springer tracts in advanced robotics 57

Gregory S. Chirikjian
Howie Choset
Marco Morales
Todd Murphey (Eds.)

Algorithmic Foundations of Robotics VIII

Selected Contributions of the Eighth International Workshop on the Algorithmic Foundations of Robotics

Springer Tracts in Advanced Robotics

Volume 57

Gregory S. Chirikjian, Howie Choset,
Marco Morales, Todd Murphey (Eds.)

Algorithmic Foundations of Robotics VIII

Selected Contributions of the
Eighth International Workshop
on the Algorithmic
Foundations of Robotics

Springer

Professor Bruno Siciliano, Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy, E-mail: siciliano@unina.it

Professor Oussama Khatib, Artificial Intelligence Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305-9010, USA, E-mail: khatib@cs.stanford.edu

Professor Frans Groen, Department of Computer Science, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, E-mail: groen@science.uva.nl

Editors

Gregory S. Chirikjian
Department of Mechanical Engineering
The Johns Hopkins University
Baltimore, Maryland 21218
USA
E-mail: gregc@jhu.edu

Marco Morales
Sistemas Digitales
Instituto Tecnológico
Autónomo de México
Rio Hondo 1, Progreso Tizapán
México D.F., 01080, México
E-mail: Marco.morales@itam.mx

Howie Choset
Carnegie Mellon University
5000 Forbes Ave
Newell-Simon Hall 3211
Pittsburgh PA 15213
USA
E-mail: choset@cs.cmu.edu

Todd Murphey
Department of Mechanical Engineering
Northwestern University
2145 Sheridan Road
Evanston, IL 60208
USA
E-mail: t-murphey@northwestern.edu

ISBN 978-3-642-00311-0 e-ISBN 978-3-642-00312-7

DOI 10.1007/978-3-642-00312-7

Springer Tracts in Advanced Robotics ISSN 1610-7438

Library of Congress Control Number: 2009941045

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

5 4 3 2 1 0

springer.com

STAR (Springer Tracts in Advanced Robotics) has been promoted under the auspices of EURON (European Robotics Research Network)

Foreword

By the dawn of the new millennium, robotics has undergone a major transformation in scope and dimensions. This expansion has been brought about by the maturity of the field and the advances in its related technologies. From a largely dominant industrial focus, robotics has been rapidly expanding into the challenges of the human world. The new generation of robots is expected to safely and dependably co-habitat with humans in homes, workplaces, and communities, providing support in services, entertainment, education, healthcare, manufacturing, and assistance.

Beyond its impact on physical robots, the body of knowledge robotics has produced is revealing a much wider range of applications reaching across diverse research areas and scientific disciplines, such as: biomechanics, haptics, neurosciences, virtual simulation, animation, surgery, and sensor networks among others. In return, the challenges of the new emerging areas are proving an abundant source of stimulation and insights for the field of robotics. It is indeed at the intersection of disciplines that the most striking advances happen.

The goal of the series of Springer Tracts in Advanced Robotics (STAR) is to bring, in a timely fashion, the latest advances and developments in robotics on the basis of their significance and quality. It is our hope that the wider dissemination of research developments will stimulate more exchanges and collaborations among the research community and contribute to further advancement of this rapidly growing field.

This volume is the outcome of the eight edition of the biennial Workshop Algorithmic Foundations of Robotics (WAFR). Edited by G. Chirikjian, H. Choset, M. Morales and T. Murphey, the book offers a collection of a wide range of topics in advanced robotics, including networked robots, distributed systems, manipulation, planning under uncertainty, minimalism, geometric sensing, geometric computation, stochastic planning methods, and medical applications.

The contents of the forty-two contributions represent a cross-section of the current state of research from one particular aspect: algorithms, and how they are inspired by classical disciplines, such as discrete and computational geometry, differential geometry, mechanics, optimization, operations research, computer science, probability

and statistics, and information theory. Validation of algorithms, design concepts, or techniques is the common thread running through this focused collection.

Rich by topics and authoritative contributors, WAFR culminates with this unique reference on the current developments and new directions in the field of algorithmic foundations. A very fine addition to the series!

Naples, Italy
September 2009

Bruno Siciliano
STAR Editor

Preface

Robot algorithms span areas including motion planning, dynamics, control, manipulation, state estimation, and perception. These algorithms draw from a number of sources including discrete and computational geometry, differential geometry, mechanics, optimization, operations research, theoretical computer science, probability and statistics, and information theory. Concepts from these classical disciplines are recombined in new ways and enriched when applied to problems in robotics. And broader fields of scientific inquiry (such as biomolecular modeling, rehabilitation, computer-integrated surgical systems, medical imaging, sensor networks, automated transportation systems, MEMS, and computer-aided design) all have benefitted from the application of robotics algorithms. The computations performed as a result of these robot algorithms may be implemented in software on classical or parallel computers, hardwired in electronics, encoded in neural networks, or in the future might be implemented in yet-to-be demonstrated technologies such as biomolecular or quantum computing. Therefore, research in robot algorithms is highly connected to the broader scientific community.

The Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR) brought together a group of approximately one hundred researchers from the US, Europe, and Mexico. The Eighth WAFR was held December 7-9, 2008 at the Camino Real Hotel in historic Guanajuato, México, which was founded in 1546. Forty-two refereed papers were presented, which are reflected in this volume. Three keynote lectures were given by: Kazuhiro Saitou, U. of Michigan; José Luis Marroqun, CIMAT, and Joel Burdick, Caltech. Topics of papers can be roughly classified into the categories of networked robots, distributed systems, manipulation, planning under uncertainty, minimalism, geometric sensing, geometric computation, stochastic planning methods, and medical applications.

In addition to the organizers, the program committee consisted of the following people who contributed reviews and advertised the workshop: Devin Balkcom, Alejandra Barrera, Timothy H. Chung, Noah Cowan, Magnus Egerstedt, Claudia Esteves, Robert Ghrist, Bill Goodwine, Dan Halperin, Seth Hutchinson, David Hsu, Lydia Kavraki, Sven Koenig, Vijay Kumar, Steve LaValle, Kevin Lynch, Mark Moll, Rafael Murrieta, Jim Ostrowski, Mark Overmars, Daniela Rus, Gildardo

Sánchez-Ante, Nicola Simeon, Sidd Srinivasa, Enrique Sucar, Jeff Trinkle, Yunfeng Wang, Alfredo Weitzenfeld, Jing Xiao, Dianna Xu.

We are in debt with many individuals and organizations whose help and dedicated work made this meeting possible. We give our thanks to: the student volunteers from ITAM, CIMAT, Tec. de Monterrey campus Guadalajara, and UAEM for their hard work; our institutions (CMU, ITAM, Johns Hopkins, Colorado) for their support; Sara Ridel, Carmen Rosas, and Leticia Vitela from Viajes Polanco for their patience and diligence to help the meeting run smoothly; Sen. Francisco Arroyo Vieyra for promoting support from the government of Guanajuato; Sergio Rodríguez and the members of Secretaría de Desarrollo Turístico del Estado de Guanajuato for their assistance with the logistics and for sponsoring the transportation to the banquet; the personnel from Hotel Camino Real who made us feel at home; and the WAFR steering committee for their thoughtful advice in preparation for the meeting.

September 2009

Gregory S. Chirikjian
Howie Choset
Marco Morales
Todd Murphey

Acknowledgements

We are in debt with many individuals and organizations whose help and dedicated work made this meeting possible. We give our thanks to: the student volunteers from ITAM, CIMAT, Tec. de Monterrey campus Guadalajara, and UAEM for their hard work; our institutions (CMU, ITAM, Johns Hopkins, Colorado) for their support; Sara Ridel, Carmen Rosas, and Leticia Vitela from Viajes Polanco for their patience and diligence to help the meeting run smoothly; Sen. Francisco Arroyo Vieyra for promoting support from the government of Guanajuato; Sergio Rodríguez and the members of Secretaría de Desarrollo Turístico del Estado de Guanajuato for their assistance with the logistics and for sponsoring the transportation to the banquet; the personnel from Hotel Camino Real who made us feel at home; and the WAFR steering committee for their thoughtful advice in preparation for the meeting.

Contents

Part VII: Geometric Computations

Part VIII: Stochastic Methods in Planning

Part IX: Medical Applications

Part X: Planning

List of Contributors

Alberto Rodriguez
The Robotics Institute,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA 15213, USA
albertor@cmu.edu

Alex Brooks
Australian Centre for
Field Robotics,
University of Sydney,
NSW 2006, Australia
a.brooks@acfr.usyd.edu.au

Alon Efrat
Department of
Computer Science,
The University of Arizona,
alon@cs.arizona.edu

Anastasios I. Mourikis
University of California,
Riverside, CA
92521, USA
mourikis@ee.ucr.edu

Ankur Dhanik
Mechanical Engineering
Department, Stanford
University, Stanford, CA
94305, USA
ankurd@stanford.edu

Anna Yershova
Dept. of Computer Science,
Duke University,
Durham, NC
27707, USA
yershova@cs.duke.edu

Ashley Deacon
Joint Center
for Structural Genomics,
SLAC, Menlo Park,
CA 94025, USA
vdbedem@slac.stanford.edu

Avik De
Johns Hopkins
University, ME dept.,
Baltimore, MD
21218, USA
jsl@jhu.edu

Benjamin Tovar
Department of
Computer Science,
University of Illinois.
Urbana, IL
61801 USA
btovar@uiuc.edu

Bill Goodwine
University of Notre Dame,
365 Fitzpatrick Hall,
Notre Dame,
IN 46556, USA
jgoodwin@nd.edu

Bruce R. Donald
Department of Computer Science,
Duke University, Durham,
NC, USA
Department of Biochemistry,
Duke University Medical Center,
Durham, NC, USA
brd+wafr08@cs.duke.edu

Burchan Bayazit
Washington University
in St. Louis, 1 Brookings
Dr., St. Louis, MO 63130, USA
bayazit@cse.wustl.edu

Christopher G. Levey
Thayer School of Engineering,
Dartmouth College,
Hanover, NH, USA

Claudia Esteves
Facultad de Matemáticas
de la Universidad
de Guanajuato,
Guanajuato, México
cesteves@cimat.mx

Dan Halperin
School of Comp. Science,
Tel-Aviv University,
Tel Aviv 69978, Israel
danha@post.tau.ac.il

Dania El-Khechen
Concordia University,
Montreal, Canada
d_elkhec@cs.concordia.ca

Daniela Rus
The Stata Center,
Building 32,
32 Vassar Street,
Cambridge, MA
02139, USA
rus@csail.mit.edu

Daniela Rus
Department of
Electrical Engineering
and Computer Science,
MIT, Cambridge,
MA, USA

Daniela Rus
Massachusetts Institute
of Technology,
32 Vassar Street,
Cambridge, MA
02139, USA
rus@csail.mit.edu

David Gifford
Massachusetts Institute
of Technology,
32 Vassar Street,
Cambridge, MA
02139, USA
gifford@csail.mit.edu

David Hsu
National University
of Singapore,
Singapore 117417,
Singapore
dyhsu@comp.nus.edu.sg

Dezhen Song
Computer Science and
Engineering Department,
Texas A&M University,
311C HRBB,
TAMU 3112,
College Station,
TX 77843, USA
dzsong@cse.tamu.edu

Dinesh Manocha
Department of Computer Science,
University of North Carolina
at Chapel Hill, NC 27599
dm@cs.unc.edu

Dinesh Manocha
Department of Computer Science,
University of North Carolina at
Chapel Hill,
dm@cs.unc.edu

Efi Fogel
School of Comp. Science,
Tel-Aviv University,
Tel Aviv 69978, Israel
efif@post.tau.ac.il

Ehsan Rehman
National University of Singapore,
Singapore 117417, Singapore
ehsanreh@comp.nus.edu.sg

Elliot R. Johnson
Department of
Mechanical Engineering,
Northwestern University
elliot.r.johnson@
u.northwestern.edu

Emilio Frazzoli
Laboratory for Information
and Decision Systems,
Massachusetts Institute
of Technology, Cambridge,
MA 01239, USA
frazzoli@mit.edu

Eric Meisner
Department of Computer Science,
Rensselaer Polytechnic Institute,
110 8th Street, Troy, NY 12180
meisne@cs.rpi.edu

Erik D. Demaine
Massachusetts Institute
of Technology,
Cambridge, USA Partially
supported by NSF CAREER
award CCF-0347776,
DOE grant
DE-FG02-04ER25647,
and AFOSR grant
FA9550-07-1-0538
edemaine@mit.edu

Erik Halvorson
Duke University, Box 90129,
Durham, NC 27708-0129
erikh@cs.duke.edu

Fred Cohen
Department of Mathematics,
University of Rochester.
Rochester, NY
14627 USA
fcohen@rochester.edu

Greg Aloupis
Université Libre de Bruxelles,
Belgique
greg.aloupis@ulb.ac.be

Gregory S. Chirikjian
Department of
Mechanical Engineering,
Johns Hopkins University,
Baltimore, MD
21218, USA
gregc@jhu.edu

Guilherme Pereira
Departamento de
Engenharia Elétrica,
Universidade Federal
de Minas Gerais, Av. Antônio
Carlos 6627, Belo Horizonte,
MG 31270-010, Brazil
gpereira@cpdee.ufmg.br

Hai Yu
Google Inc.
New York, NY
fishhai@google.com

Hanna Kurniawati
National University
of Singapore, Singapore 117417,
Singapore
hannakur@comp.nus.edu.sg

Hari Balakrishnan
Massachusetts Institute
of Technology, 32 Vassar Street,
Cambridge, MA 02139, USA
hari@csail.mit.edu

Henry van den Bedem
Joint Center for
Structural Genomics,
SLAC, Menlo
Park, CA 94025, USA
vdbedem@slac.stanford.edu

Hyunnam Lee
Samsung Techwin Robot
Business T/F,
KIPS Center 647-9,
Yeoksam-Dong,
Gangnam-Gu,
Seoul, South Korea
hyunnamlee@gmail.com

Igor Paprotny
Department of Computer Science,
Dartmouth College,
Hanover, NH, USA

Ioan A. Şucan
Rice University,
6100 Main St.,
Houston, TX 77005, USA
isucan@rice.edu

James C. Davidson
Beckman Institute,
University of Illinois,
405 North Mathews Avenue,
Urbana, IL 61801 USA
jcdavdsn@illinois.edu

