\}, \qquad F_{ext} = \begin{bmatrix} F_x & F_y & \tau \end{bmatrix}^T$$

$$\tilde{\omega} = \begin{bmatrix} 0 & -\dot{\theta} & 0 \\ \dot{\theta} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad k = \begin{bmatrix} 0 \\ 0 \\ \tilde{\omega} R I_0 R^T \omega \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and we can solve for c_n and c_t to arrive at:

$$c_n = m \frac{\dot{y}_k + \theta_k J_{y,k} - hg + \frac{h}{m}(F_y + h\frac{m}{I}\tau)}{1 + \left(J_{y,k}^2 + \mu J_{x,k} J_{y,k}\right) \frac{m}{I}} = \frac{c_t}{\mu} \tag{16}$$

At this stage we have explicitly solved for the contact forces as functions of the inertial properties, geometry of contact and pre-contact kinematic measurements. To perform identifiability analysis we require an equation that is strictly a function of kinematic measurements and inertial properties so we replace the values derived for c_n and c_t into the equation of motion (3):

$$\begin{bmatrix} \dot{x}_{k+1} - \dot{x}_k \\ \dot{y}_{k+1} - \dot{y}_k + hg \\ \dot{\theta}_{k+1} - \dot{\theta}_k \end{bmatrix} = \frac{\dot{y}_k + \dot{\theta}_k J_{y,k} - hg + \frac{h}{m}(F_y + h\frac{m}{I}\tau)}{1 + \left(J_{y,k}^2 + \mu J_{x,k} J_{y,k}\right) \frac{m}{I}} \begin{bmatrix} \mu \\ 1 \\ \frac{m}{I}\left(J_{y,k} + \mu J_{x,k}\right) \end{bmatrix} + h \begin{bmatrix} \frac{F_x}{m} \\ \frac{F_y}{m} \\ \frac{\tau}{I} \end{bmatrix} \tag{17}$$

We can further manipulate (17) to yield the linear mapping:

$$Y = \Psi \Theta \tag{18}$$

where:

$$Y = \begin{bmatrix} \begin{bmatrix} \dot{x}_{k+1} - \dot{x}_k \\ \dot{y}_{k+1} - \dot{y}_k + hg \end{bmatrix} - \begin{bmatrix} \mu \\ 1 \end{bmatrix} \left(\dot{y}_k + \dot{\theta}_k J_y - hg\right) \\ \dot{\theta}_{k+1} - \dot{\theta}_k \end{bmatrix}$$

$$\psi_1 = \begin{bmatrix} hF_y\mu + hF_x \\ hF_y \\ 0 \end{bmatrix}, \qquad \psi_2 = \begin{bmatrix} hJ_y\tau \begin{bmatrix} \mu \\ 1 \end{bmatrix} + \left(J_{y,k}^2 + \mu J_{x,k} J_{y,k}\right) h \begin{bmatrix} F_x \\ F_y \end{bmatrix} \\ \left(J_y + \mu J_x\right) hF_y + h\tau \end{bmatrix} \tag{19}$$

$$\psi_3 = \left(J_y + \mu J_x\right) \begin{bmatrix} -J_y \begin{bmatrix} \dot{x}_{k+1} - \dot{x}_k \\ \dot{y}_{k+1} - \dot{y}_k + hg \end{bmatrix} \\ \dot{y}_k + \dot{\theta}_k J_y - hg - \left(\dot{\theta}_{k+1} - \dot{\theta}_k\right) J_y \end{bmatrix}, \qquad \psi_4 = \begin{bmatrix} 0 \\ 0 \\ \left(J_y - \mu J_x\right) h J_y \tau \end{bmatrix}$$

$$\Psi = \left[\, \psi_1 \; \psi_2 \; \psi_3 \; \psi_4 \,\right]$$

$$\Theta = \left[\, \tfrac{1}{m} \; \tfrac{1}{I} \; \tfrac{m}{I} \; \tfrac{m}{I^2} \,\right]^T$$

Equation (18) is linear in the inertial parameters and assuming that Θ is the unknown vector of inertial parameters then, given samples of Y and Ψ, we can set up a constrained least squares estimation problem to determine m and I. Furthermore with the mass and moment of inertia identified we can infer the contact forces from (16), and conclude that the uniquely identifiable set is $\{m, I, c_n, c_t\}$. Assuming external-forces (excluding gravity) are set to zero then $\psi_1 = 0$, $\psi_2 = 0$ and $\psi_4 = 0$ which means that the only identifiable parameter is m/I. Replacing the value of m/I into (16) we can find c_t/m and c_n/m therefore the set of parameters that we can estimate uniquely in this case is $\{m/I, c_t/m, c_n/m\}$.

4.2 Sticking Contact Mode

During sticking contact the complementarity constraints in (8) become:

$$0 < c_n, \qquad 0 < c_t, \qquad \lambda = 0 \tag{20}$$

The first inequality is direct consequence of being in contact, as in the previous case. Since we are in sticking contact then a tangential force must exist to prevent sliding, therefore c_t must be greater than zero. A less intuitive justification can be garnered by considering the complementarity constraints of (8) and noting that since the velocity post contact will not have a component within the tangential plane of contact then $D^T v_{k+1} = 0$. In this scenario either $e^T c_t = \mu c_n$ which implies that frictional force is at its boundary and the analysis will follow as in Sect. 4.1 or that $e^T c_t \leq \mu c_n$ which implies that the frictional force lies inside the friction cone. Note that simply requiring that $D^T v_{k+1} = 0$ will result in $\lambda = 0$. Where the frictional force lies inside the boundary of its maximum, the LCP formulation from (8) simplifies to:

$$\begin{bmatrix} J_n^T M^{-1} J_n & J_n^T M^{-1} J_t \\ J_t^T M^{-1} J_n & J_t^T M^{-1} J_t \end{bmatrix} \begin{bmatrix} c_n \\ c_t \end{bmatrix} + \begin{bmatrix} J_n^T b \\ J_t^T b \end{bmatrix} = 0 \tag{21}$$

We solve for c_n and c_t and replace expressions from (15):

$$\begin{bmatrix} c_n \\ c_t \end{bmatrix} = \frac{-m}{1 + \frac{m}{I}\left(J_y^2 + J_x^2\right)} \begin{bmatrix} 1 + \frac{m}{I}J_x^2 & -\frac{m}{I}J_x J_y \\ -\frac{m}{I}J_x J_y & 1 + \frac{m}{I}J_y^2 \end{bmatrix} \begin{bmatrix} \dot{y}_k - hg + \dot{\theta}_k J_y + \frac{h}{m}(F_y + \frac{m}{I}\tau J_y) \\ \dot{x}_k + \dot{\theta}_k J_x + \frac{h}{m}(F_x + \frac{m}{I}\tau J_x) \end{bmatrix} \tag{22}$$

Replacing the expressions for contact forces into the equation of motion (3), and rearranging we have:

$$Y = \begin{bmatrix} \dot{x}_{k+1} + \dot{\theta}_k J_x \\ \dot{y}_{k+1} + \dot{\theta}_k J_y \\ \dot{\theta}_{k+1} - \dot{\theta}_k \end{bmatrix}, \quad \psi_1 = \begin{bmatrix} -\left(J_y^2 + J_x^2\right)\dot{x}_{k+1} + J_x^2 \dot{x}_k + J_x J_y \left(\dot{y}_k - hg\right) \\ -\left(J_y^2 + J_x^2\right)\dot{y}_{k+1} + J_x J_y \dot{x}_k + J_y^2 \left(\dot{y}_k - hg\right) \\ -\left(J_y^2 + J_x^2\right)\dot{\theta}_{k+1} - J_x \dot{x}_k - J_y \left(\dot{y}_k - hg\right) \end{bmatrix}$$

$$\psi_2 = h \begin{bmatrix} -F_x J_y^2 + F_y J_x J_y - J_x \tau \\ F_x J_x J_y - J_x^2 F_y - J_y \tau \\ -F_x J_x - F_y J_y + \tau \end{bmatrix}, \quad \psi_3 = \begin{bmatrix} 0 \\ 0 \\ -(J_x^2 + J_y^2)h\tau \end{bmatrix}, \quad \psi_4 = h \begin{bmatrix} F_x \\ F_y \\ 0 \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \frac{m}{I} & \frac{1}{I} & \frac{m}{I^2} & \frac{1}{m} \end{bmatrix}^T, \quad \Psi = \begin{bmatrix} \psi_1 & \psi_2 & \psi_3 & \psi_4 \end{bmatrix} \tag{23}$$

By careful inspection we conclude that the inertial parameters m and I are uniquely identifiable in the presence of known external forces (excluding gravity) and we can infer contact forces from (22). Furthermore, if external forces are non-existent then $\psi_2 = \psi_3 = \psi_4 = 0$ and in this case only the ratio of mass to second moment of inertia and the ratio of contact forces to the mass of the object are identifiable. These results are consistent with sliding contact mode.

5 Examples: Block and Ellipse

In this section we apply the identifiability analysis in Sects. 4.1 and 4.2 to two examples: 2-D block and 2-D ellipse undergoing free-fall and colliding with a fix flat surface perpendicular to the direction gravity. We use two data sets validating the derivations: (i) simulated data using a numerical implementation of time-stepping LCP and (ii) experimental data recorded with a high speed camera. In both scenarios we measure the position, orientation and velocities of the bodies as they interact with the environment and attempt to identify the inertial parameters.

5.1 Identification Formulation

We consider a rigid body as in Fig. 2 at contact, denoting pre-impact time step as k and post impact time step as $k + 1$:

$$\begin{bmatrix} \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} \dot{x}_{k} \\ \dot{y}_{k} \\ \dot{\theta}_{k} \end{bmatrix} + h \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \frac{c_n}{m} \begin{bmatrix} 0 \\ 1 \\ \frac{m}{I} J_y \end{bmatrix} + \frac{c_t}{m} \begin{bmatrix} 1 \\ 0 \\ \frac{m}{I} J_x \end{bmatrix} \tag{24}$$

Since we are interested in the inertial parameters and assume that no known external forces (except gravity) act on the object during contact, from our analysis we can identify m/I. With this in mind a simple manipulation of (24) yields the least squares optimization problem:

$$\min_x ||Y - Ax||_2 \tag{25}$$

where: $Y = \dot{\theta}_{k+1} - \dot{\theta}_k, \quad A = (\dot{x}_{k+1} - \dot{x}_k)J_x - (\dot{y}_{k+1} - \dot{y}_k + hg)J_y, \quad x = \dfrac{m}{I}$

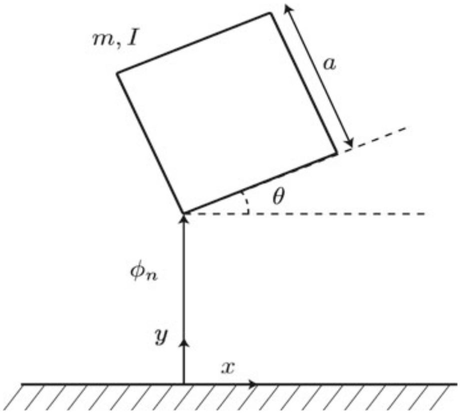
The solution of (25) yields an estimate of m/I. So far we have been agnostic to the shape of the rigid body, we note that the shape of the object will affect the choice of the contact Jacobian and will be case specific, in the subsequent sections we derive expressions for the contact Jacobians of the block and ellipse.

5.1.1 2D Block

The block (Fig. 3) is modeled as a 2D square with length a, angle of rotation θ, center of mass at location (x, y), mass m and second moment of inertia I. We define the distance of the lowest vertex to the ground as a function of the configuration of the block:

$$\phi_n = \min \left(\begin{bmatrix} f_1(q) = y - \frac{a}{\sqrt{2}} \cos(\pi/4 - \theta) \\ f_2(q) = y - \frac{a}{\sqrt{2}} \cos(\pi/4 + \theta) \\ f_3(q) = y + \frac{a}{\sqrt{2}} \cos(\pi/4 - \theta) \\ f_4(q) = y + \frac{a}{\sqrt{2}} \cos(\pi/4 + \theta) \end{bmatrix} \right) \tag{26}$$

Fig. 3 2D block in free-fall



where $f_i(q)$ denotes the vertical distance of vertex i to the ground as a function of the height of the center of mass of the block y and it orientation θ.

We derive the contact Jacobians by differentiating (12). We assume that the lowest vertex is $f_1(q)$ and that we observe a single point contact, i.e., curvature does not play a role, which means that $\frac{\partial \beta}{\partial \theta} = 0$. Invoking (12) with this constraint and noting that $\beta = \pi/4$ and $l = a/\sqrt{2}$ for vertex 1, the normal and tangential Jacobians are:

$$J_n = \left[0 \ 1 \ -\tfrac{a}{\sqrt{2}} \sin(\pi/4 - \theta) \right]^T, \qquad J_t = \left[1 \ 0 \ \tfrac{a}{\sqrt{2}} \cos(\pi/4 - \theta) \right]^T \tag{27}$$

Note that for a simple case such as the square, we could arrive at the same expression for the contact Jacobians simply by taking the partials of the contact constraints with respect to the configurations:

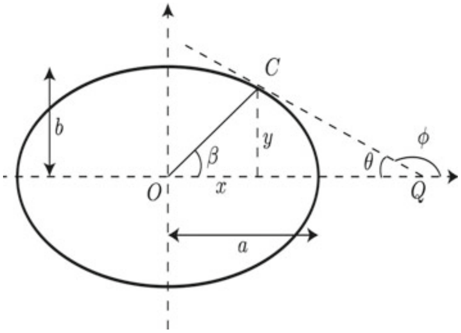
$$h(q) = \begin{bmatrix} y - \frac{a}{\sqrt{2}} \cos(\pi/4 - \theta) \\ x - \frac{a}{\sqrt{2}} \sin(\pi/4 - \theta) \end{bmatrix}, \qquad J_n = \frac{\partial h_1(q)}{\partial q}, \qquad J_t = \frac{\partial h_2(q)}{\partial q} \tag{28}$$

5.1.2 2D Ellipse

The ellipse, depicted in Fig. 4, is geometrically interesting because of the relative curvature between the surfaces in contact. Unlike the block, contact dynamics are very sensitive to orientation. Small perturbations of the object's orientation when in contact produces small changes to the contact location. We parametrize the perimeter curve of the ellipse by angle β, denote the major and minor radii of the ellipse with a and b, and refer to the contact point by C. The angle θ denotes the orientation of the ellipse with respect to line QC which for our purpose is the surface of contact. To compute the Jacobian from (12) we use geometry to relate β and θ:

$$\tan(\pi - \theta) = -\frac{b}{a} \cot \beta \xrightarrow{\frac{\partial}{\partial \theta}} \frac{\partial \beta}{\partial \theta} = -\frac{a}{b} \frac{1 + \tan^2(\pi - \theta)}{1 + \cot^2(\beta)} \tag{29}$$

Fig. 4 2D ellipse schematic



We can write the distance of any point on the perimeter of an ellipse from its center as:

$$l(\beta) = \frac{ab}{\sqrt{b^2\cos^2\beta + a^2\sin^2\beta}} \xrightarrow{\frac{\partial l}{\partial \theta}} \frac{\partial l}{\partial \theta} = \frac{ab(b^2 - a^2)\sin 2\beta}{2\sqrt{(b^2\cos^2\beta + a^2\sin^2\beta)^3}} \frac{\partial \beta}{\partial \theta} \tag{30}$$

Which allows us to find the expression for $\partial l/\partial \theta$. With these expressions we can can complete the optimization of (25).

5.2 Results from Simulated Data

To demonstrate the identification procedure we implemented a time-stepping LCP script to simulate a block and an ellipse with unit mass (kg), coefficient of restitution of 0.6 bouncing on a flat rigid fixed surface that lies perpendicular to the direction of the gravitational field and has 0.7 coefficient of friction. The ratio of mass to second moment of inertia for the block and ellipse are 6 and 0.8 (m^2) respectively. To generate data, both bodies were given a random set of initial positions and velocities and the simulation was run 100 times. Traces of example trajectories for the block and ellipse are shown in Figs. 5 and 6.