James Kuffner
School of Computer Science,
Carnegie Mellon University
kuffner@cs.cmu.edu

Jason M. O'Kane
Department of Computer
Science and Engineering,
University of South Carolina,
315 Main Street, Columbia, SC
29208, USA
jokane@cse.sc.edu

Jason Nightingale
University of Notre Dame,
365 Fitzpatrick Hall,
Notre Dame,
IN 46556, USA
jnightin@nd.edu

Jean-Bernard Hayet
Centro de Investigación
en Matemáticas,
CIMAT, Guanajuato, México
jbhayet@cimat.mx

Jean Claude Latombe
Computer Science Department,
Stanford University, Stanford,
CA 94305, USA
ankurd@stanford.edu

Jean-Claude Latombe
Department of
Computer Science,
Stanford University,
Stanford, CA 94305, USA
latombe@cs.stanford.edu

Jean-Paul Laumond
LAAS-CNRS,
University of Toulouse,
7 avenue du Colonel Roche,
31077 Toulouse
Cedex 4, France
jpl@laas.fr

Jijie Xu
Ph.D. Program,
GCCIS, Rochester
Institute of Technology,
NY, USA
jijie.xu@rit.edu

Jing Xiao
University of North
Carolina - Charlotte,
401 Woodward Hall,
9201 University City Blvd,
Charlotte, NC 28223, USA
xiao@uncc.edu

Jingang Yi
Department of Mechanical
and Aerospace Engineering,
Rutgers, The State
University of New Jersey,
Piscataway, NJ
08854-8058, USA
jgyi@rutgers.edu

Joel A. Hesch
University of Minnesota,
Minneapolis, MN 55455, USA
joel@cs.umn.edu

Jonathan Fink
Grasp Laboratory,
University of Pennsylvania,
3330 Walnut Street Philadelphia,
PA 19104-6228 USA
jonfink@grasp.upenn.edu

Joseph O'Rourke
Smith College,
Northampton, USA
orourke@cs.smith.edu

Julie C. Mitchell
Dept. of Mathematics,
University of
Wisconsin-Madison, Madison,
WI 53706,
mitchell@math.wisc.edu

Jur van den Berg
Department of Computer Science,
University of North
Carolina at Chapel Hill,
berg@cs.unc.edu

Jyh-Ming Lien
George Mason University,
4400 University Dr,
Fairfax, VA 22030, USA
jmlien@cs.gmu.edu

Jusuk Lee
Johns Hopkins University,
ME dept., Baltimore, MD
21218, USA
jsl@jhu.edu

Kaijen Hsiao
Computer Science and
Artificial Intelligence Laboratory,
Massachusetts Institute
of Technology,
32 Vassar St., Cambridge,
MA 02139 USA
kjhsiao@mit.edu

Karen Morris
Northwestern University
Interdepartmental
Neuroscience program,
Northwestern University
??@u.northwestern.edu

Ken Goldberg
Dept. of EECS and IEOR,
University of California,
Berkeley, CA, USA
goldberg@berkeley.edu

Ketan Savla
Laboratory for Information
and Decision Systems,
Massachusetts Institute
of Technology,
Cambridge, MA 01239, USA
ksavla@mit.edu

Kris Hauser
Department of
Computer Science,
Stanford University,
Stanford, CA
94305, USA
khauser@cs.stanford.edu

Leonidas J. Guibas
Stanford University,
Stanford, CA
94305, USA
guibas@cs.stanford.edu

Leslie Pack Kaelbling
Computer Science and
Artificial Intelligence Laboratory,
Massachusetts Institute
of Technology,
32 Vassar St.,
Cambridge, MA
02139, USA
kjhsiao@mit.edu

Liangjun Zhang
Department of
Computer Science,
University of
North Carolina at
Chapel Hill, NC 27599
zlj@cs.unc.edu

Lingzhi Luo
Robotics Institute,
Carnegie Mellon
University, Pittsburgh,
PA 15213, USA
lingzhil@cs.cmu.edu

Luciano Pimenta
GRASP Laboratory,
Levine Hall L402,
University of Pennsylvania,
3330 Walnut Street,
Philadelphia, PA
19104-6228, USA
pimenta@seas.upenn.edu

Lydia E. Kavraki
Rice University,
6100 Main St.,
Houston, TX 77005, USA
kavraki@rice.edu

Mac Schwager
The Stata Center,
Building 32, 32 Vassar Street,
Cambridge, MA 02139, USA
schwager@mit.edu

Martin Burger
Institute for
Computational and
Applied Mathematics,
Westfaelische Wilhelms
Universitaet Muenster
martin.burger@wwu.de

Matthew Mason
School of Computer Science,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA
mason+@cs.cmu.edu

Matthew T. Mason
The Robotics Institute,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA 15213, USA
matt.mason@cs.cmu.edu

Michael Erdmann
School of Comptuer Science,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA
me@cs.cmu.edu

Michael Erdmann
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA, 15213, USA
me@cs.cmu.edu

Mike Stilman
School of Interactive Computing,

Georgia Tech
mstilman@cc.gatech.edu

Ming Lin
Department of Computer Science, University of North Carolina at Chapel Hill,
lin@cs.unc.edu

Mirela Damian
Villanova University, Villanova, USA
mirela.damian@villanova.edu

Nancy M. Amato
Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843 USA
amato@tamu.edu

Nicholas Keller
Johns Hopkins University, ME dept., Baltimore, MD 21218, USA
jsl@jhu.edu

Nicolay M. Tanushev
University of Texas at Austin
nicktan@math.utexas.edu

Noah J. Cowan
Johns Hopkins University, ME dept., Baltimore, MD 21218, USA
jsl@jhu.edu

Nuzhet Atay
Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130, USA
atay@cse.wustl.edu

Pankaj K. Agarwal
Department of Computer Science, Duke University,
pankaj@cs.duke.edu

Peng Cheng
The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA
pcheng@mathworks.com

Quentin Lindsey
GRASP Laboratory, Levine Hall L402, University of Pennsylvania, 3330 Walnut Street, Philadelphia, PA 19104-6228, USA
quentinl@seas.upenn.edu

Rafael Murrieta-Cid
Centro de Investigación en Matemáticas, CIMAT, Guanajuato México
murrieta@cimat.mx

Rayomand Vatcha
University of North Carolina - Charlotte, 401 Woodward Hall, 9201 University City Blvd, Charlotte, NC 28223, USA
rvatcha@uncc.edu

Renato Mesquita
Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais,

Av. Antônio
Carlos 6627,
Belo Horizonte,
MG 31270-010, Brazil
renato@cpdee.ufmg.br

Richard Hind
University of Notre Dame,
255 Hurley Hall, Notre Dame,
IN 46556, USA
rhind@nd.edu

Richard Tsai
University of
Texas at Austin
ytsai@math.utexas.edu

Robin Flatland
Siena College,
Loudonville, N.Y., USA
flatland@siena.edu

Ron Alterovitz
Dept. of CS,
University of North
Carolina at Chapel Hill,
NC, USA
ron@cs.unc.edu

Ronald Parr
Duke University,
Box 90129, Durham,
NC 27708-0129
parr@cs.duke.edu

Samuel Madden
Massachusetts Institute
of Technology,
32 Vassar Street,
Cambridge, MA
02139, USA
madden@csail.mit.edu

Sébastien Collette
Université Libre
de Bruxelles, Belgique,
Chargé de Recherches
du FNRS.
secollet@ulb.ac.be

Sébastien Dalibard
LAAS-CNRS,
University of Toulouse,
7 avenue du Colonel Roche,
31077 Toulouse Cedex 4, France
sdalibar@laas.fr

Sejoon Lim
Massachusetts Institute
of Technology,
32 Vassar Street,
Cambridge, MA
02139, USA
sjlim@csail.mit.edu

Seth A. Hutchinson
Beckman Institute,
University of Illinois,
405 North Mathews Avenue,
Urbana, IL 61801 USA
seth@illinois.edu

Seth Hutchinson
Department of Electrical
and Computer Engineering,
University of
Illinois at Urbana-Champaign
seth@illinois.edu

Shankar Sastry
Dept. of EECS,
University of California,
Berkeley, CA, USA
sastry@coe.berkeley.edu

R. Sharathkumar
Department of
Computer Science,
Duke University,
sharath@cs.duke.edu

Shawna Thomas
Department of
Computer Science
and Engineering,
Texas A&M University,
College Station, TX 77843 USA
sthomas@cse.tamu.edu

Soonkyum Kim
Grasp Laboratory,
University of Pennsylvania,
3330 Walnut Street Philadelphia,
PA 19104-6228 USA
soonkyum@grasp.upenn.edu

Sourabh Bhattacharya
Department of Electrical
and Computer Engineering,
University of Illinois
at Urbana-Champaign
sbhattac@illinois.edu

Srinivas Akella
Department of Computer Science,
University of North
Carolina at Charlotte, Charlotte,
NC 28223, USA
sakella@uncc.edu

Stefan Langerman
Université Libre de
Bruxelles, Belgique,
Maître de Recherches
du FNRS
slanger@ulb.ac.be

Stefanie Wuhrer
Carleton University,
Ottawa, Canada
swuhrer@scs.carleton.ca

Stergios I. Roumeliotis
University of Minnesota,
Minneapolis, MN
55455, USA
stergios@cs.umn.edu

Steven M. LaValle
Dept. of Computer Science,
University of Illinois,
Urbana, IL 61801 USA
lavalle@cs.uiuc.edu

Steven LaValle
Department of
Computer Science,
University of Illinois.
Urbana, IL 61801 USA
lavalle@uiuc.edu

Suneeta Ramaswami
Rutgers University,
Camden, NJ, USA
rsuneeta@
camden.rutgers.edu

Time Bailey
Australian Centre for
Field Robotics,
University of Sydney,
NSW 2006, Australia
a.brooks@acfr.usyd.edu.au

Timothy H. Chung
Department of
Operations Research,
Naval Postgraduate School,
Monterey, CA
93943, USA
thchung@nps.edu

Todd D. Murphey
Department of
Mechanical Engineering,
Northwestern University
t-murphey@northwestern.edu

Tomás Lozano-Pérez
Computer Science and
Artificial Intelligence Laboratory,
Massachusetts Institute
of Technology, 32 Vassar St.,
Cambridge, MA 02139, USA
kjhsiao@mit.edu

Val Pinciu
Southern Connecticut
State University, USA
pinciu@scsu.ctstateu.edu

Vera Sacristán
Universitat Politècnica
de Catalunya,
Barcelona, Spain
Partially supported
by projects MEC
MTM2006-01267
and DURSI 2005SGR00692
vera.sacristan@upc.edu

Vijay Kumar
GRASP Laboratory,
Levine Hall L402,
University of Pennsylvania,
3330 Walnut Street,
Philadelphia, PA
19104-6228, USA
kumar@grasp.upenn.edu

Vijay Kumar
Grasp Laboratory,
University of Pennsylvania,
3330 Walnut Street Philadelphia,
PA 19104-6228 USA
kumar@grasp.upenn.edu

Vincent Duindam
Dept. of EECS,
University of California,
Berkeley, CA, USA
v.duindam@ieee.org

Volkan Isler
Department of Computer Science,
University of Minnesota,
200 Union Street SE,
Minneapolis, MN 55455
isler@cs.umn.edu

Wei Yang
Department of Computer Science,
Rensselaer Polytechnic Institute,
110 8th Street, Troy, NY 12180
yangw@cs.rpi.edu

Wooram Park
Department of
Mechanical Engineering,
Johns Hopkins University,
Baltimore, MD
21218, USA
wpark7@jhu.edu

Xinyu Tang
Google, 1600 Amphitheatre
Parkway, Mountain View,
CA 94043 USA
xtang@google.com

Yan-Bin Jia
Department of Computer Science,
Iowa State University,
Ames, IA 50011, USA
jia@cs.iastate.edu

Yanina Landa
University of California,
Los Angeles
ylanda@math.ucla.edu

Yunfeng Wang
Department of
Mechanical Engineering,
The College of
New Jersey,
Ewing, NJ 08628 USA
jwang@tcnj.edu

Part I
Networks

Probabilistic Network Formation through Coverage and Freeze-Tag

Eric Meisner, Wei Yang, and Volkan Isler

Abstract. We address the problem of propagating a piece of information among robots scattered in an environment. Initially, a single robot has the information. This robot searches for other robots to pass it along. When a robot is discovered, it can participate in the process by searching for other robots. Since our motivation for studying this problem is to form an ad-hoc network, we call it the *Network Formation Problem.* In this paper, we study the case where the environment is a rectangle and the robots' locations are unknown but chosen uniformly at random. We present an efficient network formation algorithm, Stripes, and show that its expected performance is within a logarithmic factor of the optimal performance. We also compare Stripes with an intuitive network formation algorithm in simulations. The feasibility of Stripes is demonstrated with a proof-of-concept implementation.

1 Introduction

Consider the following scenario: a number of robots are performing independent tasks autonomously in an environment where there is no communication infrastructure. Suppose, at a certain point, mission priorities change and a piece of information must be propagated to all nodes. For example, robots could be initially stationed to perform monitoring or surveillance in a large environment. Upon detection of an event, they may have to form a network or perform a collaborative task. What is a good strategy to get all robots involved as quickly as possible?

Eric Meisner and Wei Yang
Department of Computer Science, Rensselaer Polytechnic Institute 110 8th Street
Troy NY 12180
e-mail: {meisne,yangw}@cs.rpi.edu

Volkan Isler
Department of Computer Science, University of Minnesota 200 Union Street
SE Minneapolis, MN 55455
e-mail: isler@cs.umn.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 3–17.
springerlink.com

In this paper, we study this process of propagating information as quickly as possible. Specifically, we study the case where the process is initiated by a single robot. This robot could, for example, be sent out from the command and control center. Alternatively, it could be the robot that detects an intruder. The primary difficulty in solving the problem arises from the fact that the robots do not know each others' positions. The first robot must therefore start a search. Once discovered, other robots can participate in propagating the information. Since our primary motivation for studying this problem is to form a connected network, throughout the paper we will refer to it as the *Network Formation Problem*.

Our Contributions: In this paper, we study a probabilistic scenario where the locations of robots are chosen uniformly at random in a rectangular environment. For this scenario, we present a network formation strategy (Stripes) and prove that its expected performance (i.e. network formation time) is within a logarithmic factor of the optimal performance. To obtain this result, we also obtain a lower bound on the expected performance of *any* network formation strategy. We also compare Stripes with a natural, intuitive "Split and Cover" strategy and show that in large environments, Split and Cover has inferior performance to Stripes. In addition to formal performance bounds, we demonstrate the utility of Stripes with simulations and its feasibility with a proof-of-concept implementation. In the next section, we start with an overview of related work where we establish connections between the network formation problem and other fundamental problems such as rendezvous, coverage and freeze-tag.

1.1 Related Work

Considerable work has been done in designing decentralized protocols for propagating information in networks (also known as gossip protocols) [7]. Gossip protocols are mainly designed for stationary networks. In contrast, we focus on information propagation among mobile robots. The network formation problem is closely related to the Freeze-Tag problem [4]. In freeze-tag, a number of players are "frozen". A single unfrozen player must visit each frozen player in order to unfreeze it, at which point it can aid in unfreezing other players. In freeze tag, it is assumed that the players know each others' positions. In network formation, we focus on the case where the node locations are unknown to each other.

Recently, Poduri and Sukhatme explicitly addressed the problem of forming a connected network under the name *coalescence* [11, 12]. In their model, all of the nodes (other than the base station) are performing a random walk on a torus. The authors obtain bounds on the network formation time. The advantage of the random-walk strategy is that it does not require localization with respect to a global reference frame. However, the network formation is rather slow because nodes visit most locations many times. This may not be acceptable in time-critical applications. In the present work, we address the problem of explicitly designing motion strategies for network formation with guaranteed performance. Since we focus on the case where

the robot locations are unknown, the network formation problem is related to rendezvous and coverage.

The rendezvous search problem [1] asks how two players with the same speed can locate each other when placed at arbitrary locations in a known environment. Typically, each player has a radius of detection within which the players can establish communication. The goal of the players is to find each other as quickly as possible. The rendezvous problem has been studied extensively for two players on a line. Optimal or provably good strategies for various versions of this problem have been established [2].

The problem of multi-player rendezvous search on a complete graph is studied in [14]. This work addresses the question of whether players that meet should stick together or split and meet again later. The result of this work shows that, if the players have no memory, then the optimal strategy is to stick together. Also, simulations show that as the number of players increases, the split and meet strategy becomes less effective. In general, this work has limited applications because the environment is a complete graph. In other words, players can teleport to arbitrary locations at every time step.

Work in [3] describes a time optimal strategy for two-player limited visibility rendezvous in the plane with known and unknown distances. The optimal solution in this case is for one of the players to follow a semi-circular spiral trajectory. This work also relates rendezvous in the continuous domain to coverage problems. Deterministic and randomized multi-robot rendezvous strategies were proposed in [13]. More recently, results on rendezvous in simply-connected polygons have been obtained [8]. In [9], the authors provide upper and lower bounds on the time complexity of two geometric laws that result in rendezvous.

The coverage or lawn mowing problem [5, 6] has been extensively studied and is known to be closely related to Traveling Salesperson Problem. The primary difference between network formation and standard multi-robot coverage is that in coverage, all robots participate in the coverage process from the beginning. In network formation, the process starts with one robot, who must "recruit" others to participate in coverage.

In short, the algorithm presented in this paper can be considered a novel algorithm for (i) online freeze-tag, (ii) probabilistic multi-robot coverage, and (iii) network formation.

2 Problem Formulation

Since the robot locations are unknown, network formation is an online problem. Online problems are typically analyzed using competitive analysis where the input is chosen by an adversary. The competitive ratio of an online algorithm $\mathcal{O}$ is the worst case ratio of the performance of $\mathcal{O}$ to the optimal offline performance. In other words, we compare $\mathcal{O}$ with the optimal algorithm that has access to all of $\mathcal{O}$'s input in advance and consider the worst case deviation from this optimal behavior.

It is easy to see that there is no online algorithm for the network formation problem with bounded competitive ratio: no matter which path the first robot chooses, the adversary can place all other robots at the last location the first robot will visit. In the case of a square environment with area a^2, the online algorithm would take time proportional to a^2 whereas the optimal offline cost would be at most in the order of a. Therefore the competitive ratio would be a and it would grow unboundedly with the size of the environment.

Since there are no competitive online algorithms for network formation, in this paper we focus on the probabilistic case where the locations of the robots are sampled from a distribution. In the absence of any information, it is reasonable to assume that the locations are chosen uniformly at random from the environment. *The goal is to minimize the expected time to discover all robots.* In this paper, we focus on rectangular environments and uniform distributions. We believe that the rectangular environment case has practical relevance for robots operating in an open field as well as for Unmanned Aerial Vehicles. We discuss extensions to general convex environments in Section 6.

2.1 Robot Model

In network formation, several identical mobile robots are distributed uniformly at random within the bounded rectangle A with one of these robots possessing information that needs to be propagated to all of the other robots. The rectangle A has a width w and height h where $w \geq h$. We assume that the robots are initially stationary. Once discovered, they can move anywhere within the rectangle A at a constant speed and can communicate with each other within a limited range. Once a robot is within the communication range of another robot, they can exchange any information that either robot might have, including the information that needs to be propagated. In this paper, we assume that the robots can localize themselves within A. We also ignore energy limitations and assume that the robots will always move at their maximum speed and can utilize their wireless radio anytime.

Notation and Conventions: We normalize the units so that the robots move at unit speed per time-unit. In the rest of the paper, we use A to denote both the rectangle environment and its area. This convention implies that a single robot can cover the entire environment in A time units. Hence, A is a trivial upper bound on network formation since inactive nodes are stationary. The number of robots in the environment is k. We assume that the first robot enters the environment at a corner of A.

3 Stripes: An Efficient Network Formation Strategy

In this section, we present our main result: an efficient network formation strategy which we refer to as "Stripes". This strategy relies on dividing the rectangular environment into n equal sized vertical stripes $S_1, \ldots, S_n$ (Figure 1). For now, we treat n

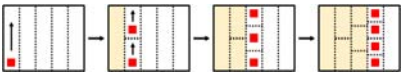


Fig. 1. Stripes strategy: The environment is divided into equal vertical stripes which are covered sequentially. Active robots split the current stripe equally. Once a stripe is covered, all active robots (including newly discovered robots) meet at the boundary of the stripe.

as a variable whose value will be fixed later. Let S denote the area of a single stripe which is equal to the time to cover a single stripe with one robot.

In the beginning, a single robot is active and proceeds to cover S_1 with a simple back and forth sweeping motion. That is, the robot follows the well-known boustrophedon[1] path. When an inactive robot is encountered and activated, this newly active robot does not join in the coverage of S_1 right away[2]. Instead, it heads to a designated location on the line that separates S_1 and S_2 and waits for the first active robot to finish its coverage of S_1 and arrive at the designated location. When the first robot is finished, it meets all newly active robots at the same designated location. Let k_1 denote the number of active robots. The robots evenly divide S_2 among themselves so that it can be covered in parallel in time S_2/k_1.

When these k_1 robots encounter other inactive robots in S_2, the same procedure is repeated and all active robots meet at a designated meeting location on the line which separates S_2 and S_3. This process repeats for the remaining stripes until all of the stripes are covered, at which point, the entire bounded area will be covered and all of the robots will be activated.

For most of the paper, we will focus on rectangular environments which are large with respect to the number of robots. In other words, we focus on the case where $w \geq h \gg k$.

In the remaining case, the number of robots exceeds the size of the shorter side of the rectangle (i.e. $k \geq h$). We refer to this case as the *saturated case*. When the number of active robots reach h, the remaining $m \times h$ area can be covered in m steps by the active robots. We present the analysis of the algorithm for the saturated case in the appendix.

3.1 Network Formation with Stripes

In this section, we establish an upper bound on the network formation time of Stripes for an unsaturated environment. We will use the following lemma.

Lemma 3.1. *If k balls are assigned to n bins randomly, the expected number of empty bins is at most $ne^{-k/n}$.*

Proof. We say that bin i is "hit" if it is non-empty after the k^{th} throw. The probability p_i that the i^{th} bin is not hit is $(1-\frac{1}{n})^k$. Let X_i be a random variable which takes the

[1] Boustrophedon = "the way of the ox" [6].

[2] In practice, this robot can participate in covering the stripe. However, this does not improve the analysis. Hence, we ignore this benefit for analysis purposes.

value one if bin i is not hit and zero otherwise. Then, $E[X_i] = p_i$. The number of empty bins is given by the random variable $\sum_{i=1}^{n} X_i$ whose expected value is np_i. The lemma follows from the inequality $(1+u) \leq e^u$.

To obtain an upper bound on the network formation time with Stripes, we treat the robots as balls and stripes as bins. Note that the coverage time is maximized when the empty stripes occur at the beginning, and non-empty stripes have the following property: let m be the number of non-empty stripes. $m-1$ stripes have only one robot inside and all the remaining robots are in a single stripe which occurs after all other stripes. We compute the expected coverage time for this worst case.

If we pick $n = \frac{k}{\log k}$, by Lemma 3.1 the number of empty stripes is at most $\frac{1}{\log k}$, which is less than or equal to 1 for $k > 1$.

This results in a coverage time of $\frac{A}{n}$ for the first non-empty stripe, $\frac{A}{n}\left(\frac{1}{2}\right)$ for the second, $\frac{A}{n}\left(\frac{1}{3}\right)$ and so on for the remaining stripes. Since there is at most one non-empty stripe, the total coverage time will be bounded by:

$$\frac{A}{n} + \sum_{i=1}^{n} \frac{A}{n}\left(\frac{1}{i}\right) = \frac{A}{n} + \frac{A}{n}\sum_{i=1}^{n} \frac{1}{i} \approx \frac{A}{n} + \frac{A}{n}\log n \tag{1}$$

By plugging in $n = \frac{k}{\log k}$, we have the following result.

Lemma 3.2. *The expected network formation time for k robots in a rectangular unsaturated environment with area A is $O(\frac{A}{k}\log^2 k)$ when robots execute the Stripes algorithm with $n = \frac{k}{\log k}$ stripes.*

Since robots group together before and after covering a stripe, there is some overhead associated with the different strategies that can be used to cover an individual stripe. In establishing Lemma 3.2, this overhead is ignored. We justify this in the next section, where we present the strategy to cover individual stripes.

3.2 Covering a Single Stripe

The Stripes algorithm calls for the set of active robots to meet and regroup between covering stripes. In this section, we present a strategy for multiple robots to cover a single stripe in a way that minimizes the overhead of meeting to redistribute assignments. Our proposed single stripe strategy ensures that each robot travels the same distance during an epoch, and arrives at a common meeting location.

In order to minimize the coverage time of a single stripe, its area must be split equally between all of the active robots. This can be done by simply dividing the strip along the long side to create equal sized rectangular areas, one for each of the active robots. However, this will result in some robots traveling longer distances to reach their assigned coverage objectives. Since this area will eventually be covered by the other robots, this produces an overhead of $2h$ per stripe, adding up to $2nh$ for the entire bounded area A. Our single stripe strategy divides the active stripe into sections such that, 1) the area covered by each robot is equal and 2) the time

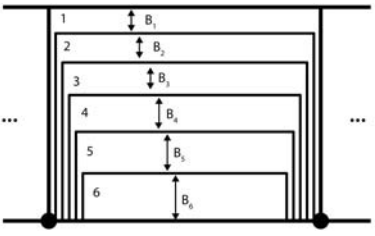


Fig. 2. Our single stripe coverage strategy divides a vertical stripe into equally sized areas that take into account any additional area covered by a robot as it travels away from the meeting locations (large black dots). This figure shows the resulting divisions for six active robots and the designated B_i values used to reference the height of the horizontal rectangular areas.

required to cover each section and then reach the meeting location is equal. In the end, the paths of the robots resemble archways with different sized horizontal areas that are inversely proportional to the distance a robot needs to travel to the meeting location (Figure 2). The "legs" of the archways represent the areas covered by each robot as they travel up to their assigned rectangular areas.

Given a particular vertical stripe to cover with r active robots, let $A_1, A_2, \ldots, A_r$ denote the total assigned coverage area for each robot, with robot i covering A_i and robot $i = 1$ covering the area that is furthest away from the meeting location. Since each stripe is vertical, its area is $h\frac{w}{n} = hw'$. We use B_i to denote the distance between the upper boundaries of each successive region (see Figure 2). This parameter can be varied to offset the additional cost of traveling. Each robot is assumed to cover one unit area per time step so the additional travel area covered by r_1 is simply twice the distance from the meeting point to its assigned coverage area or $2(h - B_1)$. This produces the following equation for calculating the area A_i of each region (equation 5):

$$A_1 = B_1 w' + 2(h - B_1) = B_1(w' - 2) + 2h \tag{2}$$

$$A_2 = B_2(w' - 2) + 2(h - B_1 - B_2) = B_2(w' - 4) + 2h - 2B_1 \tag{3}$$

$$\begin{aligned} A_3 &= B_3(w' - 4) + 2(h - B_1 - B_2 - B_3) \\ &= B_3(w' - 6) + 2h - 2B_1 = 2B_2 \end{aligned} \tag{4}$$

$$A_i = B_i(w' - 2(i-1)) + 2(h - \sum_{j=1}^{i} B_j) \tag{5}$$

Since each robot must cover the same area, we can determine the relationship between B_i and B_{i-1} by setting the generic area formula at i and $i - 1$ equal to each other (Equation 6). The remaining A_i and B_i values can be determined by successively applying Equations 7 and 8. In Equation 7, A_1 is equated to $\frac{hw'}{r}$, producing equation 8 where B_1 is expressed using the number of active robots, r, the width of a stripe $w' = \frac{w}{n}$, and the height of the overall environment, h.

$$\frac{B_i}{B_{i-1}} = 1 + \frac{4}{w' - 2i} \tag{6}$$

$$A_1 = \frac{hw'}{r} = B_1(w' - 2) + 2h \tag{7}$$

$$B_1 = \frac{\frac{hw'}{r} - 2h}{w' - 2} = \frac{h(w' - 2r)}{r(w' - 2)} \tag{8}$$

The limitation of this single stripe strategy is that the number of active robots must be less than half of a single stripe's width, $r < \frac{w'}{2}$. This does not present a problem for the analyzed environment, where it is assumed that $w = h >> k$. Therefore, from the single stripe strategy, each stripe can be divided into r equally sized areas with minimal repeated coverage and thus, eliminating the overhead coverage time from the Stripes equation.