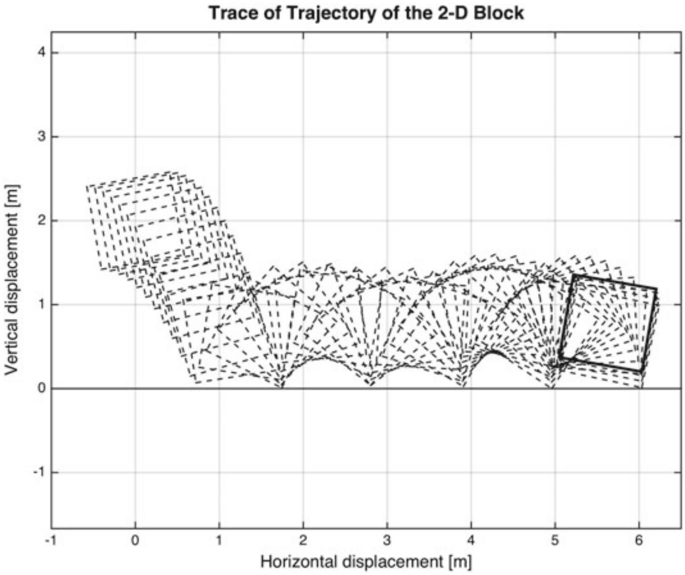


Fig. 5 2D block trace

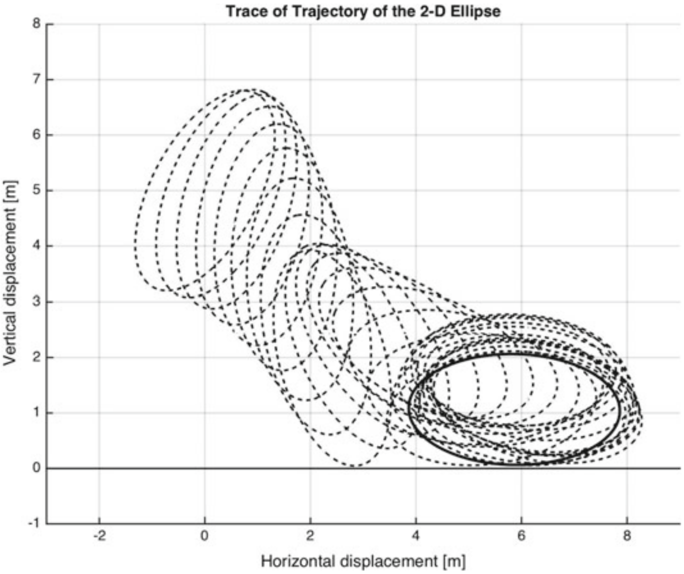


Fig. 6 2D ellipse trace

We added gaussian noise $\sim\mathcal{N}(0, \sigma^2)$ to the simulated data collected (configurations and velocities) where σ is a function of signal to noise ratio. We then used the resulted signals to calculate m/J following the least squares formulation in (25).

Table 1 shows the numerical results of the optimization as a function of signal to noise ratio where the first column denotes the mean error between predicted and actual m/J value, the second column denotes the percent error and the final column denotes the magnitude of the noise on measurements. We see good agreement between the predicted and true parameter with low levels of noise and a steady deterioration of prediction as noise is increased. We attribute the increasing error mostly to the contact Jacobians. Poor evaluation of these variables results in poor behavior prediction, which makes it difficult to estimate parameters.

Table 1 2D Block numerical simulation and identification results

Block		Ellipse		Noise
Mean error (m^2) ± Std.	% error	Mean error (m^2) ± Std.	% error	S.N.R. (dB)
−0.021 ± 0.097	−0.35	−0.031 ± 0.101	−0.31	40
−0.125 ± 0.311	−2.083	−0.136 ± 0.443	−2.34	30
−0.659 ± 0.912	−10.98	−0.712 ± 0.820	−11.71	20
−3.491 ± 1.366	−58.197	−4.891 ± 1.938	−80.44	10

5.3 Results from Experimental Data

To further validate the identifiability analysis we constructed the experimental setup in Figs. 7 and 8. We used two flat sheets of glass with support spacers to constraint in the plane the motion of a falling object. For objects we used a 3D printed square of 2 in side and an ellipse with major and minor radii of 1.5 in and 1 in. For observing the motion objects we used AprilTags [14] and a Fastec TS3 (Fastec Imaging Corp, San Diego, CA) high speed camera recording at 500 fps, which proved relatively sufficient to extract positions and orientations of the objects and velocity estimates by low-pass filtering and differentiation. For each drop experiment we considered the first 3 bounces and, and using the recovered configurations and velocities we evaluated the contact Jacobians and formed the optimization in (25).

Figures 9 and 10 show the regression results from the data collected. We note the very good agreement between the identified and true inertial parameter *m/I*. Part

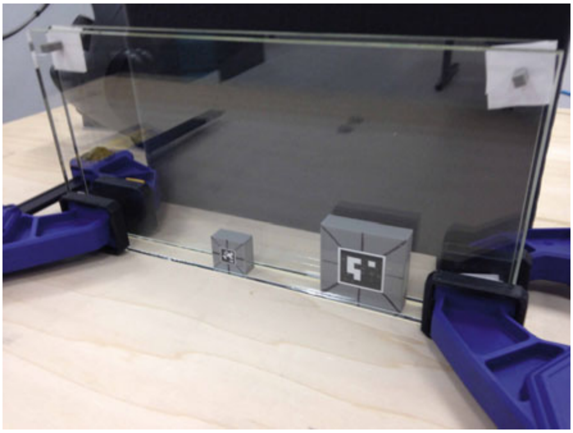


Fig. 7 Experimental setup

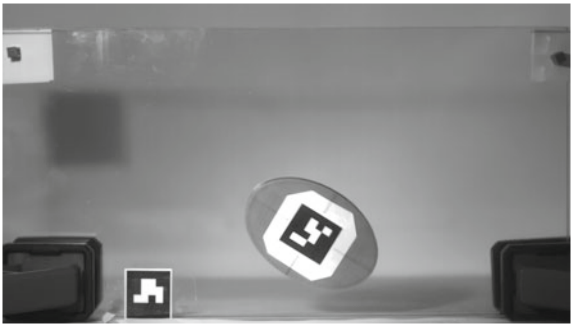


Fig. 8 Frame from ellipse drop

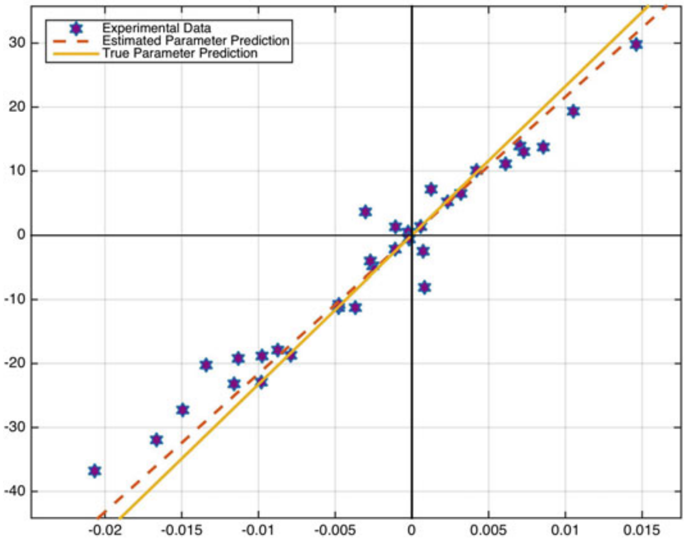


Fig. 9 Regression results: block, horizontal axes: *A* vertical axes: *Y*

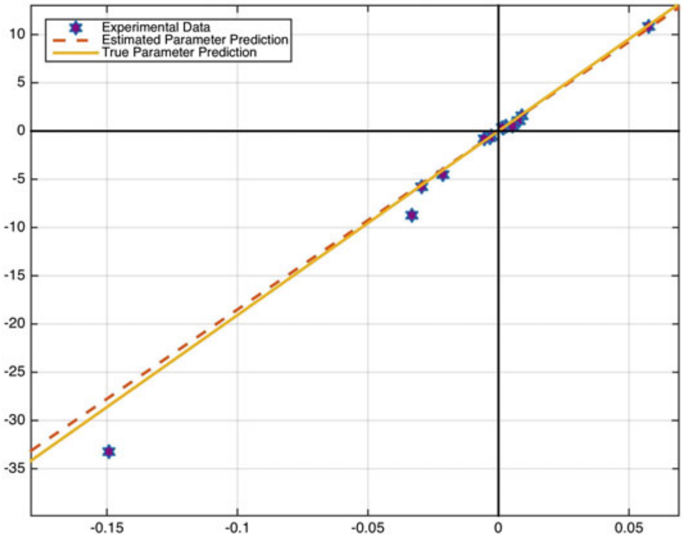


Fig. 10 Regression results: ellipse, horizontal axes: *A* vertical axes: *Y*

of the discrepancies can be attributed to small errors due to friction and jittering of the objects as they slide over the glass and small deformations during contact. The glass and the gap between were chosen to minimize friction and undesired motions such as rotations out of plane, but small disturbances are difficult to prevent. The deformations at contact were very small due to the rigidity of the objects but small amounts of deformation may mean inaccuracies in the actual contact Jacobian and can influence identification (Fig. 8).

6 Conclusions and Future Work

In this paper we investigated the problem of identifiability of inertial parameters and contact forces for a rigid body as it interacts with the environment through contact. The problem was broken down into the scenarios of sticking and sliding contact, with and without the presence of known external forces acting on the body (other than gravity). We showed that given a time history of the kinematic measurements of the object, i.e. its positions, orientations and derivatives of these quantities, without external force the parameters identifiable are the ratio of mass moment of inertia to mass of the object, and the ratio of the tangential and normal forces to mass. We also demonstrated that a known external force (other than gravity) acting on the object during the contact phase, can help in decouple the mass and mass moment of inertia, as well as the tangential and normal forces.

We validated the identifiability analysis on two planar free-falling rigid bodies undergoing frictional impact with the environment and showed that the results proved to be consistent with the predictions made by the identifiability analysis. The analysis was performed under assumptions which constitute the limitations of the work and serve as possible motivation for future efforts in this type of analysis. Future work could address issues such as increasing the number of simultaneous contacts, the number of rigid bodies, articulated rigid bodies, and extending the formulation to incorporate uncertainties in geometry.

References

1. Anitescu, M., Potra, F.A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. Nonlinear Dyn. **14**(3), 231–247 (1997)
2. Anitescu, M., Potra, F.A.: A time-stepping method for stiff multibody dynamics with contact and friction. Int. J. Numer. Methods Eng. **55**(7), 753–784 (2002)
3. Atkeson, C.G.: State estimation of a walking humanoid robot. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 3693–3699 (2012)
4. Brogliato, B., ten Dam, A., Paoli, L., Génot, F., Abadie, M.: Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. Appl. Mech. Rev. **55**(2), 107 (2002)
5. Chavan Dafle, N., Rodriguez, A.: Prehensile pushing: in-hand manipulation with push-primitives. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), p To appear (2015). http://hdl.handle.net/1721.1/98114

6. Erdmann, M.: Observing pose and motion through contact. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 723–729. IEEE (1998)
7. Gautier, M., Khalil, W.: On the identification of the inertial parameters of robots. In: Proceedings of the 27th IEEE Conference on Decision and Control, pp 2264–2269. IEEE (1988)
8. Khosla, P., Kanade, T.: Parameter identification of robot dynamics. In: 24th IEEE Conference on Decision and Control, pp. 1754–1760. IEEE (1985)
9. Koval, M.C., Dogar, M.R., Pollard, N.S., Srinivasa, S.S.: Pose estimation for contact manipulation with manifold particle filters. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4541–4548. IEEE (2013)
10. Lynch, K.M., Mason, M.T.: Stable pushing: mechanics, controllability, and planning. Int. J. Robot. Res. **15**(6), 533–556 (1996)
11. Müller P, Pöschel T (2011) Two-ball problem revisited: limitations of event-driven modeling. Physical review E, Statistical, nonlinear, and soft matter physics 83(4 Pt 1):041,304
12. Platt, R., Kaelbling, L.: Efficient planning in non-gaussian belief spaces and its application to robot grasping. ...Symposium on Robotics .. (2011)
13. Posa, M., Cantu, C., Tedrake, R.: A direct method for trajectory optimization of rigid bodies through contact. Int. J. Robot. Res. **33**(1), 69–81 (2013)
14. Richardson, A., Johannes, S., Olson, E.: AprilTag: a robust and flexible visual fiducial system. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3400-3407 (2011)
15. Rodriguez, A., Bourne, D., Mason, M., Rossano, G.F.: Failure detection in assembly: Force signature analysis. In: IEEE International Conference on Automation Science and Engineering, pp. 210–215. IEEE (2010)
16. Salawu, O.: Detection of structural damage through changes in frequency: a review. Eng. Struct. **19**(9), 718–723 (1997)
17. Slotine, J.J.E.: On the adaptive control of Robot manipulators. Int. J. Robot. Res. **6**(3), 49–59 (1987)
18. Stewart, D., Trinkle, J.C.: An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. Int. J. Numer. Methods Eng. **39**(15), 2673–2691 (1996)
19. Stewart, D.E.: Rigid-body dynamics with friction and impact. SIAM Rev. **42**(1), 3–39 (2000)
20. Trinkle, J.: A dynamic Bayesian approach to real-time estimation and filtering in grasp acquisition. In: IEEE International Conference on Robotics and Automation, pp. 85–92. IEEE (2013)
21. Yu, K.T., Leonard, J., Rodriguez, A.: Shape and Pose Recovery from Planar Pushing. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), p To appear (2015). http://hdl.handle.net/1721.1/100413
22. Zhang, L., Trinkle, J.C.: The application of particle filtering to grasping acquisition with visual occlusion and tactile sensing. In: IEEE International Conference on Robotics and Automation, pp. 3805–3812. IEEE (2012)

The Importance of a Suitable Distance Function in Belief-Space Planning

Zakary Littlefield, Dimitri Klimenko, Hanna Kurniawati and Kostas E. Bekris

1 Introduction

Uncertainty is ubiquitous, since sensing is never perfect, actuators have errors, and a robot's operating environment is often unknown. Due to this imperfect information and errors, the exact robot state is never perfectly known. Therefore, instead of finding an optimal solution in the state space, many methods represent uncertainty about the robot state as a probability distribution, and plan in the set of distributions over states, called the belief space [12, 24, 25, 31]. The computational complexity of planning in the belief space is much higher than in the state space because the size of the belief space is doubly exponential in the number of state space dimensions. Nevertheless, recent advances have shown that motion planning in belief space is becoming practical for many medium size problems [1, 6, 9, 32].

Background: Interestingly, progress in belief-space planning has been achieved through similar tools to those used in the deterministic case. In particular, this progress was achieved by sampling a small set of representative beliefs and planning with

Zakary Littlefield is supported by the NASA Space Technology Research Fellowship (NNX13AL71H). Dimitri Klimenko is supported by the Australian Postgraduate Award. Work by the Rutgers authors has been supported by NSF CCF:13307893, NSF IIS:1451737.

Z. Littlefield · K.E. Bekris (✉)
Computer Science Department, Rutgers University, New Brunswick, NJ, USA
e-mail: zwl2@cs.rutgers.edu

K.E. Bekris
e-mail: kostas.bekris@cs.rutgers.edu

D. Klimenko · H. Kurniawati
School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD, Australia
e-mail: dimitri.klimenko@uqconnect.edu.au

H. Kurniawati
e-mail: hannakur@uq.edu.au

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_39

respect to only this small set of sampled beliefs. In fact, many methods (e.g., [3, 18, 25]) in belief-space planning are extensions of sampling-based ones for the deterministic case, such as PRM, RRT, PRM*, and RRG [4, 7]. These methods typically restrict beliefs to be represented by Gaussian parameters or consider the maximum likelihood estimate of the state.

Recent work, which has shown that one can achieve asymptotic optimality without a steering function in the deterministic case [15, 20], has the potential to allow an even more straightforward way to extend sampling-based planners to belief-space planning. The similarity between deterministic planning without a steering function and belief-space planning indicate that properties critical for deterministic motion planning are likely to be critical for belief-space motion planning as well.

Similar to sampling-based methods for the deterministic case, many equivalent approaches for belief-space planning rely on distances between beliefs to partially guide their sampling and pruning operations [8, 12, 29]. Many distance functions can be potentially used and they can have significantly different effects in belief-space planning. Nevertheless, the effectiveness of the different distance functions has not been studied in the related literature on belief-space planning.

This paper focuses on understanding the suitability of commonly used distance functions in belief-space motion planning. Commonly used functions, such as L1 and Kullback–Leibler divergence KL, in general ignore the underlying distance in the state space. As a result, two beliefs whose supports do not overlap, but lie near each other in the state space, will have the same distance as two beliefs whose supports lie very far away. Figure 1 illustrates this issue. If the supports of the beliefs are unbounded, the above problems are less severe, though they still exist. While the Wasserstein or Earth Mover's Distance (EMD) alleviates the aforementioned issue, it has been rarely used in the related literature [11, 14]. This paper presents a comparative study of the effect of these metrics in two classes of belief space planning, i.e., Non-Observable Markov Decision Processes (NOMDPs) and Partially Observable Markov Decision Processes (POMDPs).