3.3 Lower Bound

In this section, we establish a lower bound on the expected network formation time that can be achieved by any algorithm. It is easy to see that the best coverage time for any area occurs when all of the robots are active at the beginning and the area is evenly split between all of them. Therefore, the absolute lower bound for total coverage time, regardless of algorithm, is $T(A,k) = \frac{A}{k}$. For the case when the robots are uniformly distributed, we can obtain a better bound using the following result.

Lemma 3.3 ([10], pp.45). *When n balls are assigned uniformly at random to n bins, with probability at least* $1 - \frac{1}{n}$*, no bin has more than* $\alpha = \frac{\log n}{\log \log n}$ *balls in it.*

Now consider any network formation strategy for k robots in area A. During the execution of this strategy, we divide the coverage process into epochs where i^{th} epoch ends when all active robots cover a (previously uncovered) total area of A/k. Let S_i denote the subset of A covered during epoch i. Let E_1 be the event that no S_i has more that α balls. By Lemma 3.3, E_1 happens with probability $(1 - 1/k)$.

When E_1 happens, the maximum number of robots in S_1 is α. Therefore, the minimum time it takes to cover S_1 is $T_1 = \frac{A}{k\alpha}$. There will be α new robots in S_2, therefore its coverage time T_2 is at least $\frac{A}{k2\alpha}$. Similarly, the i^{th} epoch will last at least $T_i = \frac{A}{ki\alpha}$ steps. Since there are k epochs, the total time for all epochs to finish will be:

$$\sum_{i=1}^{k} \frac{A}{ki\alpha} \approx \frac{A}{\alpha k} \log k = \frac{A}{k} \log \log k \tag{9}$$

When E_1 does not happen (with probability $\frac{1}{k}$), the coverage time is at least A/k as discussed earlier. This gives us the lower bound on the expected coverage time of any algorithm.

Lemma 3.4. *The expected time for k robots to cover rectangular area A is at least* $(1-\frac{1}{k})\frac{A}{k}\log\log k+(\frac{1}{k})\frac{A}{k}$.

Ignoring $o(\frac{1}{k^2})$ terms, we establish a lower bound of $\Omega(\frac{A}{k}\log\log k)$. Using Lemmata 3.2 and 3.4, we establish our main result:

Theorem 3.1. *The performance of the Stripes strategy is within a factor* $O(\frac{\log^2 k}{\log\log k})$ *of the optimal performance for a rectangular environment where* $w \geq h \gg k$.

3.4 Split and Cover

In this section, we discuss the performance of the following intuitive and natural network formation strategy: The first robot moves up and down the environment following a boustrophedon path as before. When it meets an undiscovered node, the two nodes split the undiscovered parts of the environment evenly and recursively cover their assigned partitions (see Figure 3). The justification for this natural "Split-and-Cover" strategy is that since the nodes are scattered uniformly in the environment, each split should divide the load equally between discovered robots, therefore balancing (and intuitively minimizing) the network formation time.

It turns out that Split-and-Cover is not an effective strategy for large environments for the following reason: suppose that when the split happens, one of the partitions has a very small number of robots. Even though this event has a small probability for a single split, as the number of robots (and hence the number of splits) increases, it becomes probable. When such an imbalance occurs, the algorithm can not recover from it. A small number of robots must cover a large environment making Split-and-Cover inefficient. We will further justify this argument with simulations and show that the Stripes algorithm is much more efficient than Split-and-Cover in unsaturated environments.

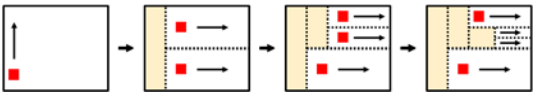


Fig. 3. Split and Cover Strategy

4 Simulations

In this section, we compare Stripes and Split-and-Cover in simulations. We also present simulation results that demonstrate the effect of the number of Stripes on the performance of the Stripes algorithm. Our simulations were performed by representing each of the individual robots as a point within the rectangular world. Each robot can be assigned to sweep along a continuous piecewise linear path. As described in the Section 3, an active robot can detect an inactive robot when it is

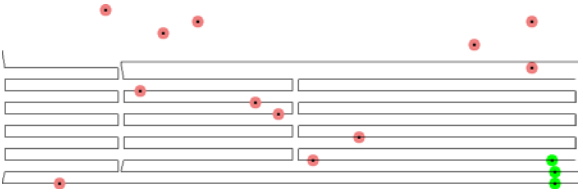


Fig. 4. This figure shows the simulation of the Stripes algorithm. Green circles are active robots, and red circles are inactive. The black lines represent the paths of active robots within the current stripe.

within communication range. Each robot moves one unit distance per unit time and maintains an internal clock, to represent the time with respect to the start of the first robot. Robots that meet can synchronize clocks. We place the robots uniformly at random within the environment at the start of each trial. The simulation runs the algorithm exactly as it is stated in Section 3.4.

4.1 Results

In the first simulation, we compare Stripes and Split-and-Cover in environments that are sparse and dense with respect to the number of robots per area. Figure 5a shows the results of the Split-and-Cover and Stripes algorithms in an environment where the concentration of robots is low. The simulations use a 1000x1000 environment and 20 robots. The plotted values are the average time to completion of 1000 trials, and the value for each number of stripes uses the same set of initial robot distributions. From these simulations, we conclude that (i) the performance of the Stripes algorithm is sensitive to the number of stripes, and (ii) in sparse environments (with a "good" stripe-number selection) Stripes outperforms Split-and-Cover. This is also justified by Figures 6a and 6b, where we plot the histogram of running times of the Stripes algorithm (with 25 Stripes) and Split-and-Cover. Stripes not only outperforms Split-and-Cover on the average but also its worst case performance is considerably better.

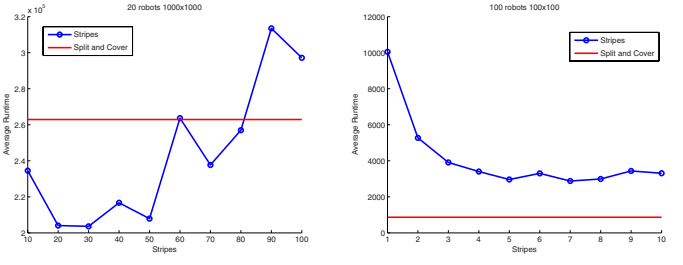
On the other hand, when the concentration of robots is high, Split-and-Cover outperforms Stripes (Figures 5b,6c,6d). There are two reasons that this is true. First, for Split-and-Cover, the unbalanced split described in the previous section occurs infrequently when the concentration of robots is high. Second, for Stripes the overhead cost of meeting up to evenly distribute the coverage of a stripe becomes very large compared to the discovery time. In most instances, saturation took place after the fist stripe due to the high concentration of the robots. Therefore, the running time of Stripes was often given by the time to discover the first stripe followed by a parallel scan with 7 robots. In conclusion, simulation results suggest that Stripes should be used in sparse environments with the number of

stripes equal to the number of robots. In dense environments, split-and-cover algorithm is expected to yield better performance.

5 Experiments

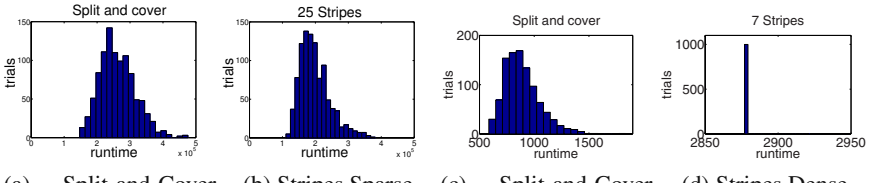
We demonstrate the feasibility of the Stripes algorithm using a small team of three Acroname Garcia robots shown in Figure 8c. The robots are each equipped with ARM/risk PCs, and wireless network adapters. When configured to work in ad-hoc mode, the robots form a wireless sensor network, each capable of determining its own set of neighbors in the network graph.

Currently, our robots do not have visual localization capabilities. Therefore, we rely on motor encoders and dead reckoning for position information. We also utilize an external stereo camera as a means of determining the robot positions off-line (to obtain ground truth). The external camera allows us to compute robot positions, but also limits the size of the work area to the field of view of the camera. Our experiments take place in a square workspace of 7.5 meters by 7.5 meters. These experiments where conducted inside of the institute gymnasium.



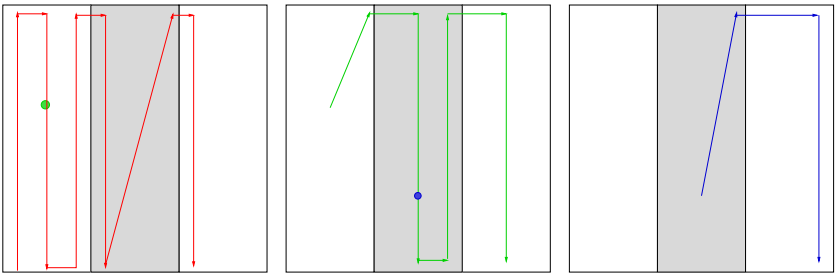
(a) Sparse environment (b) Dense environment

Fig. 5. Comparison of Split-and-Cover strategy with Stripes algorithm for varying number of stripes. The plotted values are the average time to completion of 1000 trials. **Left:** 1000×1000 unit world with 20 robots. **Right:** 100×100 unit world with 100 robots.



(a) Split-and-Cover Sparse (b) Stripes Sparse (c) Split-and-Cover Dense (d) Stripes Dense

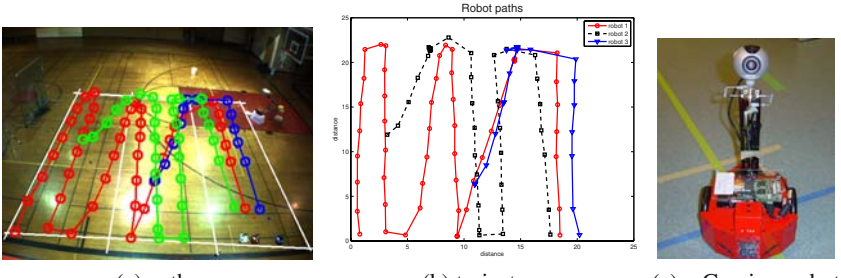
Fig. 6. Left: Distribution of time to completion for Split-and-Cover(6a), and for Stripes(6b) in a sparse environment. **Right:** Distribution of time to completion for Split-and-Cover(6c), and for Stripes(6d) in a dense environment.



(a) Robot 1 (b) Robot 2 (c) Robot 3

Fig. 7. The ideal trajectories for robots 1, 2 and 3 (7a, 7b, and 7c respectively). The stripes are denoted by alternating white and gray. The first robot starts at the lower left corner. The circle in Figure 7a is the meeting location with robot 2. The third robot is discovered by robot 2. The meeting location is shown in Figure 7b

In practice, it would be useful to use wireless connectivity and connection strength to determine when an inactive node has been detected. However, in this setup the nodes would have to be separated by very large distances before observing a noticeable decline in signal strength. Therefore, in order to demonstrate the algorithm in our restricted space, we simulate restrictions in communication by allowing inactive nodes to be discovered only when they are within a short range. Figure 7 shows the placement of the robots and their ideal strategies when executing Stripes. Figure 8a shows an image of the experimental setup from the external camera. The white lines superimposed on the image outline the workspace and stripe boundaries. Figure 8b shows actual robot trajectories during the experiment computed from the image to ground plane homography.



(a) paths (b) trajectory (c) Garcia robot platform

Fig. 8. 8a This image of the experimental setup shows the environment divided into three stripes. The paths traversed by each robot are shown in red, green and blue (lines superimposed). 8b This image shows the actual trajectory of each robot computed using the homography between the image and ground planes.

5.1 Results

For this small number of robots, we use a simpler single-stripe strategy. In the experiment, each stripe is selected so that it can be covered by a single robot in 24 time units. This corresponds to three up and down motions (Figure 7). The initial placement of the robots is shown by the start of each of the three paths in Figure 8a. Robots move approximately 1 meter per time unit. The total area of the workspace is approximately 56 m^2. The stripes are completed in 24, 16, and 8 time units respectively. Hence, the total completion time is 48 units. Note that the total coverage time for a single robot is 72 time units. The measured path lengths of each robot are 41.08, 25.848, and 13.26 meters respectively. In this setup, even though odometry errors resulted in deviations from the ideal trajectories in Figure 7, they were not significant enough to prevent proper execution of Stripes.

6 Extension to General Convex Environments

As mentioned earlier, the time to cover the environment is a trivial upper bound on the network formation time. This is because the undiscovered nodes are stationary, and by covering the environment, the first robot can guarantee that all nodes are discovered. In a convex environment, this coverage time is proportional to the area of the environment A. In the case of an unsaturated rectangular environment, we showed that the Stripes algorithm performs better than this upper bound.

In general environments, even when the environment is convex, the upper bound would be achieved. To see this, consider a long $1 \times A$ environment. In this environment, even when the robot locations are chosen randomly, the network formation time would be roughly A because the information is propagated sequentially and one of the robots is expected to be close to the "other" end of the environment. Therefore, the upper bound of A can not be beaten in some general convex environments.

7 Conclusion and Future Work

In this paper, we introduced a novel network formation problem with ties to freeze-tag and coverage problems. In the network formation problem, a robot tries to propagate a piece of information to other robots with unknown locations as quickly as possible. Once a robot is discovered, it joins in the information propagation process by searching for other robots.

We presented an algorithm for rectangular environments and analytically proved that its performance is within a logarithmic factor of the optimal performance. We demonstrated the utility of the algorithm further with simulations and a proof-of-concept implementation.

In our future work, we will address network formation in general environments. We will start with general convex environments. We believe that the Stripes

algorithm can be modified to achieve good performance in such scenarios (compared to the optimal performance). Arbitrary environments, represented as polygons with holes, seem to be more challenging due to the lack of a natural way of partitioning the environment. We plan to study this interesting and equally challenging problem in our future work.

Acknowledgements. This work is supported in part by NSF CCF-0634823 and NSF CNS-0707939, NSF IIS-07455373.

References

1. Alpern, S.: The rendezvous search problem. SIAM J. Control Optim. 33(3), 673–683 (1995)
2. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Springer, Heidelberg (2002)
3. Anderson, E.J., Fekete, S.P.: Asymmetric rendezvous on the plane. In: SCG 1998: Proceedings of the fourteenth annual symposium on Computational geometry, pp. 365–373. ACM Press, New York (1998)
4. Arkin, E., Bender, M., Fekete, S., Mitchell, J., Skutella, M.: The freeze-tag problem: How to wake up a swarm of robots (2002)
5. Arkin, E.M., Fekete, S.P., Mitchell, J.S.B.: Approximation algorithms for lawn mowing and milling. Computational Geometry 17(1-2), 25–50 (2000)
6. Choset, H.: Coverage of known spaces: The boustrophedon cellular decomposition. Auton. Robots 9(3), 247–253 (2000)
7. Deb, S., Médard, M., Choute, C.: Algebraic gossip: a network coding approach to optimal multiple rumor mongering. IEEE/ACM Trans. Netw. 14(SI), 2486–2507 (2006)
8. Ganguli, A., Cortes, J., Bullo, F.: Multirobot rendezvous with visibility sensors in nonconvex environments. IEEE Transactions on Robotics (November 2006) (to appear)
9. Martinez, S., Bullo, F., Cortes, J., Frazzoli, E.: On synchronous robotic networks - Part II: Time complexity of rendezvous and deployment algorithms. IEEE Transactions on Automatic Control (2007) (to appear)
10. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge Press (2000)
11. Poduri, S., Sukhatme, G.S.: Achieving connectivity through coalescence in mobile robot networks. In: International Conference on Robot Communication and Coordination (October 2007)
12. Poduri, S., Sukhatme, G.S.: Latency analysis of coalescence in robot groups. In: IEEE International Conference on Robotics and Automation (2007)
13. Roy, N., Dudek, G.: Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. Auton. Robots 11(2), 117–136 (2001)
14. Thomas, L., Pikounis, M.: Many-player rendevous search: stick together or split and meet? Naval Research Logistics 48, 710–721 (2001)

Appendix: Saturated Environments

In a saturated environment where $k \geq h$, the performance of Stripes can be improved with a simple modification after the number of active robots reach the height of

the environment. The main difference between the coverage time for a saturated environment and an unsaturated environment is how the $k \geq h$ constraint affects the coverage of a single stripe.

In a saturated environment, after the number of active robots in a stripe, r, exceeds the height of a stripe, it is no longer possible to divide a stripe into r equal, non-overlapping regions. However, it is possible to line up the active robots along the height of the environment. Afterwards, they can cover the remaining portion of the environment without meeting again. Therefore, when $r \geq h$, the coverage time for a stripe becomes $\frac{w}{n}$ where n is the number of stripes. We will call any stripe where this occurs as saturated.

This changes the original upper bound equation (Equation 1) to contain an additional term. For the saturated environment, the term in the logarithm will instead be the number of non-empty stripes until stripes become saturated or simply h. The remaining area $A' = A(1 - \frac{h}{n})$ can be covered in time $\frac{A'}{h}$. Therefore, the coverage time of an $A = h \times w$ environment is bounded by:

$$\left(\frac{A}{n} + \frac{A}{n}\log h\right) + w - \frac{A}{n} = \frac{A}{n}\log h + w \tag{10}$$

Planning Aims for a Network of Horizontal and Overhead Sensors

Erik Halvorson and Ronald Parr

Abstract. This paper considers the problem of planning sensor observations for a network of overhead sensors which will resolve ambiguities in the output of a horizontal sensor network. Specifically, we address the problem of counting the number of objects detected by the horizontal sensor network, using the overhead network to aim at specific areas to improve the count. The main theme of our results is that, even though observation planning is intractable for such a network, a simple, greedy algorithm for controlling the overhead sensors guarantees performance with bounded and reasonable suboptimality. Our results are general and make few assumptions about the specific sensors used. The techniques described in this paper can be used to plan sensor aims for a wide variety of sensor types and counting problems.

1 Introduction

The problems of sensor placement and observation planning have become increasingly relevant as sensor networks increase in both capability and complexity. Often, however, sensor placement and planning problems lead to instances of classical planning problems or partially observable Markov decision processes, both of which are intractable in general. Although there exist algorithms which give optimal solutions to these problems, their potentially enormous computational makes them undesirable.

Consider a horizontal network of sensors with the goal of counting the number of distinct objects it detects. Due to occlusion, the sensor network may not be able to sense all the objects and thus it may not be able to determine the exact count. This paper considers an observation planning problem where the goal is to plan the aims of a set of overhead sensors to resolve these ambiguities. The overhead sensors are used to resolve specific portions in the region of interest where the count is

Erik Halvorson and Ronald Parr
Department of Computer Science, Duke University
e-mail: {erikh,parr}@cs.duke.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 19–34.
springerlink.com

ambiguous. An example of such a network would be a set of horizontal, fixed position cameras, with pan-tilt cameras mounted on unmanned aerial vehicles (UAVs) providing the overhead sensors.

Counting the number of objects within a region is a basic problem in the field of surveillance. Once determined, the number of objects has many potential uses, such as counting people moving across a border, identifying vehicle movements, or providing an accurate count of the people attending an outdoor gathering. Traditional (non computer-based) methods typically rely on manual head counting and would not work in these situations. We consider the problem of developing an accurate count with no human involvement.

Depending on the different kinds of sensors in the network, there are a wide variety of ways to count the distinct objects. This paper will use a geometric approach to counting, inspired by the previous work of Yang, et al. [10], which used a visual hull to determine upper and lower bounds on the number of people in a scene viewed by horizontal cameras. Though the visual hull is typically associated with cameras, the concept generalizes to other sensor types which can detect occupancy. We also note that the counting method described by Yang, et al. is not specific to counting people and can be used to count any objects detectable by the sensor network.

The work of Yang et al. assumes that objects move, which helps reduce ambiguities as the patterns of occlusion change over time. Even if the objects are in motion, however, the gap between these bounds may not converge to zero or, depending upon the speed at which the objects are moving, may not converge at an acceptable rate. We consider the use of overhead sensors to supplement the horizontal network. Such sensors can provide a faster and more accurate count when object motion alone is not sufficient. Overhead sensors, like those found on aircraft, can be redirected in seconds, which makes it safe to assume that in many cases, several iterations of aiming and retargeting of the overhead sensors will be possible before the scene has changed significantly from the perspective of the horizontal based sensors.

We propose using a simple, greedy algorithm to aim the overhead sensors. Our analysis first bounds the suboptimality over a single set of aims, which we refer to as a *phase*. Since most scenes will require multiple phases, the next portion of our analysis extends these results to multiple phases, bounding the number required relative to an optimal algorithm. We also show that computing an optimal multiphase plan is intractable, and show that two closely related problems are intractable: the subproblem of orienting the overhead sensors to maximize the number of viewed potential objects, and computing the smallest number of objects consistent with a set of observations by the horizontal network.

2 Previous Work

One common approach to the counting problem involves tracking. Multi-target tracking algorithms generally either assume a known, fixed number of targets, or attempt to solve the counting problem while simultaneously tracking the targets. Many approaches to the latter problem attempt to model the arrival and departure

of new targets, generally when an unrecognized object is detected [9, 11]. Several appearance-based tracking algorithms have been specifically applied to the problem of counting people [6, 7]. Accurately determining whether a target has been previously detected, however, is non-trivial and error-prone. Counting is itself an interesting problem because it could be used to initialize many multi-target tracking algorithms.

Observation planning approaches to tracking generally assume a known number of targets; He and Chong [4], for example, formulate the tracking problem as a POMDP and use an approximate solution based on sampling. Guestrin, et al. [3] develop a greedy approach for the sensor placement problem and bound its suboptimality.

One very different approach to counting uses a geometric construction called a *visual hull*, which is defined as the intersection of all the silhouette cones seen from each sensor. A *silhouette cone* is a projection of a sensor detection into a conical region in front of the sensor. For example, in the case of a camera that has detected a change in the scene that spans several pixels, the corresponding silhouette cone would be a cone extending from the lens into the world that covers all points in the world which project onto the effected pixels. It is possible to reconstruct the geometry of one or more objects by considering the geometry of the silhouette cones as seen from several sensors. Though originally developed for this purpose, visual hulls have been shown to be useful for counting the number of distinct objects detected by a sensor network [10]. The original concept was developed by Laurentini [5], who designed algorithms for constructing the visual hull in both two and three dimensions. In this paper, we compute a planar projection of the visual hull; this projection results in a number of polygons lying in the plane. Yang, et al, [10] use these polygons to give lower and upper bounds on the number of objects which create a visual hull, and rely on the objects moving to reduce the gap in bounds. Even with considerable movement, however, this gap can remain quite large.

3 Static Bound Calculation

We begin by formalizing the concept of a visual hull. We assume that the horizontal sensors in the network are capable of detecting objects and creating *silhouette cones* (see previous section) where these objects are detected. The sensors are not, however, capable of differentiating distinct objects, so objects lying in the same cone (i.e. those seen as either fully or partially occluded) do not generate additional silhouette cones; rather, these additional detections appear to be a single occupied region. See Figure 1 (top left) for an example. Given a horizontal network of such sensors, each viewing the same scene from different angles, the silhouette cones can be combined into a *visual hull*:

Definition 3.1. A planar projection of a *visual hull* is a set of polygons P lying at the intersections of the silhouette cones from each sensor.

Assuming that the entire region of interest is covered by at least two horizontal sensors, all objects in the scene must be located within polygons, although not all

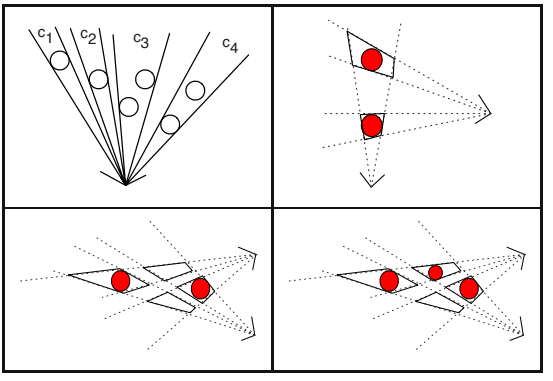


Fig. 1. (top left) Example silhouette cones. c_1 and c_2 both contain exactly one object, c_3 contains two objects where the object farther from the sensor is viewed partially occluded, and the rear object is fully occluded in c_4. (top right) A visual hull with two objects. The dashed lines are the silhouette cones, while the solid lines represent the polygons in P. (bottom) Two identical visual hulls created by different numbers of objects. By convention, we show only the sensors with detections. Empty areas are presumed clear of objects because other sensors (not shown) that cover area of interest did not detect anything.

polygons necessarily contain objects, as shown in figure 1 (bottom). Note that much of the plane is not in P because at least one of the sensors failed to detect an object in these locations and the location is thus outside the intersection of the cones. We also assume that the region of interest is bounded by walls which limit the sensor's detection range.

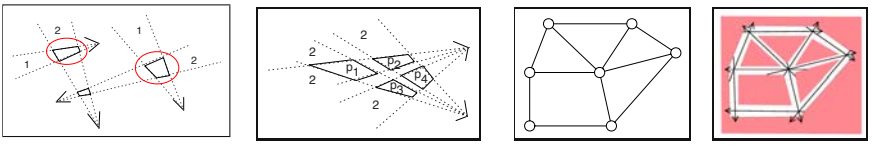
The visual hull can be created by any sensor capable of creating silhouette cones where objects are detected. For example, the cones could be created by applying background subtraction techniques with a camera. The set of polygons, along with the silhouette cones, can be used to develop bounds on the number of objects seen by the network [10]. We first formalize a simple lower bound.

Definition 3.2. The *cone upper bound* of a cone c, $cub(c)$, is the number of polygons contained in c.

Definition 3.3. A polygon, generated by intersecting set of cones C, is *provably occupied* if $\exists c \in C$ with $cub(c) = 1$

For example, in Figure 2 (a), there are two polygons provably occupied (circled). The cone counts are also given. The middle polygon is not provably occupied, since both the cones containing it have a *cub* of 2. Figure 2 (b) has no provably occupied polygons, as all cones have a cone upper bound of two.

The number of provably occupied polygons is a *lower bound* on the number of objects contained by the visual hull. This is a more rigorous definition of the lower bound presented by Yang, et al. This definition of the lower bound is weak in the sense that the minimum number of objects consistent with the visual hull could be significantly larger, as in Figure 2 (b), where the number of provably occupied



(a) (b) (c) (d)

Fig. 2. (left images) Example Visual hulls with (a) two provably occupied polygons (circled) and one ambiguous polygon and (b) no provably occupied polygons. (right images) A planar graph before (c) and after (d) the reduction of LowerBound (Theorem 3.1). The cones in this case have a very small angle, making them essentially lines. Note that the polygons occur at the intersections.

polygons is 0, but the number of objects is at least 2. Though weak, this lower bound is *informative*, in that all objects contributing to the bound have a known location in the visual hull.

A simple upper bound can also be derived by assuming that all objects are of size at least MINSIZE:

$$UB(P) = \sum_{p \in P} \left\lfloor \frac{\text{area}(p)}{\text{MINSIZE}} \right\rfloor$$

Additionally, if all objects must be larger than MINSIZE, then polygons smaller than this size can be discarded from the visual hull. Thus every polygon in the visual hull contributes at least one to the upper bound. This is the same upper bound used by Yang et al. [10]. This bound is weak in the sense that it assumes objects can fill the polygons completely, which could lead to over-estimating the true number of objects inside a single polygon. It may be possible to tighten this bound by making additional assumptions about the geometry of the objects (e.g., circles of at least some radius).

3.1 Hardness Result for Lower Bound

The *optimal lower bound* is a true count of the smallest number of objects that could produce a given visual hull. Based on the definition of the visual hull, one could equivalently define this number as the size of the smallest set of polygons such that each cone contains at least one polygon in the set. This formulation leads to the following decision problem and hardness result.

Definition 3.4. Given a Visual Hull V, represented using rational numbers, and integer k, *LowerBound* decides whether it is possible to produce V with k or fewer objects.