NOMDP Evaluation: NOMDP is the simplest class of belief-space planning, a planning under uncertainty challenge where no observation is available. Despite its simplicity, NOMDP has often been applied as an intermediate solution to complex planning under uncertainty problems [10]. The simplicity of this class of problems allow us to compare the metric on various complex motion planning problems, which are still unsolvable when the challenge is partially observable. To solve NOMDPs, this

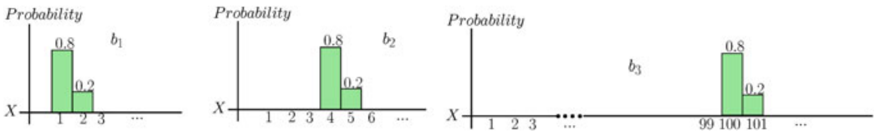


Fig. 1 Illustration of the problem with L1 and KL. Suppose the state space $\mathbb{X}$ is the 1-dim. Natural numbers and the distance $d_{\mathbb{X}}(x, x') = |x - x'|$. Then, the L1 distance $D_{L1}(b_1, b_2) = D_{L1}(b_1, b_3)$, the KL distance $D_{KL}(b_1, b_2) = D_{KL}(b_1, b_3)$, and the EMD distance $D_W(b_1, b_2) < D_W(b_1, b_3)$

paper leverages recent results in deterministic motion planning that show asymptotically optimal solutions [7] can be computed by sampling-based methods that do not require steering functions [15, 20]. These methods are extended to solve NOMDP problems. Such extensions are simpler compared to extending methods that require a steering function, because computing a good steering function in the belief space is non-trivial. Simulation results indicate that as the NOMDP problem becomes more complex, the differences in the effectiveness of different distance functions become quite prominent. In fact, in state spaces with more than 4 dimensions, just by replacing L1 or KL distance with EMD, the speed of solving the problems improves substantially and the problems transition from virtually unsolvable to solvable.

POMDP Evaluation: The second class of problems in our comparative study is the Partially Observable Markov Decision Processes (POMDP). To solve POMDP problems, Monte Carlo Value Iteration (MCVI) [2] is used here, which is an offline POMDP-solver designed for problems with continuous state spaces. In this paper, we only apply the various metrics to a 2D navigation problem when evaluating the performance in the POMDP framework, due to the limitation of existing POMDP solvers in solving problems with large action spaces. Although this limitation means we cannot yet show the full potential of EMD in POMDP, the preliminary result reveals that EMD could significantly reduce the number of belief-space samples that sampling-based POMDP-solvers need to reach a certain solution quality.

Overall Contributions: The results of this comparative study indicate that EMD is more suitable than L1 and KL, and could significantly improve the performance of belief space planning, even though its computation can be computationally more expensive. Steps towards the efficient computation of the EMD are also described here. Furthermore, this paper shows that EMD carries the Lipschitz continuity of the cost function in the state space to Lipschitz continuity of expected cost in the belief space. This is useful because this property is used in the convergence analysis of several asymptotically optimal motion planning methods without a steering function [15, 20], including methods for belief-space planning [12, 13].

2 Problem Setup for Comparative Study

A POMDP is a mathematically principled framework for planning under uncertainty in discrete-time. Formally, a POMDP is defined as a tuple $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$, where S is the set of states, A is the set of actions, and O is the set of observations. The notation T represents motion uncertainty and is defined as a conditional probability function $T(s, a, s') = \mathbb{P}(s'|s, a)$, where $s, s' \in S$ and $a \in A$. The notation Z represents sensing uncertainty and is defined as a conditional probability function $Z(s', a, o) = \mathbb{P}(o|s', a)$, where $s' \in S$, $a \in A$, and $o \in O$. At each step, a POMDP agent is in a state $s \in S$, takes an action $a \in A$, moves from s to an end state $s' \in S$, perceives an observation $o \in O$, and receives a reward $R(s, a)$ for taking action a from state s. However, a POMDP agent never knows this exact state, and instead reasons with respect to distributions over states, called beliefs. At each step, the agent's belief estimate is updated based on the action it just performed and the observation perceived. The agent's goal is to choose a suitable sequence of actions that will maximize its expected total reward, when the agent starts from the initial belief b_0. When the sequence of actions has infinite length, a discount factor $\gamma \in (0, 1)$ is specified, so that the total reward is finite and the problem is well defined.

The solution to a POMDP problem is a mapping from beliefs to the best actions, and is called an optimal policy. A policy π induces a value function $V_\pi(b)$, which specifies the expected total reward of executing policy π from belief b, and is computed as $V_\pi(b) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|b, \pi]$. An optimal policy is the policy π^* whose value function $V_{\pi^*}(b)$ is the highest among all possible policies for any belief b.

A NOMDP is a class of POMDPs where observations are not available, i.e., $Z(s, a, na) = 1$ for any $s \in S$ and $a \in A$. As a consequence, the solution of an NOMDP is a nominal path, which maps time steps to the best actions.

The problem setups for both NOMDPs and POMDPs are geared towards motion planning problems, finding a strategy to move from one state to another. Both NOMDPs and POMDPs use the same representation for state and action spaces, and for motion uncertainty. The state space is a continuous metric space, denoted by $\mathbb{X}$, that is diffeomorphic to $\mathbb{R}^n$, where n is the dimensionality of $\mathbb{X}$. The action space is the same as the control space, denoted as $\mathbb{U}$, and typically has lower or equal dimension than $\mathbb{X}$. The motion uncertainty comes from actuation error, and is represented as a stochastic dynamical system of the form:

$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(x(t), \tilde{u}(t))dt, \tag{1}$$

where $x(t) \in \mathbb{X}$, $\tilde{u}(t) \in \mathbb{U}$ and Δt is a discrete time step. The control $\tilde{u}(t)$ is the one executed by the system, which does not always correspond to the input control $u(t)$ due to actuation error, i.e., there is an error vector w based on which:

$$\tilde{u}(t) = u(t) + w. \tag{2}$$

The vector w is additive noise sampled from a probability distribution, which can be any type of distribution.

Given that NOMDPs do not have observations, NOMDPs and POMDPs differ in the objective function. NOMDPs use an objective function that is commonly used in motion planning. In this paper, the defined NOMDP challenge involves finding a sequence of control inputs that minimizes the cost, while ensuring that the collision probability is lower than a given threshold and the probability of reaching the goal is higher than a given threshold. More precisely, suppose $p = (u_1, u_2, \ldots, u_m)$ is the sequence of input controls for the system. Each control input in the sequence is applied for a unit time step, subsequently starting from the initial belief b_0. This application will induce a trajectory $\pi_b^p = (b_0, b_1, b_2, \ldots, b_m)$ that arises when applying Eq. 1 to the states in the support set of belief b_0 and following the sequence p, given the noise model w. The duration of a trajectory π is denoted as T_π. If the trajectory π is induced by p with controls for m time steps, then $T_\pi = m \cdot \Delta t$. A belief state along a trajectory π at time t is denoted as $\pi(t)$. The cost of the trajectory $\pi_{b_0}^{p}$ induced by p is $c(\pi_{b_0}^{p}) = \sum_{i=1}^{m} \int_{x \in \mathbb{X}} cost(x, u_i) \cdot b_{i-1}(x)dx$. The goal of the NOMDP solver is to find a control sequence p that generates a trajectory $\pi_{b_0}^{p}$ for which the probability of being in the goal region at time T_π is above the threshold: $\mathbb{P}(\pi(T_\pi) \in \mathbb{X}_G)) > \mathbb{P}_{goal}$, the probability for being in the free space is above the threshold: $\mathbb{P}_{free}(\pi) > \mathbb{P}_{valid}$, and which minimizes the cost $c(\pi)$.

The objective function of POMDPs uses the commonly applied definition (Sect. 2). The reward function, $R(s, a)$ for any pair of state s and action a, is a summation of collision cost and reward for being in the goal. The reason for this difference in objective function is that solving a POMDP with probability constraints is still a relatively open problem.

3 Distance Functions for Belief-Space Planning

Computing the optimal POMDP policy is computationally intractable [19]. In the past several years, however, methods which can compute a good approximation to the optimal policy fast have been proposed. Most of them rely on sampling, and depend on distance functions. They can be classified into several distinct approaches.

Point-based techniques [12, 23, 27, 28] use sampling to construct a small but representative set of beliefs, and find an approximately optimal policy by iteratively computing Bellman backups [23] on the sampled beliefs. The key is the sampling strategy, and some of the successful methods use distance functions to guide sampling. For example, PBVI [23] only keeps a newly sampled belief whenever the L1 distance between the new and existing belief is larger than a certain threshold, while GCS [13] uses EMD, and an alternative [8] uses KL divergence to perform similar rejection sampling. A fast offline point-based solver [12] also uses L1 distance for pruning. Point-based methods can handle any type of distributions but they have difficulties in solving problems in large continuous action spaces.

This work evaluates four distance functions for belief-space planning. Two of them, Kullback–Leibler (KL) and L1 divergence, are commonly used in belief-space planning. In general, these two functions do not consider the underlying state-space distance when computing distances between beliefs. This characteristic limits the effectiveness of these two distance functions to guide sampling and pruning in the belief space. To alleviate these problems, two alternatives are proposed here, Wasserstein (also known as Earth Mover's Distance (EMD)) and Hausdorff distance. Both of these functions compute distances based on an underlying state space distance. They have not been widely used in belief-space planning, but they are widely used in computer vision and optimal transport, which means efficient implementations for these distance computations abound. For the following, the beliefs are defined as distributions over a common state space, denoted as $\mathbb{X}$.

A. **Wasserstein Distance/**EMD: Intuitively, Wasserstein distance or EMD computes the distance between two distributions as the amount of work to move the probability mass of one distribution to another. More formally, EMD is defined as:

$$D_W(b, b') = \inf_f \left\{ \int_{x \in \mathbb{X}} \int_{x' \in \mathbb{X}} d_{\mathbb{X}}(x, x') f(x, x') \partial x \partial x' \right| \\ \left. b = \int_{x'} f(x, x') dx' \ , \ b'(x') = \int_x f(x, x') dx \right\}, \tag{3}$$

where $d_{\mathbb{X}}$ is the distance in the state space $\mathbb{X}$ and f is a joint density function.

EMD carries the Lipschitz continuity of a cost function in the state space to Lipschitz continuity of expected cost in the belief space. Formally, let $d_{\mathbb{X}}$ be the distance function on a separable state space $\mathbb{X}$ and $cost(x, u)$ be the cost of applying control $u \in \mathbb{U}$ at state $x \in \mathbb{X}$ for the duration of one unit time. Let beliefs b and b' be distributions over $\mathbb{X}$ and let the cost function w.r.t. a belief b be $cost(b, u) = \int_{x \in \mathbb{X}} cost(x, u) b(x) dx$.

Theorem For any control input $u \in \mathbb{U}$, if the cost function satisfies Lipschitz continuity in the state space, i.e., $|cost(x, u) - cost(x', u)| \leq C \cdot d_{\mathbb{X}}(x, x')$ for any $x \in \mathbb{X}$, then the cost function in the belief space with the EMD metric is also Lipschitz continuous: $|cost(b, u) - cost(b', u)| \leq C \cdot D_W(b, b')$.

Proof The proof is based on the well-known Kantorovich duality of the Wasserstein distance [5]. The Kantorovich distance is defined as

$$D_K(b, b') = \sup_{g \in Lip_1} \left(\int_{x \in \mathbb{X}} g(x) b(x)\, dx - \int_{x \in \mathbb{X}} g(x) b(x)\, dx \right),$$

where Lip_1 is the set of all 1-Lipschitz functions over $\mathbb{X}$. Now, by definition:

$$\left| cost(b, u) - cost(b', u) \right| = \left| \int_{x \in \mathbb{X}} cost(x, u) b(x)\, dx \ - \int_{x \in \mathbb{X}} cost(x, u) b'(x)\, dx \right| . \tag{4}$$

To satisfy the 1-Lipschitz requirement of the Kantorovich distance, we can use the scaled distance function in the state space $\mathbb{X}$, i.e., $d'_{\mathbb{X}} = C \cdot d_{\mathbb{X}}$. It is known that if we use $d'_{\mathbb{X}}$ as the state space distance, then the belief space distance $D'_W = C \cdot D_W = C \cdot D_K$. The last equality is due to the duality of the Kantorovich and Wasserstein distance. This means that by using the state space metric $d'_{\mathbb{X}}$, Eq. (4) can be bounded as: $\left| cost(b, u) - cost(b', u) \right| \leq C \cdot D_K(b, b')$, and hence $\left| cost(b, u) - cost(b', u) \right| \leq C \cdot D_W(b, b')$. □

From the literature [11], it is known that the value function in POMDPs is Lipschitz continuous when the metric in the belief space is EMD. The above theorem uses a similar proving strategy similar to the prior work [11] but generalizes the result to any cost function that is Lipschitz continuous in the state space.

B. **Hausdorff Distance**: Hausdorff distance is a popular metric in computer vision. This function computes distance between two sets based on a max-min operation. In terms of distances between beliefs, it is possible to define it with respect to the beliefs' support set. Slightly abusing the notation of the arguments, this function can be defined in the belief space as:

$$D_H(b, b') = \max\{d_H(b, b'), d_H(b', b)\}$$

$$\text{where } d_H(b, b') = \max_{x \in support(b)} \left\{ \min_{x' \in support(b')} \left\{ d_{\mathbb{X}}(x, x') \right\} \right\},$$

and $support(b) = \{x \in \mathbb{X} \mid b(x) > 0\}$, while $d_{\mathbb{X}}(x, x')$ is the state space distance.

Hausdorff is simpler to compute than EMD. Nevertheless, since Hausdorff measures distances between sets, rather than distributions, it ignores the probability values. This means that if two distributions have exactly the same support, even though the probabilities are significantly different, the two distributions will be considered to lie at the same point in the belief space. This problem is exactly the opposite of the problems faced by L1 and KL-divergence, as described below.

C. KL**-Divergence**: A commonly used distance function in belief-space planning is the Kullback–Leibler (KL) divergence. It measures the difference in information content between two distributions. More formally, KL divergence is defined as:

$$D_{KL}(b, b') = \int_{x \in \mathbb{X}} b(x) \left(\ln b(x) - \ln b'(x) \right) dx. \tag{5}$$

In the general case, KL-divergence does not consider the underlying state space distance. But, for certain distributions, it does so to some extent. For instance, when applied to Gaussian beliefs, KL-divergence is partially based on the Euclidean distance of the mean. More completely, the KL-divergence between two multivariate Gaussian beliefs, denoted as $b_1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $b_2 = \mathcal{N}(\mu_2, \Sigma_2)$, is

$$D_{KL}(b_1, b_2) = \frac{1}{2} \left((\mu_2 - \mu_1)^T \, \Sigma_1^{-1} \, (\mu_2 - \mu_1) + \, tr \left(\Sigma_2^{-1} \Sigma_1 \right) + \ln \frac{|\Sigma_2|}{|\Sigma_1|} - K \right), \tag{6}$$

where K is the dimension of the underlying state space.

When applied to general beliefs with continuous state space, one usually starts by discretizing the state space $\mathbb{X}$ into uniform grid cells, and then computes the KL-divergence using Eq. (5) as if the beliefs are discrete distributions. This computation means that state-space distance is *only* considered up to the resolution of the grid cells, which is very limited.

Note that KL divergence is not symmetric and hence is not a true metric. One can symmetrize KL simply by adding the reverse distance or by computing distance to the mean of the two distributions (i.e., $\frac{D_{KL}(b, \frac{b+b'}{2}) + D_{KL}(b', \frac{b+b'}{2})}{2}$). The later strategy is called Jensen-Shannon divergence. In the accompanying comparative study, two implementations are used: (i) the Jensen-Shannon and (ii) an approximation of the beliefs using Gaussian distributions per Eq. (6).

D. `L1` **Distance**: Another commonly used distance function in belief-space planning, and also the simplest to compute, is `L1`, which is defined as:

$$D_{L1}(b, b') = \int_{x \in \mathbb{X}} |b(x) - b'(x)|\, dx.$$

Most belief-space planners use `L1` distance for discrete distributions, that is $\mathbb{X}$ is discrete and `L1` is computed as a summation over $\mathbb{X}$ rather than an integration.

When the state space is continuous and `L1` distance is used, then the state space is discretized at a suitable resolution and the `L1` distance computation is applied as if the beliefs are distributions over the discretized state space. This discretization means that `L1` distance, similar to `KL` divergence, considers the underlying state-space distance *only* up to the resolution of the state space discretization, which is often very limited.