Theorem 3.1. *LowerBound is NP-complete.*

Proof. The reduction follows from Planar Vertex Cover. Given a planar graph consisting of only straight edges[1] and the vertices in general position, fill in the empty

[1] Chrobak and Payne [1] give a linear-time algorithm for producing a drawing of a planar graph consisting of only straight lines.

regions of the graph with walls. Place a single sensor for each edge in this manner: For the edge (u, v), select either u or v - we will use u for the purposes of this proof. Position a sensor at the chosen vertex looking down the edge towards v. These sensors should be thought of as having a very small field of view. From each sensor, place a cone down the edge, terminating at the wall beyond v. With proper placement of the cones, the only created polygons will be located at the vertices of the graph. See Figure 2 (c, d).

The original graph has a size k vertex cover if and only if this visual hull could have been created by k objects. Since edges in the graph became cones in the visual hull, placing objects in polygons is the same as placing vertices in the cover. Thus, LowerBound can solve Planar Vertex Cover and LowerBound is NP-hard. Note that, given a set of polygons, it is easy to verify that all the cones contain at least one; thus, LowerBound is also trivially in NP, and thus is NP-complete. □

This reduction creates a visual hull with many long, narrow passages and intersections at the vertices. We expect that the proof can be generalized to regions without interior walls by adding a polynomial number of additional sensors (without detections) aimed at the additional intersections which occur when the walls are removed. The details of this construction are somewhat messy because it requires ensuring that the additional sensors have coordinates which can be expressed compactly.

4 Aim Planning

The general *aim planning* problem involves aiming auxiliary sensors to query the status of portions of the visual hull, reducing the gap between the upper and lower bounds. Since it may not be possible to cover the entire visual hull at once, multiple phases of sensor aiming could be required before all possible information has been extracted from a scene, where a *phase* specifies a single aim for each overhead sensor. The goal of this section is to give an algorithm for planning the aims for the overhead sensors and bound the number of phases required to reduce the gap in bounds, UB - LB, to zero (or the smallest number possible).

Our analysis of the multi-phase aim planning problem is divided into parts. First, we analyze a single sensor aim and the possible suboptimality resulting from a simple aiming strategy. We then consider the subproblem of choosing a set of sensor aims to maximize the number of potential objects viewed, and the suboptimality resulting from a greedy strategy. Finally, we combine these results to address the full, multi-phase aiming problem.

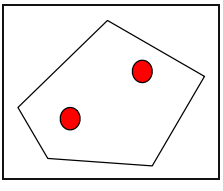
4.1 Overhead Sensor Model

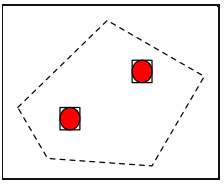
The overhead sensors are aimed by directing the sensor towards a particular area in the plane. To abstract away from the specifics of the hardware, we describe sensor aims by the corresponding area in the plane which is sensed, rather the specific motions or joint angles required to position the sensor.

Definition 4.1. An *aim* is the area in the plane that is within the field of view of an overhead sensor for some possible configuration of the sensor.

We generally assume that the area covered by an aim corresponds to a ball in some metric space, e.g. the L_∞ ball corresponding to the coverage area of a solid state image sensor that is high overhead.

The overhead sensors behave in a manner similar to the horizontal sensors; they detect occupancy in a conical region extending from the sensor and into the scene. What makes the overhead sensors special is their cones are orthogonal to the plane. Moreover, with the mild assumption that objects are not stacked on top of each other, the overhead sensors are immune to ambiguities from occlusions. Each detection by an overhead sensor introduces a new polygon in the plane corresponding to the intersection of the sensor's detection cone with the plane. Figure 3 (left and center) give an example of this sensor model, both before viewing and after.





```
function polyselect(S) ; S = list of sensors
  if S is empty, stop
  for i:1..size(S)
    mx[i] = maxaim(S[i])
  sstar = argmax(mx[i])
  swap(S[0], S[sstar])
  mark the view sstar as viewed
  polyselect(S[1..size(S)])
```

Fig. 3. (left and center) Result of overhead aims. On the left, assume that 5 horizontal cones have produced a polygon which is large enough to contain several objects, in this case 2. The center shows the result of an overhead aim that contains the entire original polygon. Two objects have been detected and polygons corresponding to the intersection of the detection cones with the plane are added, while the rest of the original polygon is removed. The new, square polygons would arise from the detection cones of square image sensor pixels. (right) The Polyselect Algorithm.

4.2 Bound Tightening

This section proves that no reasonable algorithm can do too poorly at tightening the gap between the bounds. We will use the informative lower bound (LB) and the simple upper bound (UB) defined in Section 3. Before stating the major result of this section, we define a useful property of an aim:

Definition 4.2. Given an aim v which covers unviewed polygons $p(v)$ in the visual hull, $C(v)$ is the number of *potential objects* seen by v:

$$C(v) = \sum_{p \in p(v)} \left\lfloor \frac{\text{area}(p \cap v)}{\text{MINSIZE}} \right\rfloor$$

$C(v)$ also provides a lower bound on the greatest change in bounds caused by a single aim.

Lemma 4.1. *Assuming all objects are the same size, choosing aim v will reduce the gap in bounds by* at least $C(v)$*, regardless of the number of objects detected in the aim.*

Proof. Suppose the overhead sensor detects a total of k objects. Viewing k objects creates k new polygons, each of size roughly equal to the size of the objects, as in Figure 3. All other area inside the aim will be removed from the visual hull. Since all objects are the same size, the UB decreases by $C(v) - k$ (the area removed from the visual hull) and the LB increases by k, giving a net change of $C(v)$. □

This does not, however, provide an upper bound on the maximum change in the gap between bounds. Consider viewing an empty polygon; after viewing, this polygon will be removed from the visual hull, changing the *cone upper bounds* (see Section 3) for all the cones that it occupied. It is possible that the cone upper bound is reduced to 1 for each of these cones, creating several new provably occupied polygons. We refer to this deduction as *inference*. For example, in Figure 2 (b), viewing p_1 warrants the inference that both the polygons p_2 and p_3 are provably occupied.

To quantify the change in bounds as a result of inference, let $c_{\max}$ be the maximum number of cones per polygon; $c_{\max}$ is at most the number of sensors in the horizontal network, since each polygon can be composed of at most one cone per sensor. Since the motivation for using overhead sensors will be that the horizontal sensors are sparse enough to create ambiguities, it is reasonable to assume that $c_{\max}$ will not be large in practical applications. This definition, along with Lemma 4.1, leads to an approximation ratio for a general class of algorithms.

Theorem 4.1. *Let A and B be two algorithms for aiming the overhead sensors that choose v_A and v_B (respectively), with $C(v_A) = C(v_B)$. Let A be an optimal algorithm with respect to the bounds gap, whereas B is any algorithm yielding $C(v_A) = C(v_B)$. B is a $c_{\max} + 1$ approximation to A.*

Proof. Consider the change in bounds for algorithm B. In the worst case, the bounds will change by $C(v_B)$, as demonstrated by Lemma 4.1; this corresponds with the case where all the polygons are fully occupied. A can, however, potentially change the bounds by as much as $C(v_A) + |p(v_A)| \cdot (c_{\max})$ by seeing only empty polygons and, for each one, inferring that up to $c_{\max}$ other polygons are occupied. This maximum change in bounds for A can be at most $C(v_A) \cdot (c_{\max} + 1)$, since each polygon contributes at least one to $C(v_A)$. Thus, B is a $c_{\max} + 1$ approximation. □

Since this theorem makes no assumptions about B, *any* algorithm for choosing an aim with $C(v_A)$ potential objects would be a $c_{\max} + 1$ approximation algorithm. Of course, the number of potential objects viewed by an optimal algorithm is not known *a priori*. If B maximizes the number of potential objects viewed, however, then A cannot view more, and B must be a $c_{\max} + 1$ approximation.

4.3 Maximizing the Number of Potential Objects Viewed by Multiple Sensors

This section considers the problem of choosing a set of aims to maximize the number of viewed potential objects. The main result of this section is that a simple, greedy approach yields a constant factor approximation for the largest number of

potential objects the overhead network can see. If the overhead sensors have distinct sets of possible aims, then the greedy algorithm is a 2-approximation. If the overhead sensors are *interchangeable* in the sense that all aims are possible for all sensors, then the greedy algorithm is an $\frac{e}{e-1}$ approximation.

Figure 3 (right) presents the pseudocode for a greedy aiming algorithm called *Polyselect*. Polyselect assumes the existence of a function called *maxaim* that exhaustively considers all possible aims for a sensor and returns the maximum number of new potential objects viewable given the set of aims possible for the sensor. Clearly, there are many opportunities for caching and incremental computation in the implementation of maxaim. Among all sensors for which an aim is not already assigned, Polyselect chooses the sensor and aim that maximizes the number of previously unviewed potential objects. The area chosen by this aim is marked so that subsequent aims do not consider the overlap and the procedure continues until aims are determined for all sensors.

4.3.1 Non-interchangeable Sensors

Theorem 4.2. *Polyselect is a 2-approximation of the optimal aim selection procedure.*

Proof. Polyselect is a 2-approximation because if it chooses a suboptimal aim, then the potential objects contributing to this suboptimality were previously viewed by Polyselect.

More formally, let $G_1, G_2, \ldots, G_m$ be the total number of previously unviewed potential objects seen by the aims chosen by Polyselect, given in descending order, i.e., the ordering chosen by Polyselect. Now consider the output of an optimal algorithm, $O_1, O_2, \ldots, O_m$, where O_j is the optimal aim for sensor j in the Polyselect ordering. Both quantities are only the *new* potential objects seen by each sensor, meaning that O_k does not count any potential objects counted by $O_{1\ldots k-1}$.

Define the *loss* to be the difference between the number of potential objects viewed by the greedy algorithm and the number viewed by an optimal algorithm. Trivially:

$$\text{loss} = \sum_i (O_i - G_i) \leq \sum_i \max\{0, O_i - G_i\}$$

Now consider some $O_j > G_j$, i.e. one of the sensors that contributes to the final summation. For this sensor j, there is an aim viewing a larger number of potential objects than what Polyselect chose, and there are at least $O_j - G_j$ more potential objects at this aim. Since the greedy algorithm chose the aim giving G_j (instead of O_j), however, these additional potential objects must have been covered by sensors Polyselect fixed earlier, and are accounted for in $G_1, G_2, \ldots, G_{j-1}$. Thus, the suboptimality must be bounded by the total number of potential objects seen by Polyselect. More formally,

$$\text{loss} \leq \sum_i \max\{0, O_i - G_i\} \leq \sum_i G_i$$

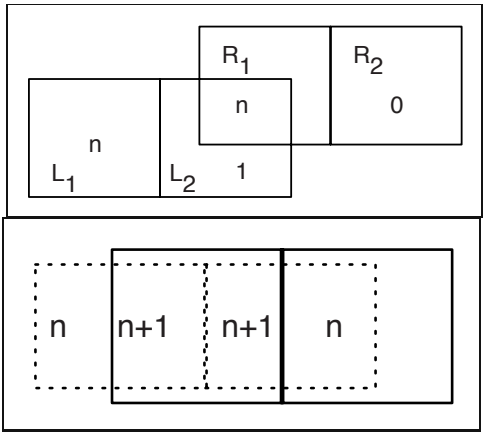


Fig. 4. (top) An example demonstrating the tightness of the approximation ratio in theorem 4.2. There are two aims available to each sensor; for L, the aims available cover n or $n+1$ potential objects. For R, the aims available cover n or 0 potential objects. The potential objects seen by R_1 are a subset of those seen by L_2. The optimal aims are clearly L_1 and R_1, but Polyselect picks L_2 and R_2. (bottom) An example where Polyselect will give a 4/3-approximation with interchangeable sensors. A sensor can choose to cover any two adjacent sets of potential objects. The optimal aims are the two dashed rectangles, while Polyselect chooses the solid rectangles. (The dashed rectangles are smaller for expository purposes only.)

Substituting into the original expression for the loss:

$$\sum_i O_i - \sum_i G_i \leq \sum_i G_i \Rightarrow \sum_i G_i \geq \frac{1}{2}\sum_i O_i$$

yielding a 2-approximation for the optimal set of aims. □

This approximation ratio is also tight. Consider the scenario in Figure 4 (top). Polyselect will choose to aim the sensors at L_2 and R_2, yielding a total of $n+1$ potential objects. An optimal algorithm, however, will aim the sensors at L_1 and R_1, with a total of $2n$ potential objects.

4.3.2 Interchangeable Sensors

If the sensors are *interchangeable*, meaning that all aims are possible for all sensors, the greedy algorithm achieves a better approximation ratio. This result draws upon earlier work on maximizing submodular functions. Nemhauser et al. [8] established several equivalent criteria for a set function z, defined over the subsets of the set A, to be a *submodular non-decreasing function*. We use the following criterion:

$$z(S \cup \{i\}) - z(S) \geq z(T \cup \{i\}) - z(T) \geq 0\,, \forall S \subset T \subset A, \forall i \in A$$

Lemma 4.2. *Let A be the set of available aims and $z_C : 2^A \rightarrow \mathbf{N}$ be the number of potential objects viewed by a subset of these aims. z_C is a non-decreasing, submodular function.*

Proof. Let $S \subset T$ be subsets of A. Consider adding an additional aim i to both sets. Since z_C counts the number of *distinct* potential objects viewed by a subset of the aims, the additional aim i cannot contribute fewer new potential objects to S than it would to T. z_C is also non-decreasing because adding an aim cannot reduce the number of potential objects. □

Note that interchangeability is necessary for submodularity. Without interchangeability, z_C is not a set function, as there are some aims which are not available to all the sensors.

Theorem 4.3. *If the overhead sensors are interchangeable, Polyselect is an $e/(e-1)$-approximation of the optimal aim selection procedure.*

Proof. Nemhauser et al. describe a greedy, $e/(e-1)$ approximation algorithm that starts with an empty set and iteratively builds a solution by adding the item i which maximizes $z(S \cup \{i\}) - z(S)$. Polyselect follows the same procedure and is therefore an instance of this algorithm with the objective function z_C. Since z_C is submodular, the $e/(e-1)$ approximation follows as an immediate consequence of the Nemhauser et al. results. □

Figure 4 (top) shows a case where Polyselect with interchangeable sensors yields a 4/3-approximation to the optimal solution. This is the worst case we have devised, suggesting that the bound in Theorem 4.3 may not be tight.