4 Algorithms for Comparative Study

To determine which distance function is most suitable for belief space planning given its complexity, we employ a sampling-based framework.

A. **Non-Observable Markov Decision Processes** (`NOMDP`**s**): The framework follows tree sampling-based planners. It is based on a `BestNear` variant of `RRT` [17, 30] but is adapted to belief space planning. It has been formally shown - at least for the deterministic case - that this method can improve path quality over time even when there is no access to a steering function [15, 16]. Convergence to optimality requires Lipschitz continuity in the state and control spaces.

Algorithm 1: SPARSE_BELIEF_TREE($\mathbb{B}$, $\mathbb{U}$, b_0, N, T_{max}, δ_n, δ_s)

```
$G = \{V \rightarrow \{b_0\}, E \rightarrow 0\}$;
for $N$ iterations do
    $b_{selected} \leftarrow$ SelectNode ($\mathbb{B}$,$V$,$\delta_n$);
    $b_{new} \leftarrow$ Random_Prop ($b_{select}$, $\mathbb{U}$, $T_{max}$);
    if IsNodeLocallyBest ($b_{new}$, $S$, $\delta_s$) then
        $V \leftarrow V \cup \{b_{new}\}$;
        $E \leftarrow E \cup \{\overline{b_{select} \rightarrow b_{new}}\}$;
        Prune_Dominated_Nodes($b_{new}$, $V$, $E$, $\delta_s$);
```

An outline of the algorithmic framework is shown in Algorithm 1. As input, the planner receives the belief space $\mathbb{B}$, control space $\mathbb{U}$, initial belief b_0 and number of iterations N. In addition, Algorithm 1 receives a maximum propagation duration T_{max} and two radius parameters δ_n and δ_s, which are explained below. The selection process (Line 3) is summarized in Algorithm 2. A random δ-belief distribution b_{rand} is first sampled in the belief space $\mathbb{B}$ and the set of belief distributions B_{near} within a distance threshold δ_n is computed. If no belief is found within this threshold, then the closest distribution is returned, similar to the basic RRT approach. If there are beliefs in the set $D_{\mathbb{B}}$, then the one with the best cost, e.g., trajectory duration, is returned.

Algorithm 2: SelectNode($\mathbb{B}$, V, δ_n)

```
$b_{rand} \leftarrow$ Sample_Belief($\mathbb{B}$);
$B_{near} \leftarrow$Near($V, b_{rand}, \delta_n$);
If $B_{near} = \emptyset$ return Nearest($\mathbb{V}, b_{rand}$);
Else return $\arg\min_{b \in B_{near}}$ cost($b$);
```

Line 4 of Algorithm 1 is the propagation primitive used to add new belief states to the tree. The subroutine is detailed in Algorithm 3. First, a time duration is uniformly sampled up to a maximum time T_{max}. The sampled time must be a constant multiple of the minimum Δt in order to satisfy the requirement for piece-wise constant control inputs, which are sampled after the time duration. Given these control inputs, the belief distribution can be updated through the transition model.

Algorithm 3: Random_Prop(b_{prop}, $\mathbb{U}$, T_{max})

```
$t \leftarrow$ Sample(0, $T_{max}$); $\Upsilon \leftarrow$ Sample($\mathbb{U}$, $t$);
return $b_{new} \leftarrow \int_0^t \mathbb{T}(b(t), \Upsilon(t)) b_{prop}\, dt$;
```

To perform pruning, the set of nodes V in the tree data structure $G(V, E)$ of algorithm Algorithm 1 are split into two subsets V_{active} and $V_{inactive}$. Nodes in V_{active} have the best cost from the start in a neighborhood of radius δ_s around them. These nodes are considered for propagation by the algorithm. Nodes that are dominated by others in terms of path cost in their local neighborhood, are:

- Pruned if they are leaves or have no children in V_{active}.
- Or added to the set $V_{inactive}$ if they do have children in V_{active}. Nodes in $V_{inactive}$ are not selected for propagation.

Algorithm 4 details a simple operation to determine how to prune existing nodes in the tree. It is called only if the new belief distribution b_{new} has the best cost in its local δ_s neighborhood. Then, the set of existing nodes that are dominated in terms of path cost are set to be inactive. If these nodes are also leaves of the tree, they are removed from the tree. This process can continue up the tree if the parents were also inactive. It helps to reduce the size of the stored tree and promotes the selection of nodes with good path quality.

Algorithm 4: `Pruning`(b_{new}, G, δ_s)

```
$B_{dominated} \leftarrow$ FindDominated$(G, b_{new}, \delta_s)$;
for $b \in B_{dominated}$ do
    $b.set\_inactive()$;
    while IsLeaf $(b)$ and $b$ is inactive do
        $x_{parent} \leftarrow$Parent$(b)$;
        $E \leftarrow E \setminus \{\overline{b_{parent} \rightarrow b}\}$;
        $V \leftarrow V \setminus \{b\}$;
        $b \leftarrow b_{parent}$;
```

It is apparent that throughout the operation of this sampling-based framework for belief space planning, there is heavy use of distance calls and a significant dependence on the choice of the distance function.

B. **Partially Observable Markov Decision Processes** (`POMDP`**s**): The framework follows a point-based approach, similar to Monte Carlo Value Iteration (`MCVI`)[2], an extension of `SARSOP` [12]. The later is considered the fastest offline and general `POMDP` solver for problems with continuous state space. The `MCVI` method is slightly modified to use the distance function for pruning sampled beliefs. Algorithm 5 details the algorithm employed for `POMDP`s, which incorporates `SARSOP` as its sampling strategy (Line 6). The function $Nearest(b)$ in Line 7 and 8 returns the belief in the tree $\mathscr{T}$ that is nearest to belief b. The only modification made to `MCVI` is the addition of the condition in Line 7, which rejects a newly sampled belief whenever its nearest neighbor in $\mathscr{T}$ is within a given threshold.

Algorithm 5: `ModifiedMCVI`(b_0)

```
1  Initialize belief tree $\mathscr{T}$ by setting $b_0$ as the root of $\mathscr{T}$ ;
2  Initialize policy graph $\Gamma$ with an empty graph ;
3  Initialize the upper bound $\overline{V}$ of the value function to be $\infty$ ;
4  Initialize the lower bound $\underline{V}$ of the value function to be -$\infty$ ;
5  while $|\overline{V}(b_0) - \underline{V}(b_0)| > \epsilon$ do
6      Sample new belief $b$ ;
7      if $distance(b, Nearest(b)) < \delta_{th}$ then
8          $b \leftarrow Nearest(b)$ ;
9      else
10         Add $b$ to $\mathscr{T}$ ;
11     MCVI Backup($\Gamma$, b) ;
12     $\overline{V}$ = UpdateUpperBound(b) ;
13     $\underline{V}$ = UpdateLowerBound(b) ;
```

C. **Algorithmic Details to Improve Speed**: The implementation of the distance functions was optimized to reduce computation time. `KL` and `L1` distances were implemented through the use of binning. The full state space grid is not required, only the nonzero entry bins. This allows dealing with high-dimensional problems, saves space, and reduces computation time. The Hausdorff and EMD functions do not require binning but become much more efficient using bins. The bin width is chosen so that several individual bins can be contained within the pruning radius δ_s when solving NOMDPs. This discretization introduces an approximation error.

To further speed up the EMD computation, an approximation is employed to occasionally replace the expensive call to the standard method. The motivation stems from the fact that if two distributions are too far apart, their EMD distance is close to the distance between their centroids. So, if two distributions overlap according to their diameters and their discretization, then the standard call to the EMD computation is performed. Otherwise, the distance between the two centroids is used, which is a fast operation.

In the following experiments KL-Gaussian represents the approximation of a set of particles as a Gaussian distribution and performs distances using the closed form expression from Eq. 6. This distance function does not use binning as the Gaussian parametrization provides an efficient representation.

5 Experimental Evaluation

5.1 Non-observable Markov Decision Processes (NOMDPs)

All distances are evaluated in the scenarios shown in Fig. 2. All scenarios produce non-Gaussian belief distributions due to the nonlinear dynamics. The objective is to reach a goal region in state-space with at least 90% probability. Valid trajectories have a collision probability of less than 20%.

2D Rigid Body. This introductory example is a 2D rigid body moving among two narrow corridors. Due to errors in actuation and requirement for collision avoidance, the robot can only move through the lower corridor. The state space is 2D (x, y) and the control space is also 2D (v, θ), where $v \in [0, 10]$ and $\theta \in [-\pi, \pi]$. The dynamics

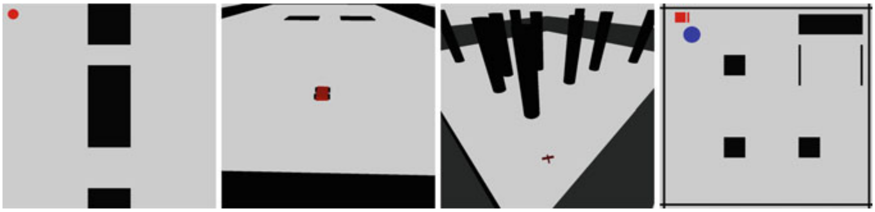


Fig. 2 The considered scenarios, which are better viewed in color. The 2D rigid body (*left*) must move from the *left* to the *right* side. The car (*left middle*) must drive through one of 3 corridors. The fixed-wing airplane (*middle right*), must move to the opposite corner while avoiding cylinders. The manipulator (*right*) must push the round object into the storage location at the *top right*

follow this model:

$$\dot{x} = \tilde{v}\cos(\tilde{\theta}), \quad \dot{y} = \tilde{v}\sin(\tilde{\theta}),$$

where $\tilde{v} = v + \mathscr{N}(0, 1)$ and $\tilde{\theta} = \theta + \mathscr{N}(0, 0.3)$. Numerical integration is performed using Euler integration.

2^{nd}**-order Car**. A four-wheeled vehicle with dynamics needs to reach a goal region, while ensuring low collision probability. The state space is 5D $(x, y, \theta, v, \omega)$, the control space is 2D $(a, \dot{\omega}) \in ([-1, 1], [-2, 0.2])$, actuation error is $(\mathscr{N}(0, 0.05), \mathscr{N}(0, 0.002))$, and the dynamics are:

$$\begin{aligned} &\dot{x} = v\cos(\theta)\cos(\omega), \quad && \dot{y} = v\sin(\theta)\cos(\omega), \\ &\dot{\theta} = v\sin(\omega), && \dot{v} = \tilde{a}. \end{aligned}$$

Numerical integration is performed using Runge-Kutta order four (RK4). The environment is more complex than before since there are multiple feasible paths through each of the 3 corridors.

Fixed-wing airplane. An airplane flying among multiple cylinders. The state space is 9D $(x, y, z, v, \alpha, \beta, \theta, \omega, \tau)$, the control space is 3D $(\tau_{des} \in [4, 8], \alpha_{des} \in [-1.4, 1.4], \beta_{des} \in [-1, 1])$ and the dynamics are (from [21]):

$$\begin{gathered} \dot{x} = v\cos(\omega)\cos(\theta), \qquad \dot{y} = v\cos(\omega)\sin(\theta)), \qquad \dot{z} = v\sin(\omega), \\ \dot{v} = \tau * \cos(\beta) - C_d k v^2 - g\sin(\omega), \quad \dot{\omega} = \cos(\alpha)(\frac{\tau\sin(\beta)}{v} + C_l k v) - g\frac{\cos(\omega)}{v}, \\ \dot{\theta} = v\frac{\sin(\alpha)}{\cos(\omega)}(\frac{\tau\sin(\beta)}{v} + C_l k v), \quad \dot{\tau} = \widetilde{\tau_{des}} - \tau \quad \dot{\alpha} = \widetilde{\alpha_{des}} - \alpha, \quad \dot{\beta} = \widetilde{\beta_{des}} - \beta, \end{gathered}$$

where $\widetilde{\tau_{des}} = \tau_{des} + \mathscr{N}(0, 0.03)$, $\widetilde{\alpha_{des}} = \tau_{des} + \mathscr{N}(0, 0.01)$, and $\widetilde{\beta_{des}} = \beta_{des} + \mathscr{N}(0, 0.01)$. Numerical integration is performed using RK4. This problem has a state space that is generally larger than most planners in belief space can handle computationally. Leveraging sampling-based techniques with proper distance functions makes planning for the airplane model possible.

Non-prehensile manipulator. The task is to push an object to the goal. The state space is 5D $(x_{man}, y_{man}, x_{obj}, y_{obj}, \theta_{manip})$ and the control space is 2D (v, θ), where $v \in [0, 10]$ and $\theta \in [-\pi, \pi]$. The dynamics are:

$$\dot{x}_{man} = \tilde{v}\cos(\tilde{\theta}), \qquad \dot{y}_{man} = \tilde{v}\sin(\tilde{\theta}),$$

where $\tilde{v} = v + \mathscr{U}(-1, 1)$ and $\tilde{\theta} = \theta + \mathscr{U}(-0.3, 0.3)$. Numerical integration is performed using RK4. The object cannot be moved unless the manipulator moves in the direction of the object and pushes it, which implies that there is contact between the manipulator and the object. Once pushed, the object moves as if it is attached to the manipulator. Notice that the noise model used in this setup is a uniform distribution, meaning that the resulting belief distributions are clearly non-Gaussian.

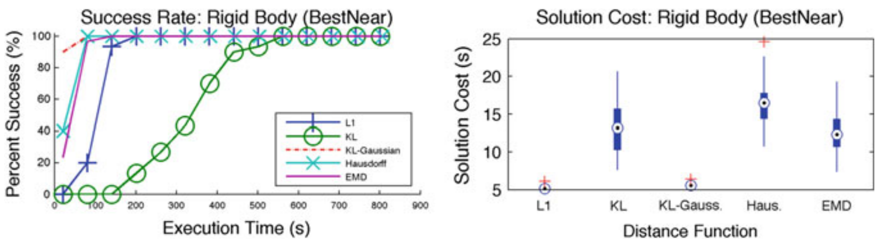


Fig. 3 Results for the 2D rigid body - better viewed in color

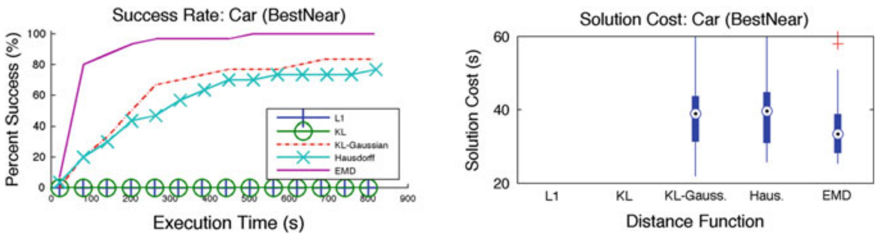


Fig. 4 Results for the car. L1 and KL failed to produce solutions within the time constraint

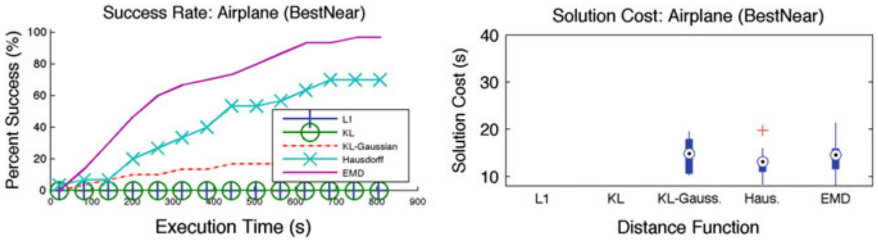


Fig. 5 Results for the airplane. L1 and KL failed to produce solutions within the time constraint

Experimental Setup and Results: Each combination of distance function and algorithm is evaluated in each of the four scenarios described in the previous subsection. The key criterion is the success rate for finding solutions in the allotted time. The distance thresholds in the algorithms for each metric are selected so that a similar number of nodes is kept among all alternatives. This allows for a fair comparison on the effect of a metric to the quality of sample placement. The experiments were executed on Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz machines. Each experiment is repeated 30 times with different random seeds. The results are averaged over these runs and are presented in Figs. 3, 4, 5 and 6.

In most scenarios, belief metrics that do consider the underlying state-space distance, such as EMD, Hausdorff, and KL-Gaussian, perform significantly better than those that do not. KL and L1 metrics consider the state space distance up to the resolution of the state space discretization (as described in Sect. 4). Such consideration is sufficient when the state space is small (as in Fig. 3). As the size of the state

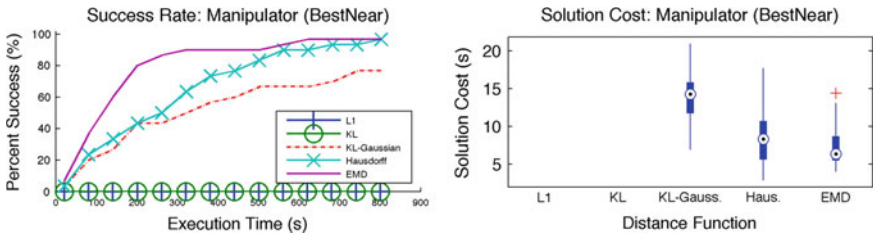


Fig. 6 Manipulation results. L1 and KL failed to produce solutions within the time constraint

space increases, this is no longer sufficient. These results corroborated the hypothesis regarding the importance of taking into account the underlying state space distance in computing distance between beliefs.

EMD performs substantially better than the alternatives since it considers both state space distance and the distributions. The Hausdorff distance performs better than L1 and KL because it still considers the underlying state-space distance. But Hausdorff does not consider the distribution, and therefore is outperformed by EMD. KL-Gaussian performs better than L1 and KL because it considers the mean and variance of the corresponding Gaussian and in this way considers the underlying state space distance. Nevertheless, the usefulness of this distance function depends on how well a Gaussian distribution approximates the actual distribution.

The path costs achieved by the different metrics for successful runs are comparable. Therefore, the success rate is the key criterion for evaluating the effect of the different metrics to the performance of belief-space planners.

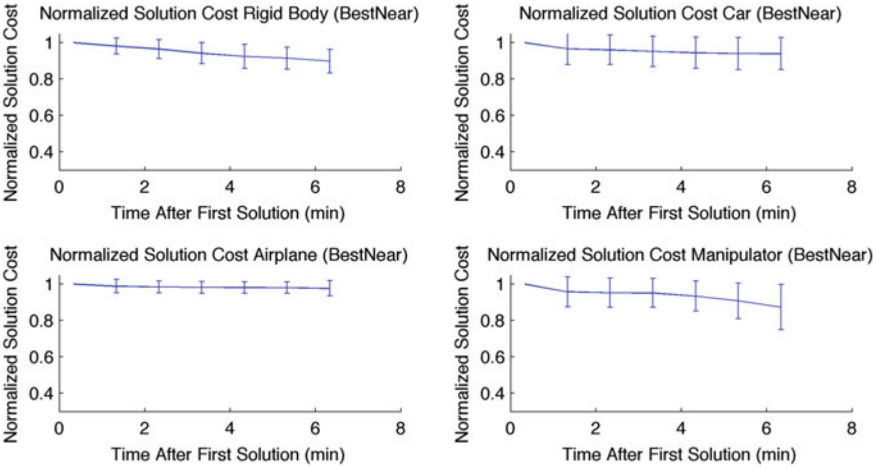


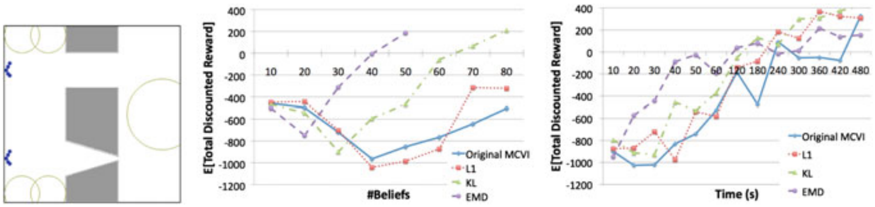
Fig. 7 The normalized costs over time for EMD show the benefit of providing improving solutions. Each trial's initial solution is normalized to 1. The cost over time is shown with one st. dev

Results on Cost Reduction Over Time: Figure 7 normalizes to one the first solution generated for each individual run of the algorithm using EMD. Then, all subsequent improvements to the path cost are plotted relative to the initial cost. The figure averages all runs, all of which show improvement over time. The improvement is most prominent for the manipulator experiments. This comparison is performed only for EMD as it provides the best performance. Path cost can also improve over time using Hausdorff and KL-Gaussian but there are fewer data points to extract useful conclusions given the reduced success ratio of these metrics.

5.2 Partially Observable Markov Decision Processes (POMDPs)

The different distance functions are tested on a 2D navigation problem (Fig. 8a). The robot is a point robot, and may start from any position marked by the blue region. Its task is to reach the goal region (the large circular region on the right) as fast as possible while avoiding collision with obstacles (the grey polygons) as much as possible. The robot's motion is discretized into 5 actions, which is moving a unit distance in the direction of N, NE, E, SE, and S. Its motion has error, which is represented as a bounded uniform distribution. The robot can only localize itself up to a certain accuracy inside the small circular regions on the left of the obstacles. The problem is modeled as a POMDP with continuous state space, but with discrete action and observation spaces. The reward function of the POMDP model is a sum of the goal reward, collision penalty, and moving cost.

Experimental Setup: The original and modified MCVI method (Algorithm 5) is evaluated separately for the L1, KL, and EMD metrics. To set the required parameters, such as distance threshold, short preliminary runs with various parameters were executed. The best parameters for each method were retained. Each method was then executed with the appropriate parameters to generate 15 different sets of policies. To generate each set of policies, the method is ran until the difference between the upper and lower bound of the initial belief is less than a given threshold, so that at the end of the runs, the quality of the policies generated by the different methods are similar. Throughout each run, the method outputs the intermediate policies at every



Scenario Results

Fig. 8 Comparative study on a POMDP problem

time interval and at every time the policy graph reaches a certain size. Each policy is then evaluated through 1, 000 simulation runs. The expected total discounted reward of a method is computed as the average total discounted reward over all simulation runs of all the policies generated by the same method within the same interval.

Results: The results are presented in Fig. 8b–c. They indicate that EMD significantly improves placement of sampled beliefs. Nevertheless, the benefit of using EMD degrades over time. The reason is that a naive implementation of distance computation is used, checking the distance between a new sampled belief and all beliefs that are already in the belief tree. As time increases, the size of the belief tree increases, and so is the time taken for this computation. This step affects the EMD computation much more than other metrics because each EMD evaluation is more expensive. Nevertheless, there has been a lot of work on speeding up EMD computation [22, 26], which can help to alleviate this problem.

It would be interesting to see how the POMDP evaluation with EMD performs on problems with similar scale as the NOMDP scenarios. Nevertheless, existing POMDP solvers cannot solve problems with action spaces and planning horizon as large and as long as the examples used in the case of NOMDPs (i.e., beyond the 2D rigid body scenario). Nevertheless, as shown in the case of the NOMDP results, the benefit of EMD becomes more visible as the problem becomes more complex. This is likely to be true for POMDPs as well. The cost of reward computation in the POMDP test case is negligible. In more complex motion planning problems, the reward computation often includes expensive collision checks. In such problems, the ability to solve problems with smaller number of sampled beliefs, which reduces the number of backup operations (and hence reward computations), would be more beneficial. Therefore, the expectation is that EMD would show additional benefits when the POMDP problems represent more complex planning problems.

6 Discussion and Conclusion

This work demonstrates that using the Wasserstein distance in belief space planning provides significant improvements over commonly used alternatives, such as Kullback–Leibler divergence and L1 distance. This is especially apparent when planning for higher-dimensional systems. By considering an appropriate metric in belief space, it is possible to gain benefits from recent advances in sampling-based planning, which allow the computation of trajectories of increasing path quality.

With the rise of new methods for belief space planning, it is time to take a step back to understand critical components that make belief space planners perform well. This paper is a preliminary attempt to provide such an insight.

References

1. Bai, H., Hsu, D., Kochenderfer, M.J., Lee, W.S.: Unmanned aircraft collision avoidance using continuous-state POMDPs. Robot. Sci. Syst. **1**, 1–8 (2012)
2. Bai, H.Y., Hsu, D., Lee, W.S., Ngo, V.A.: Monte Carlo value iteration for continuous-state POMDPs. In: Hsu, D., et al. (eds.) WAFR. Springer, Heidelberg (2010)
3. Bry, A., Roy, N.: Rapidly-exploring random belief trees for motion planning under uncertainty. In: ICRA (2011)
4. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion. The MIT Press, Cambridge (2005)
5. Dudley, R.M.: Real Analysis and Probability. Cambridge University Press, Cambridge (2002)
6. Horowitz, M., Burdick, J.: Interactive non-prehensile manipulation for grasping via POMDPs. In: Proceedings of the IEEE International Conference on Robotics and Automation (2013)
7. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. IJRR **30**(7), 846–894 (2011)
8. Kneebone, M., Dearden, R.: Navigation planning in probabilistic roadmaps with uncertainty. In: Proceedings of the International Conference on Automated Planning and Scheduling (2009)
9. Koval, M.C., Pollard, N.S., Srinivasa, S.: Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In: RSS (2014a)
10. Koval, M., Pollard, N., Srinivasa, S.: Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. In: RSS, Berkeley, USA, July 2014b
11. Kurniawati, H., Patrikalakis. N.M.: Point-based policy transformation: adapting policy to changing POMDP models. In: WAFR (2012)
12. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: RSS (2008)
13. Kurniawati, H., Bandyopadhyay, T., Patrikalakis, N.M.: Global motion planning under uncertain motion, sensing, and environment map. Auton. Robots Spec. Issue RSS **30**(3), 2012 (2011)
14. Kurniawati, H., Du, Y., Hsu, D., Lee, W.S.: Motion planning under uncertainty for robotic tasks with long time horizons. IJRR **30**(3), 308–323 (2011)
15. Li, Y., Littlefield, Z., Bekris, K.E.: Sparse methods for efficient asymptotically optimal kinodynamic planning. In: WAFR (2014)
16. Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. In: IJRR (2015), accepted to appear
17. Littlefield, Z., Li, Y., Bekris, K.E.: Efficient sampling-based motion planning with asymptotic near-optimality guarantees for systems with dynamics. In: IROS, Tokyo Big Sight, Japan (2013)
18. Melchior, N., Simmons, R.: Particle RRT for path planning with uncertainty. In: ICRA (2007)
19. Papadimitriou, C., Tsitsiklis, J.N.: The complexity of Markov decision processes. JMOR **12**(3), 441–450 (1987)
20. Papadopoulos, G., Kurniawati, H., Patrikalakis, N.M.: Analysis of asymptotically optimal sampling-based motion planning algorithms for Lipschitz continuous dynamical systems, 12 May 2014. http://arxiv.org/abs/1405.2872
21. Paranjape, A.A., Meier, K.C., Shi, X., Chung, S.-J., Hutchinson, S.: Motion primitives and 3-D path planning for fast flight through a forest. In: IROS (2013)
22. Pele, O., Werman, M.: Fast and robust earth mover's distances. In: ICCV (2009)
23. Pineau, J., Gordon, G., Thrun, S.: Pointspsbased value iteration: an anytime algorithm for POMDPs. In: IJCAI (2003)
24. Platt, R., Kaelbling, L., Lozano-Perez, T., Tedrake, R.: Non-gaussian belief space planning: correctness and complexity. In: ICRA (2012)
25. Prentice, S., Roy, N.: The belief roadmap: efficient planning in belief space by factoring the covariance. IJRR **28**(11–12), 1448–1465 (2009)
26. Shirdhonkar, S., Jacobs, D.: Approximate earth mover's distance in linear time. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
27. Smith, T., Simmons, R.: Point-based POMDP algorithms: improved analysis and implementation. In: Proceedings of the Uncertainty in Artificial Intelligence (2005)

28. Spaan, M.T.J., Vlassis, N.: Perseus: randomized point-based value iteration for POMDPs. J. Artif. Intell. Res. **24**, 195–220 (2005)
29. Thrun, S.: Monte-Carlo POMDPs. In: NIPS (2000)
30. Urmson, C., Simmons, R.: Approaches for heuristically biasing RRT growth. In: IROS, pp. 1178–1183 (2003)
31. van den Berg, J., Abbeel, P., Goldberg, K.: LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. In: RSS (2010)
32. van den Berg, J., Patil, S., Aterovitz, R., Abbeel, P., Goldberg, K.: Planning, sensing, and control of steerable needles. In: Workshop on the Algorithmic Foundation of Robotics (2010)

Robobarista: Object Part Based Transfer of Manipulation Trajectories from Crowd-Sourcing in 3D Pointclouds

Jaeyong Sung, Seok Hyun Jin and Ashutosh Saxena

1 Introduction

Consider the espresso machine in Fig. 1 — even without having seen the machine before, a person can prepare a cup of latte by visually observing the machine and by reading a natural language instruction manual. This is possible because humans have vast prior experience of manipulating differently-shaped objects that share common parts such as ‘handles’ and ‘knobs’. In this work, our goal is to enable robots to generalize their manipulation ability to novel objects and tasks (e.g. toaster, sink, water fountain, toilet, soda dispenser). Using a large knowledge base of manipulation demonstrations, we build an algorithm that infers an appropriate manipulation trajectory given a point-cloud and natural language instructions.

The key idea in our work is that many objects designed for humans share many similarly-operated *object parts* such as ‘handles’, ‘levers’, ‘triggers’, and ‘buttons’; and manipulation motions can be transferred even among completely different objects if we represent motions with respect to *object parts*. For example, even if the robot has never seen the ‘espresso machine’ before, the robot should be able to manipulate it if it has previously seen similarly-operated parts in other objects such as ‘urinal’, ‘soda dispenser’, and ‘restroom sink’ as illustrated in Fig. 2. Object parts that are operated in similar fashion may not carry the same part name (e.g., ‘handle’) but would rather have some similarity in their shapes that allows the motion to be transferred between completely different objects.

J. Sung (✉) · S.H. Jin · A. Saxena
Department of Computer Science, Cornell University, Ithaca, NY, USA
e-mail: jysung@cs.cornell.edu

S.H. Jin
e-mail: sj372@cornell.edu

A. Saxena
e-mail: asaxena@cs.cornell.edu

© Springer International Publishing AG 2018

A. Bicchi and W. Burgard (eds.), *Robotics Research*, Springer Proceedings
in Advanced Robotics 3, DOI 10.1007/978-3-319-60916-4_40

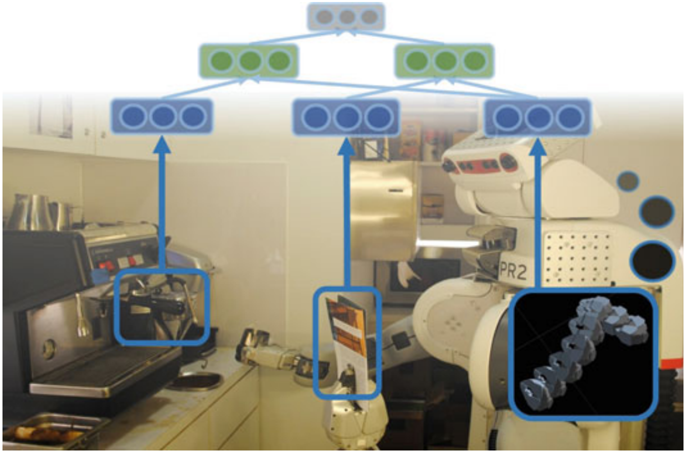


Fig. 1 First encounter of an espresso machine by our PR2 robot. Without ever having seen the machine before, given the language instructions and a point-cloud from Kinect sensor, our robot is capable of finding appropriate manipulation trajectories from prior experience using our deep learning model

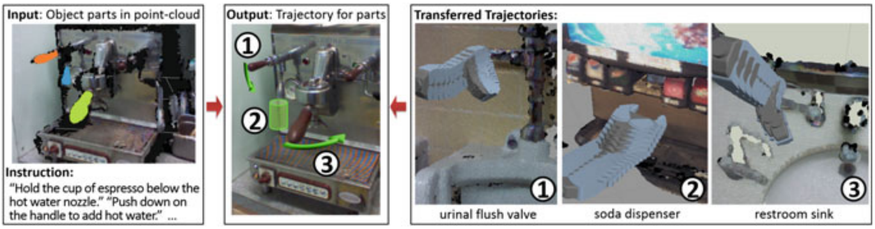

Input: Object parts in point-cloud
Instruction:
"Hold the cup of espresso below the hot water nozzle." "Push down on the handle to add hot water." ...
Output: Trajectory for parts
1
2
3
Transferred Trajectories:
1
urinal flush valve
2
soda dispenser
3
restroom sink

Fig. 2 Object part and natural language instructions input to manipulation trajectory as output. Objects such as the espresso machine consist of distinct object parts, each of which requires a distinct manipulation trajectory for manipulation. For each part of the machine, we can re-use a manipulation trajectory that was used for some other object with similar parts. So, for an object part in a point-cloud (each object part colored on *left*), we can find a trajectory used to manipulate some other object (labeled on the *right*) that can be *transferred* (labeled in the *center*). With this approach, a robot can operate a new and previously unobserved object such as the 'espresso machine', by successfully transferring trajectories from other completely different but previously observed objects. Note that the input point-cloud is very noisy and incomplete (*black* represents missing points)

If the sole task for the robot is to manipulate one specific espresso machine or just a few types of 'handles', a roboticist could manually program the exact sequence to be executed. However, in human environments, there is a large variety in the types of object and their instances. Classification of objects or object parts (e.g. 'handle') alone does not provide enough information for robots to actually manipulate them. Thus, rather than relying on scene understanding techniques [7, 17, 33], we directly use 3D point-cloud for manipulation planning using machine learning algorithms.