4.3.3 Hardness of Maximizing Number of Viewed Potential Objects

Could a polynomial time algorithm choose a maximizing set of aims? This section shows that, in general, some form of approximation will be necessary because the basic problem is intractable.

Definition 4.3. Given a collection S of overhead sensors (with $|S| = c$) and a set P of polygons represented with rational coordinates, MaxObject decides whether there exists a set of aims which allow the sensors in the network to see at least k potential objects.

Theorem 4.4. *MaxObject is NP-hard.*

Proof. This problem is NP-hard so long as the number of overhead sensors is considered part of the input. The reduction follows from the *c-center* problem: *Given a set of points P (with $|P| = n$) in $\Re^d$, does there exist a set of c "balls" of radius r which can completely cover all the points in P?*[2]

[2] This problem is generally known as the *p-center* problem.

The *c-center* problem is NP-hard even for $d = 2$ so long as c is part of the input, even when the metric is L_∞ [2]. Note that "balls" of radius r in L_∞ are axis parallel squares of size $2r$. An instance of the c-center problem can be converted to an instance of MaxObject by creating very small (MINSIZE) polygons for each point, and then creating c sensors which can each view a square of size $2r$. Clearly, an algorithm which can solve this instance of MaxObject can also be used to solve the original instance of c-center. MaxObject is thus NP-hard. □

This theorem demonstrates that finding the aims maximizing the number of viewed potential objects is intractable if the number of overhead sensors is part of the problem input. If the number of sensors is a constant c and there are n discrete aims per sensor, then the maximizing set of aims can be found in polynomial time via exhaustive search since there are $O(n^c)$ possible choices. If the set of aims is continuous then finding the maximizing set of aims will require techniques from computational geometry. In either case, the runtime of these procedures can be quite high, even for moderate values of c, making approximation algorithms more practical.

4.4 Multi-phase Bound Resolution

This section considers how Polyselect performs when applied over multiple phases of sensor aims. A *phase* assigns an aim to each sensor and processes the results of the aims, updating the visual hull. In each phase, the network gathers more information about the count in the region. It is assumed that the objects do not move between phases, a reasonable assumption if the objects are either stationary or moving slowly relative to the speed of the sensor movements – a reasonable assumption if the overhead sensors are pan-tilt cameras which can pan or tilt in a second or less.

The goal of this section is to determine how many greedy phases are required to minimize UB - LB, relative to an optimal algorithm. This problem is particularly interesting because the optimal strategy could be conditional: The selection of a certain aim could depend upon the outcome of earlier aims. This section will use the word *resolve* to mean determining the status of a potential object, either through inference or viewing.

The analysis in this section proceeds in two steps. The first step uses the results from Sections 4.2 and 4.3 to bound the performance of the greedy algorithm over the course of one *round*, where a round is the number of phases an optimal algorithm takes to resolve all the potential objects. This bound leads to a simple recurrence which can then be solved to give an upper bound on the total number of greedy rounds required to minimize the gap between the bounds. This section considers both interchangeable and general sensors.

Lemma 4.3. *If an optimal algorithm requires k phases (one round) to resolve n potential objects, then Polyselect will view at least* $n/(2(c_{\max}+1))$ *potential objects in one round with non-interchangeable sensors, and at least* $(n(e-1))/(e(c_{\max}+1))$ *with interchangeable sensors.*

Proof. By Theorem 4.1, an algorithm that exploits inference can resolve at most a factor of $c_{\max}+1$ more potential objects than an algorithm that doesn't plan to exploit inference. To resolve n potential objects, the optimal algorithm must view at least $n/(c_{\max}+1)$ potential objects. If it is possible to view $n/(c_{\max}+1)$ potential objects, then by Theorem 4.2, Polyselect will view at least $n/2(c_{\max}+1)$ when the sensors are not interchangeable and at least $(n(e-1))/(e(c_{\max}+1))$ when they are interchangeable. □

Theorem 4.5. *Using a greedy d-approximation to plan the sensor aims in each phase requires no more than* $d(c_{\max}+1)\log_2 n$ *times as many rounds as an optimal algorithm that plans to exploit inference.*

Proof. Suppose that after some round i of the greedy algorithm, n_{left} potential objects remain. The same set of aims used by the optimal algorithm will suffice to resolve these n_{left} potential objects. Therefore, by Lemma 4.3, the greedy algorithm will be able to view at least $n_{left}/d(c_{\max}+1)$ in the next round. Each round, in the worst case, Polyselect cuts the number of remaining potential objects by a constant factor. Letting $a = d(c_{\max}+1)$ be this constant fraction, this reasoning leads to a simple recurrence:

$$T(n) = T\left(\left(1-\frac{1}{a}\right)n\right)+1$$

Solving the recurrence yields:

$$T(n) = \log_{\frac{a}{a-1}} n = \frac{\log_2 n}{\log_2 a - \log_2 (a-1)}$$

The denominator, $\log_2 a - \log_2 (a-1)$, is a finite difference approximation of the derivative of $\log_2$ at a. Since log is concave, this must be *larger* than the true derivative of $\log_2$ at a, $1/a$, implying:

$$T(n) = \frac{\log_2 n}{\log_2 a - \log_2 (a-1)} \leq \frac{\log_2 n}{\frac{1}{a}} = a\log_2 n$$

For $a = d(c_{\max}+1)$, $T(n) \leq d(c_{\max}+1)\log_2 n$. □

Corollary 4.1. *Polyselect requires at most* $2(c_{\max}+1)\log n$ *more rounds than an optimal algorithm when using general sensors.*

Corollary 4.2. *Polyselect requires at most* $\frac{e}{e-1}(c_{\max}+1)\log n$ *more rounds than an optimal algorithm when using interchangeable sensors.*

4.4.1 Hardness of Multi-phase Planning

In the previous sections, we demonstrated that applying an approximation algorithm to aim a set of sensors at each phase has bounded suboptimality relative to an optimal planning algorithm. One question remains, however: Could a polynomial time algorithm compute this optimal plan? This section shows that computing such an optimal plan is intractable, even when the number of sensors is fixed.

Definition 4.4. Given a collection S of c overhead sensors and a set P of polygons, NumPhases decides whether it is possible to view P with m phases.

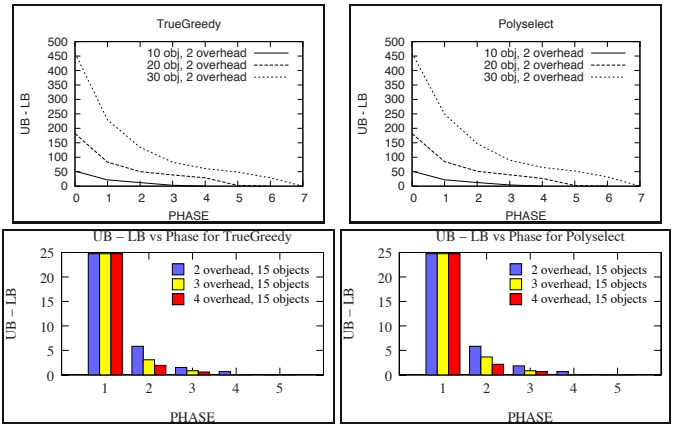


Fig. 5. (top) Plot of the gap in bounds for TrueGreedy (left) and Polyselect (right) vs Phase for two sensors and several different numbers of objects. (bottom) A plot of the gap in bounds for TrueGreedy (let) and Polyselect (right) vs. Phase for 15 objects and various numbers of overhead sensors. Data were averaged over 15 experiments for the top plots and over 12 for the bottom plots. Note that in both cases, the plots are essentially the same.

Theorem 4.6. *NumPhases is NP-hard, even when the number of sensors is fixed* a priori.

Proof. The reduction is from the rectilinear c-center problem, and follows a similar line of reasoning as used in Theorem 4.4. Given a set of points P, create a very small polygon for each point; these polygons should be small enough that none overlap. Next, create a single overhead sensor with a square field of view of radius r and position it such that it can aim at any location within the region of interest.

An algorithm to decide this instance of NumPhases will also decide the original instance of rectilinear c-center. Consider the set of aims chosen by the algorithm deciding NumPhases. These k aims would correspond with k squares (of size $2r$) covering all the points in P, thus also deciding the original decision problem. Therefore, NumPhases is NP-hard. □

This result is much stronger than the result proved in Section 4.3.3 as the problem remains NP-hard even when the number of sensors a constant.

5 Empirical Results

We evaluated our greedy approach using a simulated version of our counting problem with nine horizontal sensors by running Polyselect to completion and measuring the change in bounds over time. To implement `maxaim`, we developed a sweepline approach which finds local maxima in the number of viewed potential objects as the overhead sensor's aim is swept in the y-direction. This sweepline algorithm was then run for a discrete set of x positions (each separated by a constant amount),

generating a set of local optima. This set of detected local maxima is then used as a basis for choosing the aims for Polyselect. We compared the performance of Polyselect to a procedure that truly maximizes the area of viewed polygons (*True-Greedy*) using a brute-force search over all combinations of aims. Both algorithms chose from the same set of aims. The optimal, non-myopic strategy is too expensive to compute because the non-myopic strategy is conditional and could require computing the change in bounds for all possible sequences of aims, as opposed to all possible sequences of just the local maxima. All of the tested configurations had interchangeable sensors.

With two overhead sensors TrueGreedy runs up to 40× slower than PolySelect, and TrueGreedy can be hundreds of times slower with three or more sensors. Figure 5 (top) shows two plots of the bound gap (UB - LB), for TrueGreedy and Polyselect, with various numbers of objects. Figure 5 (bottom) shows the gap in bounds for TrueGreedy and Polyselect for various numbers of overhead sensors. Note that no more than ten phases were required for any of the experiments. We have noticed empirically that Polyselect is often an excellent approximation algorithm, in many cases choosing equivalent aims to TrueGreedy. Consequently, the suboptimality for both sets of plots in Figure 5 is less than a fraction of an object, even for many sensors. As the graphs demonstrate, the suboptimality of using Polyselect is reasonable.

6 Conclusion

We described a simple, greedy method for planning the aims of a set of overhead sensors to resolve an ambiguous count of the number of objects seen by a network of horizontal sensors. We proved that the suboptimality of this approach is both bounded and reasonable, and works well in practice. We also demonstrated that solving the sensor aiming problem optimally is intractable.

Acknowledgment

This work was partially supported by the Sloan Foundation, and by NSF IIS award 0209088, NSF CAREER award 0546709, and by the DARPA CSSG program. Any opinions, findings, conclusions or recommendations are those of the authors only. The authors also wish to thank Pankaj Agarwal for many useful suggestions.

References

1. Chrobak, M., Payne, T.: A linear-time algorithm for drawing a planar graph on a grid. Information Processing Letters 54(4) (1989)
2. Fowler, R.J., Paterson, M.S., Tanimoto, S.L.: Optimal packing and covering in the plane are NP-Complete. Information Processing Letters 12, 133–137 (1981)
3. Guestrin, C., Krause, A., Singh, A.: Near-optimal sensor placements in gaussian processes. In: ICML (2005)

4. He, Y., Chong, E.: Sensor scheduling for target tracking: A Monte Carlo sampling approach. Digital Signal Processing 16, 533–545 (2006)
5. Laurentini, A.: The visual hull concept for silhouette-based image understanding. IEEE PAMI 16, 150–162 (1994)
6. Liu, X., Tu, P., Rittscher, J., Perera, A., Krahnstoever, N.: Detecting and counting people in surveillance applications. In: Advanced Video and Signal Based Surveillance (2005)
7. Masoud, O., Papanikolopoulos, N.: A novel method for tracking and counting pedestrians in real-time using a single camera. IEEE Transactions on Vehicular Technology 50(5), 1267–1278 (2001)
8. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions. Mathematical Programming 14, 265–294 (1978)
9. Särkkä, S., Vehtari, A., Lampinen, J.: Rao-blackwellized particle filter for multiple target tracking. Information Fusion Journal 8, 2–15 (2007)
10. Yang, D., Gonzalez-Banos, H., Guibas, L.: Counting people in crowds with a real-time network of simple image sensors. In: IEEE ICCV (2003)
11. Zhao, T., Nevatia, R.: Tracking multiple humans in crowded environment. In: IEEE CVPR, vol. 2, pp. 406–413 (2004)

Mobile Wireless Sensor Network Connectivity Repair with K-Redundancy

Nuzhet Atay and Burchan Bayazit

Abstract. Connectivity is an important requirement for wireless sensor networks especially in real-time monitoring and data transfer applications. However, node movements and failures change the topology of the initial deployed network, which can result in partitioning of the communication graph. In this paper, we present a method for maintaining and repairing the communication network of a dynamic mobile wireless sensor network. We assume that we cannot control the motion of wireless sensor nodes, but there are robots whose motion can be controlled by the wireless sensor nodes to maintain and repair the connectivity of the network. At the heart of our method lies a novel graph property, *k*-redundancy, which is a measure of the importance of a node to the connectivity of a network. We first show that this property can be used to estimate repair time of a dynamic network. Then, we present a dynamic repair algorithm that minimizes expected repair time. Finally, we show the effectiveness of our method with extensive simulations and its feasibility with experiments on real robots and motes.

1 Introduction

Communication connectivity is a fundamental requirement for wireless sensor networks for the effective use of such systems. It is also observed by several researchers such as [13] that local connectivity improved the system performance in multi-robot applications. In this paper, we are addressing this problem and propose a new method to provide connectivity in a wireless sensor network. In our approach, we classify the nodes as uncontrolled and controlled nodes. The uncontrolled nodes could be mobile or static and we can control the motion of the later class. In the

Nuzhet Atay
Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130, USA
e-mail: atay@cse.wustl.edu

Burchan Bayazit
Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130, USA
e-mail: bayazit@cse.wustl.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 35–49.
springerlink.com

rest of the paper, we will call controllable nodes as robots and uncontrollable nodes as mobile nodes. Our goal is to improve connectivity of the system with the help of mobile robots. Our approach is based on in-network computing, where robots do not know the intentions of mobile nodes, but mobile nodes plan and guide the movements of robots to provide better connectivity.