Such machine learning algorithms require a large dataset for training. However, collecting such large dataset of expert demonstrations is very expensive as it requires joint physical presence of the robot, an expert, and the object to be manipulated. In this work, we show that we can crowd-source the collection of manipulation demonstrations to the public over the web through our Robobarista platform and still outperform the model trained with expert demonstrations.

The key challenges in our problem are in designing features and a learning model that integrates three completely different modalities of data (point-cloud, language and trajectory), and in handling significant amount of noise in crowd-sourced manipulation demonstrations. Deep learning has made impact in related application areas (e.g., vision [5, 29], natural language processing [47]). In this work, we present a deep learning model that can handle large noise in labels, with a new architecture that learns relations between the three different modalities. Furthermore, in contrast to previous approaches based on learning from demonstration (LfD) that learn a mapping from a state to an action [4], our work complements LfD as we focus on the entire manipulation motion (as opposed to a sequential state-action mapping).

In order to validate our approach, we have collected a large dataset of *116 objects* with *250 natural language instructions* for which there are *1225 crowd-sourced manipulation trajectories* from 71 non-expert users via our Robobarista web platform (http://robobarista.cs.cornell.edu). We also present experiments on our robot using our approach. In summary, the key contributions of this work are:

- a novel approach to manipulation planning via *part-based transfer* between different objects that allows manipulation of novel objects,
- incorporation of *crowd-sourcing* to manipulation planning,
- introduction of *deep learning model* that handles three modalities with noisy labels from crowd-sourcing, and
- contribution of the first large manipulation dataset and experimental evaluation on this dataset.

2 Related Work

Scene Understanding. There has been great advancement in scene understanding [28, 33, 63], in human activity detection [21, 52], and in features for RGB-D images and point-clouds [31, 48]. And, similar to our idea of using part-based transfers, the deformable part model [17] was effective in object detection. However, classification of objects, object parts, or human activities alone does not provide enough information for a robot to reliably plan manipulation. Even a simple category such as kitchen sinks has so much variation in its instances, each differing in how it is operated: pulling the handle upwards, pushing upwards, pushing sideways, and so on. On the other hand, direct perception approach skips the intermediate object labels and directly perceives affordance based on the shape of the object [16, 30]. It focuses on detecting the part known to afford certain action such as ‘pour’ given the object, while we focus on predicting the correct motion given the object part.

Manipulation Strategy. For highly specific tasks, many works manually sequence different controllers to accomplish complicated tasks such as baking cookies [8] and folding the laundry [35], or focus on learning specific motions such as grasping [26] and opening doors [13]. Others focus on learning to sequence different movements [36, 53] but assume that there exist perfect controllers such as *grasp* and *pour*.

For a more general task of manipulating new instances of objects, previous approaches rely on finding articulation [41, 51] or using interaction [25], but they are limited by tracking performance of a vision algorithm. Many objects that humans operate daily have parts such as “knob” that are small, which leads to significant occlusion as manipulation is demonstrated. Another approach using part-based transfer between objects has been shown to be successful for grasping [10, 12]. We extend this approach and introduce a deep learning model that enables part-based transfer of *trajectories* by automatically learning relevant features. Our focus is on the generalization of manipulation trajectory via part-based transfer using point-clouds without knowing objects a priori and without assuming any of the sub-steps (‘approach’, ‘grasping’, and ‘manipulation’).

Learning from Demonstration (LfD). The most successful approach for teaching robots tasks, such as helicopter maneuvers [1] or table tennis [37], has been based on LfD [4]. Although LfD allows end users to demonstrate the task by simply taking the robot arms, it focuses on learning individual actions and separately relies on high level task composition [11, 34] or is often limited to previously seen objects [39, 40]. We believe that learning a single model for an action like “turning on” is impossible because human environment has many variations.

Unlike learning a model from demonstration, instance-based learning [2, 15] replicates one of the demonstrations. Similarly, we directly transfer one of the demonstrations but focus on generalizing manipulation planning to completely new objects, enabling robots to manipulate objects they have never seen before.

Deep Learning. There has been great success with deep learning, especially in the domains of vision and natural language processing (e.g. [29, 47]). In robotics, deep learning has previously been successfully used for detecting grasps on multi-channel input of RGB-D images [32] and for classifying terrain from long-range vision [18].

Deep learning can also solve multi-modal problems [32, 38] and structured problems [46]. Our work builds on prior works and extends neural network to handle three modalities which are of completely different data type (point-cloud, language, and trajectory) while handling lots of label-noise originating from crowd-sourcing.

Crowd-sourcing. Teaching robots how to manipulate different objects has often relied on experts [1, 4]. Among previous efforts to scale teaching to the crowd [9, 23, 54], Forbes et al. [15] employs a similar approach towards crowd-sourcing but collects multiple instances of similar table-top manipulation with same object, and others build web-based platform for crowd-sourcing manipulation [56, 57]. These approaches either depend on the presence of an expert (due to a required special software) or require a real robot at a remote location. Our Robobarista platform borrows some components of [3], but works on any standard web browser with OpenGL support and incorporates real point-clouds of various scenes.

3 Our Approach

The intuition for our approach is that many differently-shaped objects share similarly-operated object parts; thus, the manipulation trajectory of an object can be transferred to a completely different object if they share similarly-operated parts. We formulate this problem as a structured prediction problem and introduce a deep learning model that handles three modalities of data and deals with noise in crowd-sourced data. Then, we introduce the crowd-sourcing platform Robobarista to easily scale the collection of manipulation demonstrations to non-experts on the web.

3.1 Problem Formulation

The goal is to learn a function f that maps a given pair of point-cloud $p \in \mathscr{P}$ of object part and language $l \in \mathscr{L}$ to a trajectory $\tau \in \mathscr{T}$ that can manipulate the object part as described by free-form natural language l:

$$f : \mathscr{P} \times \mathscr{L} \rightarrow \mathscr{T}$$

Point-cloud Representation. Each instance of point-cloud $p \in \mathscr{P}$ is represented as a set of n points in three-dimensional Euclidean space where each point (x, y, z) is represented with its RGB color (r, g, b): $p = \{p^{(i)}\}_{i=1}^{n} = \{(x, y, z, r, g, b)^{(i)}\}_{i=1}^{n}$. The size of the set vary for each instance. These points are often obtained by stitching together a sequence of sensor data from an RGBD sensor [22].

Trajectory Representation. Each trajectory $\tau \in \mathscr{T}$ is represented as a sequence of m *waypoints*, where each waypoint consists of gripper status g, translation (t_x, t_y, t_z), and rotation (r_x, r_y, r_z, r_w) with respect to the origin: $\tau = \{\tau^{(i)}\}_{i=1}^{m} = \{(g, t_x, t_y, t_z, r_x, r_y, r_z, r_w)^{(i)}\}_{i=1}^{m}$ where $g \in \{$“open”, “closed”, “holding”$\}$. g depends on the type of the end-effector, which we have assumed to be a two-fingered gripper like that of PR2 or Baxter. The rotation is represented as quaternions (r_x, r_y, r_z, r_w) instead of the more compact Euler angles to prevent problems such as the gimbal lock [43].

Smooth Trajectory. To acquire a smooth trajectory from a waypoint-based trajectory τ, we interpolate intermediate waypoints. Translation is linearly interpolated and the quaternion is interpolated using spherical linear interpolation (Slerp) [45].

3.2 Can Transferred Trajectories Adapt Without Modification?

Even if we have a trajectory to transfer, a conceptually transferable trajectory is not necessarily directly compatible if it is represented with respect to an inconsistent reference point.

To make a trajectory compatible with a new situation without modifying the trajectory, we need a representation method for trajectories, based on point-cloud information, that allows a *direct transfer of a trajectory without any modification*.

Challenges. Making a trajectory compatible when transferred to a different object or to a different instance of the same object without modification can be challenging depending on the representation of trajectories and the variations in the location of the object, given in point-clouds.

For robots with high degrees of freedom arms such as PR2 or Baxter robots, trajectories are commonly represented as a sequence of joint angles (in configuration space) [55]. With such representation, the robot needs to modify the trajectory for an object with forward and inverse kinematics even for a small change in the object's position and orientation. Thus, trajectories in the configuration space are prone to errors as they are realigned with the object. They can be executed without modification only when the robot is in the exact same position and orientation with respect to the object.

One approach that allows execution without modification is representing trajectories with respect to the object by aligning via point-cloud registration (e.g. [15]). However, if the object is large (e.g. a stove) and has many parts (e.g. knobs and handles), then object-based representation is prone to errors when individual parts have different translation and rotation. This limits the transfers to be between different instances of the same object that is small or has a simple structure.

Lastly, it is even more challenging if two objects require similar trajectories, but have slightly different shapes. And this is made more difficult by limitations of the point-cloud data. As shown in left of Fig. 2, the point-cloud data, even when stitched from multiple angles, are very noisy compared to the RGB images.

Our Solution. Transferred trajectories become compatible across different objects when trajectories are represented (1) in the task space rather than the configuration space, and (2) in the principal-axis based coordinate frame of the object *part* rather than the robot or the object.

Trajectories can be represented in the task space by recording only the position and orientation of the end-effector. By doing so, we can focus on the actual interaction between the robot and the environment rather than the movement of the arm. It is very rare that the arm configuration affects the completion of the task as long as there is no collision. With the trajectory represented as a sequence of gripper position and orientation, the robot can find its arm configuration that is collision free with the environment using inverse kinematics.

However, representing the trajectory in task space is not enough to make transfers compatible. It has to be in a common coordinate frame regardless of object's orientation and shape. Thus, we align the negative z-axis along gravity and align the x-axis along the principal axis of the object *part* using PCA [20]. With this representation, even when the object part's position and orientation changes, the trajectory does not need to change. The underlying assumption is that similarly operated object parts share similar shapes leading to a similar direction in their principal axes.

4 Deep Learning for Manipulation Trajectory Transfer

We use deep learning to find the most appropriate trajectory for the given point-cloud and natural language. Deep learning is mostly used for binary or multi-class classification or regression problem [5] with a uni-modal input. We introduce a deep learning model that can handle three completely different modalities of point-cloud, language, and trajectory and solve a structural problem with lots of label noise.

The original structured prediction problem ($f : \mathscr{P} \times \mathscr{L} \to \mathscr{T}$) is converted to a binary classification problem ($f : (\mathscr{P} \times \mathscr{L}) \times \mathscr{T} \to \{0, 1\}$). Intuitively, the model takes the input of point-cloud, language, and trajectory and outputs whether it is a good match (label $y = 1$) or a bad match (label $y = 0$).

Model. Given an input of point-cloud, language, and trajectory, $x = ((p, l), \tau)$, as shown at the bottom of Fig. 3, the goal is to classify as either $y = 0$ or 1 at the top. The first h^1 layer learns a separate layer of features for each modality of x $(= h^0)$ [38]. The next layer learns the relations between the input (p, l) and the output τ of the original structured problem, combining two modalities at a time. The left combines point-cloud and trajectory and the right combines language and trajectory. The third

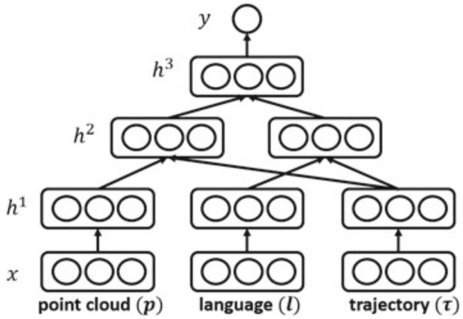


Fig. 3 Our deep learning model for transferring manipulation trajectory. Our model takes the input x of three different modalities (point-cloud, language, and trajectory) and outputs y, whether it is a good match or bad match. It first learns features separately (h^1) for each modality and then learns the relation (h^2) between input and output of the original structured problem. Finally, last hidden layer h^3 learns relations of all these modalities

layer h^3 learns the relation between these two combinations of modalities and the final layer y represents the binary label.

Every layer h^i uses the rectified linear unit [65] as the activation function:

$$h^i = a(W^i h^{i-1} + b^i) \text{ where } a(\cdot) = max(0, \cdot)$$

with weights to be learned $W^i \in \mathbb{R}^{M\times N}$, where M and N represent the number of nodes in $(i-1)$-th and i-th layer respectively. The logistic regression is used in last layer for predicting the final label y. The probability that $x = ((p, l), \tau)$ is a "good match" is computed as: $P(Y = 1|x; W, b) = 1/(1 + e^{-(Wx+b)})$.

Label Noise. When data contains lots of noisy label (noisy trajectory τ) due to crowd-sourcing, not all crowd-sourced trajectories should be trusted as equally appropriate as will be shown in Sect. 7.

For every pair of input $(p, l)_i$, we have $\mathcal{T}_i = \{\tau_{i,1}, \tau_{i,2}, ..., \tau_{i,n_i}\}$, a set of trajectories submitted by the crowd for $(p, l)_i$. First, the best candidate label $\tau_i^* \in \mathcal{T}_i$ for $(p, l)_i$ is selected as one of the labels with the smallest average trajectory distance (Sect. 5) to other labels:

$$\tau_i^* = \operatorname*{argmin}_{\tau\in\mathcal{T}_i} \frac{1}{n_i}\sum_{j=1}^{n_i} \Delta(\tau, \tau_{i,j})$$

We assume that at least half of the crowd tried to give a reasonable demonstration. Thus a demonstration with the smallest average distance to all other demonstrations must be a good demonstration.

Once we have found the most likely label τ_i^* for $(p, l)_i$, we give the label 1 ("good match") to $((p, l)_i, \tau_i^*)$, making it the first positive example for the binary classification problem. Then we find more positive examples by finding other trajectories $\tau' \in \mathcal{T}$ such that $\Delta(\tau_i^*, \tau') < t_g$ where t_g is a threshold determined by the expert. Similarly, negative examples are generated by finding trajectories $\tau' \in \mathcal{T}$ such that it is above some threshold $\Delta(\tau_i^*, \tau') > t_w$, where t_w is determined by expert, and they are given label 0 ("bad match").

Pre-training. We use the stacked sparse de-noising auto-encoder (SSDA) to train weights W^i and bias b^i for each layer [61, 65]. Training occurs layer by layer from bottom to top trying to reconstruct the previous layer using SSDA. To learn parameters for layer i, we build an auto-encoder which takes the corrupted output $\tilde{h}^{i-1}$ (binomial noise with corruption level p) of previous layer as input and minimizes the loss function [65] with max-norm constraint [49]:

$$\begin{aligned}
&W^* = \operatorname*{argmin}_{W} \|\hat{h}^{i-1} - h^{i-1}\|_2^2 + \lambda\|h^i\|_1 \\
&\text{where} \quad \hat{h}^{i-1} = f(W^i h^i + b^i) \qquad h^i = f({W^i}^T \tilde{h}^{i-1} + b^i) \qquad \tilde{h}^{i-1} = h^{i-1}X \\
&\qquad\qquad \|W^i\|_2 \le c \qquad\qquad X \sim B(1, p)
\end{aligned}$$

Fine-tuning. The pre-trained neural network can be fine-tuned by minimizing the negative log-likelihood with the stochastic gradient method with mini-batches: $NLL = -\sum_{i=0}^{|D|} log(P(Y = y^i|x^i, W, b))$. To prevent over-fitting to the training data, we used dropout [19], which randomly drops a specified percentage of the output of every layer.

Inference. Given the trained neural network, inference step finds the trajectory τ that maximizes the output through sampling in the space of trajectory $\mathscr{T}$:

$$\underset{\tau' \in \mathscr{T}}{\text{argmax}}\; P(Y = 1|x = ((p, l), \tau'); W, b)$$

Since the space of trajectory $\mathscr{T}$ is infinitely large, based on our idea that we can transfer trajectories across objects, we only search trajectories that the model has seen in training phase.

Data pre-processing. As seen in Sect. 3.1, each of the modalities (p, l, τ) can have any length. Thus, we pre-process to make each fixed in length.

We represent point-cloud p of any arbitrary length as an occupancy grid where each cell indicates whether any point lives in the space it represents. Because point-cloud p consists of only the part of an object which is limited in size, we can represent p using two occupancy grids of size $10 \times 10 \times 10$ with different scales: one with each cell representing $1 \times 1 \times 1$ (cm) and the other with each cell representing $2.5 \times 2.5 \times 2.5$ (cm).

Each language instruction is represented as a fixed-size bag-of-words representation with stop words removed. Finally, for each trajectory $\tau \in \mathscr{T}$, we first compute its smooth interpolated trajectory $\tau_s \in \mathscr{T}_s$ (Sect. 3.1), and then normalize all trajectories $\mathscr{T}_s$ to the same length while preserving the sequence of gripper status.

5 Loss Function for Manipulation Trajectory

Prior metrics for trajectories consider only their translations (e.g. [27]) and not their rotations *and* gripper status. We propose a new measure, which uses dynamic time warping, for evaluating manipulation trajectories. This measure non-linearly warps two trajectories of arbitrary lengths to produce a matching, and cumulative distance is computed as the sum of cost of all matched waypoints. The strength of this measure is that weak ordering is maintained among matched waypoints and that every waypoint contributes to the cumulative distance.

For two trajectories of arbitrary lengths, $\tau_A = \{\tau_A^{(i)}\}_{i=1}^{m_A}$ and $\tau_B = \{\tau_B^{(i)}\}_{i=1}^{m_B}$, we define matrix $D \in \mathbb{R}^{m_A \times m_B}$, where $D(i, j)$ is the cumulative distance of an optimally-warped matching between trajectories up to index i and j, respectively, of each trajectory. The first column and the first row of D is initialized as $D(i, 1) = \sum_{k=1}^{i} c(\tau_A^{(k)}, \tau_B^{(1)})\, \forall i \in [1, m_A]$ and $D(1, j) = \sum_{k=1}^{j} c(\tau_A^{(1)}, \tau_B^{(k)})\, \forall j \in [1, m_B]$, where c is a local cost function between two waypoints (discussed later). The rest of

D is completed using dynamic programming: $D(i, j) = c(\tau_A^{(i)}, \tau_B^{(j)}) + \min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\}$.

Given the constraint that $\tau_A^{(1)}$ is matched to $\tau_B^{(1)}$, the formulation ensures that every waypoint contributes to the final cumulative distance $D(m_A, m_B)$. Also, given a matched pair $(\tau_A^{(i)}, \tau_B^{(j)})$, no waypoint preceding $\tau_A^{(i)}$ is matched to a waypoint succeeding $\tau_B^{(j)}$, encoding weak ordering.

The pairwise cost function c between matched waypoints $\tau_A^{(i)}$ and $\tau_B^{(j)}$ is defined:

$$c(\tau_A^{(i)}, \tau_B^{(j)}; \alpha_T, \alpha_R, \beta, \gamma) = w(\tau_A^{(i)}; \gamma) w(\tau_B^{(j)}; \gamma) \left(\frac{d_T(\tau_A^{(i)}, \tau_B^{(j)})}{\alpha_T} + \frac{d_R(\tau_A^{(i)}, \tau_B^{(j)})}{\alpha_R} \right) \left(1 + \beta d_G(\tau_A^{(i)}, \tau_B^{(j)}) \right)$$

$$\begin{aligned} \text{where} \quad d_T(\tau_A^{(i)}, \tau_B^{(j)}) &= ||(t_x, t_y, t_z)_A^{(i)} - (t_x, t_y, t_z)_B^{(j)}||_2 \\ d_R(\tau_A^{(i)}, \tau_B^{(j)}) &= \text{angle difference between } \tau_A^{(i)} \text{ and } \tau_B^{(j)} \\ d_G(\tau_A^{(i)}, \tau_B^{(j)}) &= \mathbb{1}(g_A^{(i)} = g_B^{(j)}) \\ w(\tau^{(i)}; \gamma) &= exp(-\gamma \cdot ||\tau^{(i)}||_2) \end{aligned}$$

The parameters α, β are for scaling translation and rotation errors, and gripper status errors, respectively. γ weighs the importance of a waypoint based on its distance to the object part. Finally, as trajectories vary in length, we normalize $D(m_A, m_B)$ by the number of waypoint pairs that contribute to the cumulative sum, $|D(m_A, m_B)|_{path^*}$ (i.e. the length of the optimal warping path), giving the final form:

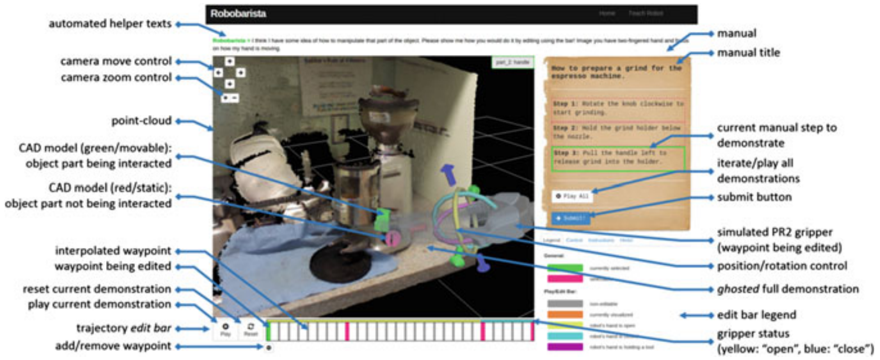
$$distance(\tau_A, \tau_B) = \frac{D(m_A, m_B)}{|D(m_A, m_B)|_{path^*}}$$

This distance function is used for noise-handling in our model and as the final evaluation metric.

6 Robobarista: Crowd-Sourcing Platform

In order to collect a large number of manipulation demonstrations from the crowd, we built a crowd-sourcing web platform that we call Robobarista (see Fig. 4). It provides a virtual environment where non-expert users can teach robots via a web browser, without expert guidance or physical presence with a robot and a target object.

The system simulates a situation where the user encounters a previously unseen target object and a natural language instruction manual for its manipulation. Within the web browser, users are shown a point-cloud in the 3-D viewer on the left and a *manual* on the right. A manual may involve several instructions, such as "Push down and pull the handle to open the door". The user's goal is to demonstrate how to manipulate the object in the scene for each instruction.



Robobarista
Home
Teach Robot
automated helper texts
camera move control
camera zoom control
point-cloud
CAD model (green/movable): object part being interacted
CAD model (red/static): object part not being interacted
interpolated waypoint
waypoint being edited
reset current demonstration
play current demonstration
trajectory edit bar
add/remove waypoint
part_2: handle
Play All
Submit!
Legend
Control
Instructions
Help
General:
currently selected
Play/Edit Bar:
Play
Reset
manual
manual title
current manual step to demonstrate
iterate/play all demonstrations
submit button
simulated PR2 gripper (waypoint being edited)
position/rotation control
ghosted full demonstration
edit bar legend
gripper status (yellow: "open", blue: "close")

Robobarista > I think I have some idea of how to manipulate that part of the object. Please show me how you would do it by editing using the bar! Image you have two-fingered hand and [illegible] on how my hand is moving.

How to prepare a grind for the espresso machine.

Step 1: Rotate the knob clockwise to start grinding.

Step 2: Hold the grind holder below the nozzle.

Step 3: Pull the handle left to release grind into the holder.

Fig. 4 Screen-shot of Robobarista, the crowd-sourcing platform running on Chrome browser. We have built Robobarista platform for collecting a large number of crowd demonstrations for teaching the robot

The user starts by selecting one of the instructions on the right to demonstrate (Fig. 4). Once selected, the target object part is highlighted and the trajectory *edit bar* appears below the 3-D viewer. Using the *edit bar*, which works like a video editor, the user can playback and edit the demonstration. Trajectory representation as a set of waypoints (Sect. 3.1) is directly shown on the *edit bar*. The bar shows not only the set of waypoints (red/green) but also the interpolated waypoints (gray). The user can click the 'play' button or hover the cursor over the edit bar to examine the current demonstration. The blurred trail of the current trajectory (*ghosted*) demonstration is also shown in the 3-D viewer to show its full expected path.

Generating a full trajectory from scratch can be difficult for non-experts. Thus, similar to Forbes et al. [15], we provide a trajectory that the system has already seen for another object as the initial starting trajectory to edit.[1]

In order to simulate a realistic experience of manipulation, instead of simply showing a static point-cloud, we have overlaid CAD models for parts such as 'handle' so that functional parts actually move as the user tries to manipulate the object.

A demonstration can be edited by: (1) modifying the position/orientation of a waypoint, (2) adding/removing a waypoint, and (3) opening/closing the gripper. Once a waypoint is selected, the PR2 gripper is shown with six directional arrows and three rings. Arrows are used to modify position while rings are used to modify the orientation. To add extra waypoints, the user can hover the cursor over an interpolated (gray) waypoint on the *edit bar* and click the plus(+) button. To remove an existing waypoint, the user can hover over it on the *edit bar* and click minus(−) to remove. As modification occurs, the edit bar and ghosted demonstration are updated with a new interpolation. Finally, for editing the status (open/close) of the gripper, the user can simply click on the gripper.

[1] We have made sure that it does not initialize with trajectories from other folds to keep *5-fold cross-validation* in experiment section valid.

For broader accessibility, all functionality of Robobarista, including 3-D viewer, is built using Javascript and WebGL.

7 Experiments

Data. In order to test our model, we have collected a dataset of 116 point-clouds of objects with 249 object parts (examples shown in Fig. 5). There are also a total of 250 natural language instructions (in 155 manuals).[2] Using the crowd-sourcing platform Robobarista, we collected 1225 trajectories for these objects from 71 non-expert users on the Amazon Mechanical Turk. After a user is shown a 20-s instructional video, the user first completes a 2-min tutorial task. At each session, the user was asked to complete 10 assignments where each consists of an object and a manual to be followed.

For each object, we took raw RGB-D images with the Microsoft Kinect sensor and stitched them using Kinect Fusion [22] to form a denser point-cloud in order to incorporate different viewpoints of objects. Objects range from kitchen appliances such as 'stove', 'toaster', and 'rice cooker' to 'urinal', 'soap dispenser', and 'sink' in restrooms. The dataset will be made available at http://robobarista.cs.cornell.edu.

Baselines. We compared our model against several baselines:

(1) *Random Transfers (chance)*: Trajectories are selected at random from the set of trajectories in the training set.

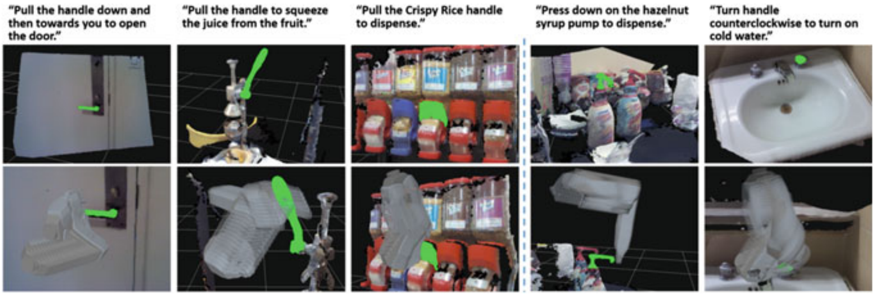

"Pull the handle down and then towards you to open the door."
"Pull the handle to squeeze the juice from the fruit."
"Pull the Crispy Rice handle to dispense."
"Press down on the hazelnut syrup pump to dispense."
"Turn handle counterclockwise to turn on cold water."

Fig. 5 Examples from our dataset, each of which consists of a natural language instruction (*top*), an object part in point-cloud representation (highlighted), and a manipulation trajectory (*below*) collected via Robobarista. Objects range from kitchen appliances such as stove and rice cooker to urinals and sinks in restrooms. As our trajectories are collected from non-experts, they vary in quality from being likely to complete the manipulation task successfully (*left* of *dashed line*) to being unlikely to do so successfully (*right* of *dashed line*)

[2] Although not necessary for training our model, we also collected trajectories from the expert for evaluation purposes.

(2) *Object Part Classifier*: To test our hypothesis that intermediate step of classifying object part does not guarantee successful transfers, we built an object part classifier using multiclass SVM [58] on point-cloud features including local shape features [28], histogram of curvatures [42], and distribution of points. Once classified, the nearest neighbor among the same object part class is selected for transfer.
(3) *Structured support vector machine (SSVM)*: It is a standard practice to hand-code features for SSVM [59], which is solved with the cutting plane method [24]. We used our loss function (Sect. 5) to train and experimented with many state-of-the-art features.
(4) *Latent Structured SVM (LSSVM) + kinematic structure*: The way an object is manipulated depends on its internal structure, whether it has a revolute, prismatic, or fixed joint. Borrowing from Sturm et al. [51], we encode joint type, center of the joint, and axis of the joint as the latent variable $h \in \mathcal{H}$ in Latent SSVM [64].
(5) *Task-Similarity Transfers + random*: It finds the most similar training task using (p, l) and transfer any one of the trajectories from the most similar task. The pair-wise similarities between the test case and every task of the training examples are computed by average mutual point-wise distance of two point-clouds after ICP [6] and similarity in bag-of-words representations of language.
(6) *Task-similarity Transfers + weighting*: The previous method is problematic when non-expert demonstrations for the same task have varying qualities. Forbes et al. [15] introduces a score function for weighting demonstrations based on weighted distance to the "seed" (expert) demonstration. Adapting to our scenario of not having any expert demonstration, we select the τ that has the lowest average distance from all other demonstrations for the same task (similar to noise handling of Sect. 4).
(7) *Our model without Multi-modal Layer*: This deep learning model concatenates all the input of three modalities and learns three hidden layers before the final layer.
(8) *Our model without Noise Handling*: Our model is trained without noise handling. All of the trajectory collected from the crowd was trusted as a ground-truth label.
(9) *Our model with Experts*: Our model is trained using trajectory demonstrations from an expert which were collected for evaluation purpose.

7.1 Results and Discussions

We evaluated all models on our dataset using *5-fold cross-validation* and the results are in Table 1. Rows list the models we tested including our model and baselines. Each column shows one of three evaluations. First two use dynamic time warping for manipulation trajectory (DTW-MT) from Sect. 5. The first column shows averaged DTW-MT for each instruction manual consisting of one or more language instructions. The second column shows averaged DTW-MT for every test pair (p, l).

As DTW-MT values are not intuitive, we added the extra column "accuracy", which shows the percentage of transferred trajectories with DTW-MT value less than 10. Through expert surveys, we found that when DTW-MT of manipulation

Table 1 Results on our dataset with *5-fold cross-validation*. Rows list models we tested including our model and baselines. And each column shows a different metric used to evaluate the models

Models	Per manual	Per instruction	
	DTW-MT	DTW-MT	Accuracy (%)
Chance	28.0 (±0.8)	27.8 (±0.6)	11.2 (±1.0)
Object part classifier	–	22.9 (±2.2)	23.3 (±5.1)
Structured SVM	21.0 (±1.6)	21.4 (±1.6)	26.9 (±2.6)
LSSVM + kinematic [51]	17.4 (±0.9)	17.5 (±1.6)	40.8 (±2.5)
Similarity + random	14.4 (±1.5)	13.5 (±1.4)	49.4 (±3.9)
Similarity + weights [15]	13.3 (±1.2)	12.5 (±1.2)	53.7 (±5.8)
Ours w/o Multi-modal	13.7 (±1.6)	13.3 (±1.6)	51.9 (±7.9)
Ours w/o noise-handling	14.0 (±2.3)	13.7 (±2.1)	49.7 (±10.0)
Ours with experts	12.5 (±1.5)	12.1 (±1.6)	53.1 (±7.6)
Our model	13.0 (±1.3)	12.2 (±1.1)	60.0 (±5.1)

trajectory is less than 10, the robot came up with a reasonable trajectory and will very likely be able to accomplish the given task.

Can manipulation trajectory be transferred from completely different objects? Our full model performed 60.0% in accuracy (Table 1), outperforming the chance as well as other baseline algorithms we tested on our dataset.

Figure 6 shows two examples of successful transfers and one unsuccessful transfer by our model. In the first example, the trajectory for pulling down on a cereal dispenser is transferred to a coffee dispenser. Because our approach to trajectory representation is based on the principal axis (Sect. 3.2), even though cereal and coffee dispenser handles are located and oriented differently, the transfer is a success. The second example shows a successful transfer from a DC power supply to a slow cooker, which have "knobs" of similar shape. The transfer was successful despite the difference in instructions ("Turn the switch.." and "Rotate the knob..") and object type.

The last example of Fig. 6 shows an unsuccessful transfer. Despite the similarity in two instructions, transfer was unsuccessful because the grinder's knob was facing towards the front and the speaker's knob was facing upwards. We fixed the z-axis along gravity because point-clouds are noisy and gravity can affect some manipulation tasks, but a more reliable method for finding the object coordinate frame and a better 3-D sensor should allow for more accurate transfers.

Does it ensure that the object is actually correctly manipulated? We do not claim that our model can find and execute manipulation trajectories for all objects. However, for a large fraction of objects which the robot has never seen before, our model outperforms other models in finding correct manipulation trajectories. The contribution of this work is in the novel approach to manipulation planning

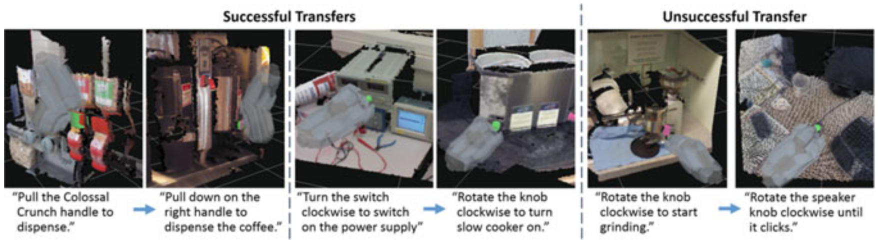

Successful Transfers
Unsuccessful Transfer
"Pull the Colossal Crunch handle to dispense."
"Pull down on the right handle to dispense the coffee."
"Turn the switch clockwise to switch on the power supply"
"Rotate the knob clockwise to turn slow cooker on."
"Rotate the knob clockwise to start grinding."
"Rotate the speaker knob clockwise until it clicks."

Fig. 6 Examples of successful and unsuccessful transfers of manipulation trajectory from *left* to *right* using our model. In first two examples, though the robot has never seen the 'coffee dispenser' and 'slow cooker' before, the robot has correctly identified that the trajectories of 'cereal dispenser' and 'DC power supply', respectively, can be used to manipulate them

which enables robots to manipulate objects they have never seen before. For some of the objects, correctly executing a transferred manipulation trajectory may require incorporating visual and force feedbacks [60, 62] in order for the execution to adapt exactly to the object as well as find a collision-free path [50].

Can we crowd-source the teaching of manipulation trajectories? When we trained our full model with expert demonstrations, which were collected for evaluation purposes, it performed at 53.1% compared to 60.0% by our model trained with crowd-sourced data. Even with the significant noise in the label as shown in last two examples of Fig. 5, we believe that our model with crowd demonstrations performed better because our model can handle noise and because deep learning benefits from having a larger amount of data. Also note that all of our crowd users are real non-expert users from Amazon Mechanical Turk.

Is segmentation required for the system? In vision community, even with the state-of-the-art techniques [14, 29], detection of 'manipulatable' object parts such as 'handle' and 'lever' in point-cloud is by itself a challenging problem [31]. Thus, we rely on human expert to pre-label parts of object to be manipulated. The point-cloud of the scene is over-segmented into thousands of supervoxels, from which the expert chooses the part of the object to be manipulated. Even with the input of the expert, segmented point-clouds are still extremely noisy because of the poor performance of the sensor on object parts with glossy surfaces.

Is intermediate object part labeling necessary? The *Object Part Classifier* performed at 23.3%, even though the multiclass SVM for finding object part label achieved over 70% accuracy in five major classes of object parts ('button', 'knob', 'handle', 'nozzle', 'lever') among 13 classes. Finding the part label is not sufficient for finding a good manipulation trajectory because of large variations. Thus, our model which does not need part labels outperforms the *Object Part Classifier*.

Can features be hand-coded? What kinds of features did the network learn? For both SSVM and LSSVM models, we experimented with several state-of-the-art features for many months, and they gave 40.8%. The *task similarity* method gave

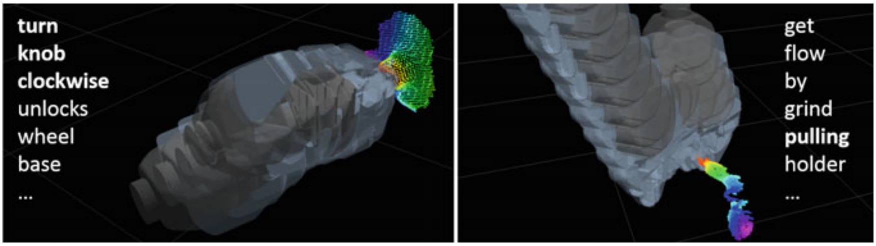


Fig. 7 Visualization of a sample of learned high-level feature (two nodes) at last hidden layer h^3. The point-cloud in the picture is given arbitrary axis-based color for visualization purpose. The *left* shows a node #1 at layer h^3 that learned to ("turn", "knob", "clockwise") along with relevant point-cloud and trajectory. The *right* shows a node #51 at layer h^3 that learned to "pull" handle. The visualization is created by selecting a set of words, a point-cloud, and a trajectory that maximize the activation at each layer and passing the highest activated set of inputs to higher level

a better result of 53.7%, but it requires access to all of the raw training data (all point-clouds and language) at test time, which leads to heavy computation at test time and requires a large storage as the size of training data increases.

While it is extremely difficult to find a good set of features for three modalities, our deep learning model which does not require hand-designing of features learned features at the top layer h^3 such as those shown in Fig. 7. The left shows a node that correctly associated point-cloud (axis-based coloring), trajectory, and language for the motion of turning a knob clockwise. The right shows a node that correctly associated for the motion of pulling the handle.

Also, as shown for two other baselines using deep learning, when modalities were simply concatenated, it gave 51.9%, and when noisy labels were not handled, it gave only 49.7%. Both results show that our model can handle noise from crowd-sourcing while learning relations between three modalities.

7.2 Robotic Experiments

As the PR2 robot stands in front of the object, the robot is given a natural language instruction and segmented point-cloud. Using our algorithm, manipulation trajectories to be transferred were found for the given point-clouds and languages. Given the trajectories which are defined as set of waypoints, the robot followed the trajectory by impedance controller (`ee_cart_imped`) [8]. Some of the examples of successful execution on PR2 robot are shown in Fig. 8 and in video at the project website: http://robobarista.cs.cornell.edu.

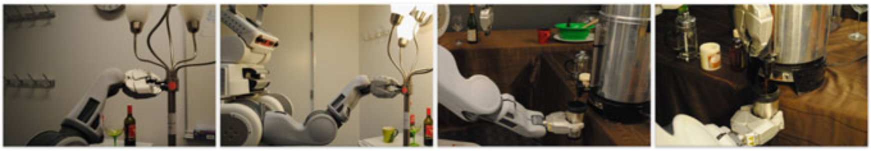


Fig. 8 Examples of transferred trajectories being executed on PR2. On the *left*, PR2 is able to rotate the ‘knob’ to turn the lamp on. On the *right*, using two transferred trajectories, PR2 is able to hold the cup below the ‘nozzle’ and press the ‘lever’ of ‘coffee dispenser’

8 Conclusion

In this work, we introduced a novel approach to predicting manipulation trajectories via part based transfer, which allowed robots to successfully manipulate objects it has never seen before. We formulated it as a structured-output problem and presented a deep learning model capable of handling three completely different modalities of point-cloud, language, and trajectory while dealing with large noise in the manipulation demonstrations. We also designed a crowd-sourcing platform Robobarista that allowed non-expert users to easily give manipulation demonstration over the web. Our deep learning model was evaluated against many baselines on a large dataset of 249 object parts with 1225 crowd-sourced demonstrations. In future work, we plan to share the learned model using the knowledge-engine, RoboBrain [44].

Acknowledgements We thank Joshua Reichler for building the initial prototype of the crowd-sourcing platform. We thank Ian Lenz and Ross Knepper for useful discussions. This research was funded in part by Microsoft Faculty Fellowship (to Saxena), NSF Career award (to Saxena) and Army Research Office.

References

1. Abbeel, P., Coates, A., Ng, A.: Autonomous helicopter aerobatics through apprenticeship learning. IJRR (2010)
2. Aha, D.W., Kibler, D.: Albert. M.K.: Instance-based learning algorithms. Mach. Learn. **6**(1), 37–66 (1991)
3. Alexander, B., Hsiao, K., Jenkins, C., Suay, B., Toris, R.: Robot web tools [ros topics]. IEEE Robot. Autom. Mag. **19**(4), 20–23 (2012)
4. Argall, B., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**(5), 469–483 (2009)
5. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Analysis Mach. Intell. **35**(8), 1798–1828 (2013)
6. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: International Society for Optics and Photonics, Robotics-DL tentative, pp. 586–606 (1992)
7. Blaschko, M., Lampert, C.: Learning to localize objects with structured output regression. In: ECCV (2008)
8. Bollini, M., Barry, J., Rus, D.: Bakebot: baking cookies with the pr2. In: IROS PR2 Workshop (2011)

9. Crick, C., Osentoski, S., Jay, G., Jenkins, O.C.: Human and robot perception in large-scale learning from demonstration. In: HRI, ACM (2011)
10. Dang, H., Allen, P.K.: Semantic grasping: planning robotic grasps functionally suitable for an object manipulation task. In: IROS (2012)
11. Daniel, C., Neumann, G., Peters, J.: Learning concurrent motor skills in versatile solution spaces. In: IROS, IEEE (2012)
12. Detry, R., Ek, C.H., Madry, M., Kragic, D.: Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In: ICRA (2013)
13. Endres, F., Trinkle, J., Burgard, W.: Learning the dynamics of doors for robotic manipulation. In: IROS (2013)
14. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI **32**(9), 1627–1645 (2010)
15. Forbes, M., Chung, M.J.-Y., Cakmak, M., Rao, R.P.: Robot programming by demonstration with crowdsourced action fixes. In: Second AAAI Conference on Human Computation and Crowd sourcing (2014)
16. Gibson, J.J.: The Ecological Approach to Visual Perception. Psychology Press, Hillsdale (1986)
17. Girshick, R., Felzenszwalb, P., McAllester, D.: Object detection with grammar models. In: NIPS (2011)
18. Hadsell, R., Erkan, A., Sermanet, P., Scoffier, M., Muller, U., LeCun, Y.: Deep belief net learning in a long-range vision system for autonomous off-road driving. In: IROS, pp. 628–633. IEEE (2008)
19. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv: 1207.0580
20. Hsiao, K., Chitta, S., Ciocarlie, M., Jones, E.: Contact-reactive grasping of objects with partial shape information. In: IROS (2010)
21. Hu, N., Lou, Z., Englebienne, G., Krse, B.: Learning to recognize human activities from soft labeled data. In: Proceedings of Robotics: Science and Systems, Berkeley, USA (2014)
22. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P. et al.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: ACM Symposium on UIST (2011)
23. Jain, A., Wojcik, B., Joachims, T., Saxena, A.: Learning preferences for manipulation tasks from online coactive feedback. Int. J. Robot. Res. **34**(10), 1296–1313 (2015)
24. Joachims, T., Finley, T., Yu, C.-N.J.: Cutting-plane training of structural svms. Mach. Learn. (2009)
25. Katz, D., Kazemi, M., Bagnell, J.A., Stentz, A.: Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In: ICRA, pp. 5003–5010. IEEE (2013)
26. Kehoe, B., Matsukawa, A., Candido, S., Kuffner, J., Goldberg, K.: Cloud-based robot grasping with the google object recognition engine. In: ICRA (2013)
27. Koppula, H., Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. In: RSS (2013)
28. Koppula, H., Anand, A., Joachims, T., Saxena, A.: Semantic labeling of 3d point clouds for indoor scenes. In: NIPS (2011)
29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
30. Kroemer, O., Ugur, E., Oztop, E., Peters, J.: A kernel-based approach to direct action perception. In: ICRA (2012)
31. Lai, K., Bo, L., Fox,D.: Unsupervised feature learning for 3d scene labeling. In: ICRA (2014)
32. Lenz, I., Lee, H., Saxena, A.: Deep learning for detecting robotic grasps. In: RSS (2013)
33. Li, L.-J., Socher, R., Fei-Fei, L.: Towards total scene understanding: classification, annotation and segmentation in an automatic framework. In: CVPR (2009)
34. Mangin, O., Oudeyer, P.-Y. et al.: Unsupervised learning of simultaneous motor primitives through imitation. In: IEEE ICDL-EPIROB (2011)

35. Miller, S., Van Den Berg, J., Fritz, M., Darrell, T., Goldberg, K., Abbeel, P.: A geometric approach to robotic laundry folding. IJRR (2012)
36. Misra, D., Sung, J., Lee, K., Saxena, A.: Tell me dave: context-sensitive grounding of natural language to mobile manipulation instructions. In: RSS (2014)
37. Mülling, K., Kober, J., Kroemer, O., Peters, J.: Learning to select and generalize striking movements in robot table tennis. IJRR **32**(3), 263–279 (2013)
38. Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A.Y.: Multimodal deep learning. In: ICML (2011)
39. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: ICRA (2009)
40. Phillips, M., Hwang, V., Chitta, S., Likhachev, M.: Learning to plan for constrained manipulation from demonstrations. In: RSS (2013)
41. Pillai, S., Walter, M., Teller, S.: Learning articulated motions from visual demonstration. In: RSS (2014)
42. Rusu, R., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: ICRA (2011)
43. Saxena, A. Driemeyer, J., Ng, A.: Learning 3-d object orientation from images. In: ICRA (2009)
44. Saxena, A., Jain, A., Sener, O., Jami, A., Misra, D.K., Koppula. H.S.: Robo brain: large-scale knowledge engine for robots. Technical report, August 2014
45. Shoemake, K.: Animating rotation with quaternion curves. SIGGRAPH **19**(3), 245–254 (1985)
46. Socher, R., Lin, C.C., Manning, C., Ng, A.Y.: Parsing natural scenes and natural language with recursive neural networks. In: ICML (2011)
47. Socher, R., Pennington, J., Huang, E., Ng, A., Manning, C.: Semi-supervised recursive autoencoders for predicting sentiment distributions. In: EMNLP (2011)
48. Socher, R., Huval, B., Bhat, B., Manning, C., Ng, A.: Convolutional-recursive deep learning for 3d object classification. In: NIPS (2012)
49. Srivastava, N.: Improving neural networks with dropout. Ph.D. thesis, University of Toronto (2013)
50. Stilman, M.: Task constrained motion planning in robot joint space. In: IROS (2007)
51. Sturm, J., Stachniss, C., Burgard, W.: A probabilistic framework for learning kinematic models of articulated objects. JAIR **41**(2), 477–526 (2011)
52. Sung, J., Ponce, C., Selman, B., Saxena, A.: Unstructured human activity detection from rgbd images. In: ICRA (2012)
53. Sung, J., Selman, B., Saxena, A.: Synthesizing manipulation sequences for under-specified tasks using unrolled markov random fields. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2014)
54. Tellex, S., Knepper, R., Li, A., Howard, T., Rus, D., Roy, N.: Asking for help using inverse semantics. RSS (2014)
55. Thrun, S., Burgard, W., Fox, D., et al.: Probabilistic Robotics. MIT press, Cambridge (2005)
56. Toris, R., Chernova, S.: Robots for me and robots for you. In: Proceedings of the Interactive Machine Learning Workshop, Intelligent User Interfaces Conference, pp. 10–12 (2013)
57. Toris, R., Kent, D., Chernova, S.: The robot management system: a framework for conducting human-robot interaction studies through crowdsourcing. J. Hum.-Robot Interact. **3**(2), 25–49 (2014)
58. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: ICML ACM (2004)
59. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y., Singer, Y.: Large margin methods for structured and interdependent output variables. JMLR, **6**(9) (2005)
60. Vina, F., Bekiroglu, Y., Smith, C., Karayiannidis, Y., Kragic, D.: Predicting slippage and learning manipulation affordances through gaussian process regression. In: Humanoids (2013)
61. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders. In: ICML (2008)
62. Wieland, S., Gonzalez-Aguirre, D., Vahrenkamp, N., Asfour, T., Dillmann, R.: Combining force and visual feedback for physical interaction tasks in humanoid robots. In: Humanoid Robots (2009)

63. Wu, C., Lenz, I., Saxena, A.: Hierarchical semantic labeling for task-relevant rgb-d perception. In: RSS (2014)
64. Yu, C.-N., Joachims, T.: Learning structural svms with latent variables. In: ICML (2009)
65. Zeiler, M.D., Ranzato, M., Monga, R. et al.: On rectified linear units for speech processing. In: ICASSP (2013)