In order to provide better connectivity, we first introduce a new graph property, *k*-redundancy, to determine the communication characteristics of a dynamic wireless sensor network. This property provides a tool to identify low-connected parts of a communication graph and means to reinforce the network structure before disconnection happens. Briefly, we define *k*-redundancy of a node as the minimum number of node removals required to disconnect any two neighbors of that node. This provides a measure to represent the importance of a node in connecting its neighbors. *k*-redundancy is also important for the robustness of the network because as the redundancy of nodes increase, the routes between neighboring nodes increases.

As we will see in Section 4, *k*-redundancy can be utilized to estimate the repair time in a network. One approach to provide better connectivity is to assign some robots to provide communication bridges if the network is disconnected (reactive repair). An alternative approach is to place robots before the disconnection so that the repair time would be minimum if disconnection happens (proactive repair). In this paper we compare both approaches and show how *k*-redundancy information can be used to improve proactive repair performance. For this purpose, we introduce several proactive repair strategies and compare their performances using simulations with a realistic network simulator (NS-2 [1]). Our results show that by using *k*-redundancy, we can reduce the disconnections in a dynamic mobile network. We also provide real hardware experiments with several mobile robots and motes to show the applicability of our algorithm to real systems.

The rest of the paper is organized as follows. The next section gives a brief summary of the related research and brief comparison to our approach when it is applicable. We introduce problem definition in section 3. Section 4 introduces the concept of *k*-redundancy and Section 5 describes our solution. In section 6, we present our simulation results. Section 7 shows the implementation of our method on real hardware and Section 8 concludes the paper.

2 Related Work

Using mobility to maintain connectivity has attracted many researchers. The general approach has been using mobility for carrying data between disconnected components of the network [26], and using mobile vehicles to improve data collection by actively using vehicles as data carriers [21]. One other approach is storing data when connectivity is disrupted, and sending it when connectivity repairs [25, 19]. The problem with these approaches is the latency in data transfer for time critical applications. The main advantage of our approach is that we are using mobile nodes for forming a connected network where data transfer is never interrupted. There are also approaches to maintain uninterrupted connectivity with dynamic networks. For

example, the decentralized planner of [5] can be extended to retain a communication network. A dynamic radiomap is used in [14] to navigate in good communication areas. In [22], the authors propose a technique for providing radio connectivity while moving a group of robots from one configuration to another. Another approach [12] aims to provide radio connectivity and line of sight while moving a swarm from one configuration to another. Generally, these approaches have explicit assumptions on the communication properties which can be violated in practice. The advantages of our technique are *1:* We assume there can be obstacles in the environment, *2:* We allow links to be canceled and reformed, *3:* We do not have any assumptions on the communication model, i.e. there is no assumption on the communication range or the properties of links.

3 Problem Definition

We have a network of mobile nodes whose motion we cannot control. Our purpose is to maintain and repair connectivity of this network using a group of mobile robots which are capable of moving to appropriate regions and build communication bridges. We assume robots are controlled by nodes. The network is monitored by nodes and nodes determine where robots should be located to improve connectivity. Robots and nodes do not have location information, neither for themselves nor for other members of the network. However, each of them is capable of measuring distance and determining the direction to a neighbor if that neighbor is in its line of sight. Each member of the network is equipped with a low-power radio for wireless communication with limited range. Robots and nodes are holonomic with limited speed. There are obstacles in the environment which can obstruct line of sight and interrupt communication. Nodes and robots can fail anytime. We do not assume any communication model, environment map, or motion prediction of nodes when deciding robot locations. Our solution is distributed and we do not use any global information.

4 K-Redundancy and Expected Repair Time

4.1 K-Redundancy

K-connectivity is a property which is used to define the minimum number of nodes that need to be removed in order to partition a graph. If a graph is K-connected, the graph remains connected if any $K-1$ nodes are removed. One problem with this property is that it gives information about the whole graph, not about individual nodes or parts of the graph. So even if most of the graph is fully connected, a small low-connected region determines the connectivity of the whole graph. This fact considerably limits the information we can obtain about graphs.

Although K-connectivity is defined for the graph as a whole and a global property, we can modify this concept to define the connectivity property of individual nodes. For this purpose, we define a new graph property (i,j,k)-redundancy for

each node. We are using this property to represent the goodness of the connectivity among the neighbors of each node. It should be noted that a node could create a communication bridge between any pair of its neighbors, but if there are alternative routes between the neighbors, the importance of that node on connectivity reduces.

Definition 4.1 (i-neighborhood). *Let v be a node of graph G with vertex set $V(G)$ and edge set $E(G)$. We denote i-neighborhood of node v as $N_i(v)$, which is the set of nodes whose distance to the node v is at most i. The subgraph induced by $N_i(v)$ is denoted by $G[N_i(v)]$, which is the set of vertices $N_i(v)$ with edges whose both endpoints are in $N_i(v)$.*

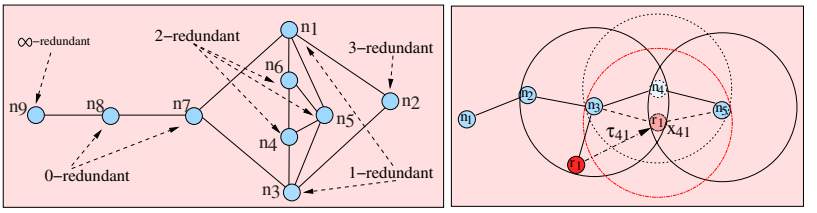
Definition 4.2 ((i,j,k)-redundancy). *A node n is defined to be (i,j,k)-redun-dant if $N_i(n)-S$ is contained in one connected component of $G[N_j(n)-S]$ for all sets S with $k-1$ vertices of $N_j(n)$ and $j \geq i$.*

In other words, a node n is (i,j,k)-redundant if k is the minimal number of nodes in the j-neighborhood of n to separate any two nodes in the i-neighborhood n. We can also say that a node n is (∞,∞,k)-redundant if the graph G is k-connected, and $G[V(G)-\{n\}]$ is $(k-1)$-connected. Various definitions of local k-connectivity can be derived by adjusting i and j. We denote by k-redundancy the special case $(1,\infty,k)$-redundancy, which is measuring the connectivity of 1-neighbors of n over the whole graph. k-redundancy for nodes with only one neighbor is undefined following this definition, but we define their connectivity as ∞ for practical purposes. It should be noted that a 0-redundant node is an articulation point (cut vertex) of the graph. Our definition is a generalization of this property, which enables us to obtain more information about the connectivity of the graph. To the best of our knowledge, there is no graph property similar to (i,j,k)-redundancy. In our implementation, we find (i,j,k)-redundancy by evaluating the accessibility of each node in i-neighborhood of n. The details of our implementation can be found at [3]. We see an efficient algorithm for finding (i,j,k)-redundancy as an interesting open problem.

Fig. 1(a) shows an example graph where the vertices represent the nodes and edges represent the connections between the nodes. Nodes have different redundancy values which is a representation of their role in the connectivity. For example, n_8 is 0-redundant because n_9 and n_7 can communicate only with the help of n_8, so removing it from the graph results in disconnection. n_3 is 1-redundant because in case it fails, all of its neighbors can communicate with the help of n_1, removing also n_1 partitions the graph. n_6 is 2-redundant because after removing it, at least two more nodes need to be removed to partition its neighbors (n_3 and n_5 can be removed to isolate n_4).

4.2 Expected Repair Time

Now consider a scenario where a wireless sensor network is disconnected and we have some robots that can move in and build a communication bridge that would



(a) (b)

Fig. 1. (a) Nodes have various k-redundancy values according to their role in the connectivity of the graph. (b) Node n_4 fails and r_1 repairs the network. x_{41} is the optimal location for robot r_1 to repair the network in case n_4 fails, and τ_{41} is the time to reach there.

reconnect the disconnected parts. In this section, we will investigate the expected repair time if there is only one disconnection.

We first define *network repair time* (τ_{repair}) as the time from the moment of disconnection to reach a network topology where all the nodes are connected again. Reconnection may occur because of the dynamic changes in the network, such as the random movements of nodes, without utilizing any robot for repair, but in this part, we only consider the time for using a robot to repair a disconnected network.

Utilizing a robot enables us to formalize the expected repair time in a wireless sensor network with several nodes of various k-redundancy levels. Remember that in order to make two neighbors of a k-redundant node unreachable to each other, it requires k nodes to be disconnected from the network in addition to that node. Assuming that the network topology is stable from the time disconnection occurs to the time a robot repairs it, we can repair the network by sending a robot to the location of the node that caused disconnection. However, in some cases, a robot can repair the connection even before reaching the location of the failed node. We call x_{ij} the optimal position that a robot j needs to move to repair the network in case node i fails and network gets disconnected. We define τ_{ij} as the time of robot j to reach x_{ij}. Fig. 1(b) illustrates the optimal location to repair the network and the time for the closest robot to reach there. In this figure, r_1 repairs the network in case node n_4 fails. If we assume that a node has probability p to disconnect, then the probability of the network disconnecting at a k-redundant node is p^{k+1}, because k additional nodes need to fail in addition to that node so as to obtain a disconnected network. Then, we can write expected repair time as a function of the probability that the network disconnects at a given node and the time for a robot to reach to a position to repair the network.

$$E(\tau) = \sum_{i=1}^{n} min_j(\tau_{ij}) * p^{k_i+1}, \quad 1 \leq j \leq m \tag{1}$$

where n is the number of nodes, m is the number of robots, and k_i is k-redundancy for the node i.

In Equation 1, we assume that all nodes have the same probability of getting disconnected. The equation could easily be extended to include different probabilities for each node (perhaps based on the signal strength, direction of nodes or distance between nodes). Please also note that this equation presents a theoretical basis for our algorithm, and we do not need to know the exact values of p and τ_{ij} for analysis purposes. In later sections, we will discuss different approaches to find robot placements.

5 Network Repair

Our goal is to provide at least the minimum k-redundancy for all mobile nodes in the wireless sensor network. We achieve this by continuously checking k-redundancy for each node and request assistance from a mobile robot if k-redundancy becomes less than the minimum redundancy. Please note that, if the minimum redundancy is selected to be 0, reduction in the redundancy means that the network is disconnected. Alternatively, we can enforce a high redundancy value to (a) provide more robust network, (b) increase the throughput between mobile nodes. In this section, we will discuss how we can find k-redundancy for each node. We will also present two methods for repairing a disconnection. In the first method, the reactive algorithm, the network directs robots when the k-redundancy of a node becomes less than minimum value. In the second method, the proactive method, the network place the robots at locations that minimize the repair time in case k-redundancy becomes less than the minimum value.

5.1 Computing K-Redundancy

(i,j,k)-redundancy definition requires finding all alternative paths between the i-hop neighbors of a node, where paths can cover the j-hop neighborhood. Hence, in order for a mobile node to find its redundancy, a communication mechanism is required. In our approach, each node stores j-hop neighborhood information. To determine its role in connectivity, each node enumerates all ways of communication between each pair of i-hop neighbors. This way, each node can determine its role in communication. The pair which has the least number of ways of communication determines that node's (i,j,k)-redundancy. We are only interested in evaluating the importance of a node in connecting its immediate neighbors, so we compute $(1,j,k)$. In practice, j must be small, as a result, computed redundancy value $(1,j,k)$-redundancy can be different from k-redundancy, which is defined as $(1,\infty,k)$-redundancy. However, the local value is a lower limit on the redundancy of nodes, i.e., nodes can have higher redundancies if they are computed globally, but nodes cannot have lower redundancy. So, redundancy values computed using local information are a good indicator of graph connectivity. Our experiments suggest that 2-redundancy is sufficient for practical applications of network repair [3].

5.2 Reactive Repair

The reconnection process starts when a node starts drifting away from one of its immediate neighbors. If losing this neighbor does not reduce k-redundancy of this node to a value less than the minimum redundancy, no action is taken. Otherwise, one of the robots in the network needs to move to that region and form connections with those two nodes. If possible, robots try to form connections to as many nodes as possible, which in turn increases redundancy of the node. After this time, robots assume the responsibility of maintaining connectivity just as a regular node, i.e. they can call other robots to repair connection. This allows the robots to build bridges consisting of several robots between the disconnected parts of the network. To avoid unnecessary deployment of robots, a periodic connectivity detection mechanism works on the nodes who has connection to the robot. If all nodes can communicate between each other without the help of the robot, the robot is unnecessary and leaves that region.

5.3 Proactive Repair

In proactive repair, we place robots in locations that minimize the repair time in case of node failures or disconnections. Once the robots are in these locations, they are utilized around there until a node's redundancy goes below than minimum redundancy, i.e., reactive repair is required. The best location for a robot to minimize the expected repair time is the location that minimizes Equation. 1.

Finding the best locations to minimize expected repair time can be represented as the facility location problem. Facility location problem has been studied extensively because of its practical applications, and it involves challenging combinatorial and geometrical problems. In general, the facility location problem is finding the locations of a set of facilities F to serve a set of demand locations D with minimum cost, where cost function c_{ij} is the weighted distance function for $j \in D$ and $i \in F$ that is nearest to j. In our original problem, facilities are the robots, demand locations are the locations that the robots can move to repair the network, and cost function is the weighted distance where weight is the probability of a node failing and causing disconnectivity. As facility location problems are NP-Hard, solution to Equation. 1 is also NP-Hard. We propose three policies for solving this problem: (i) robots can be placed only around nodes, and if a node fails, a robot can repair the network by taking its place (P1), (ii) a robot can repair the network by taking the failed node's place as before, but robots can be placed anywhere (P2), (iii) robots can be placed anywhere, but instead of moving all the way to take the place of the failed node, it moves to the point which is closest to the robot and enough to repair the network (P3).

In the following, we will discuss how our policies can be implemented by using several variations of the facility location problem.

5.3.1 Discrete Demand and Facility Sets for Policy I

Facility location problem is called k-median when both demand and facility sets are discrete, and at most k facilities can be used. Our first policy is k-median, as we assume that robots can be located only around nodes, and they can move to the previous location of the failed node to repair the network. So, both feasible facility locations and demand locations are node locations, and k is the number of idle robots.

K-median problem is NP-Hard [11], so exact solutions are not feasible. However, there are several heuristics that work well in practice. We discuss two different algorithms for the solution of this problem:

Greedy Approach: In the first algorithm, we locate robots around low redundant nodes. This approach comes from the fact that a robot near a lower redundancy node would have faster repair time than a robot near a higher redundancy node. Please see [3] for the proof demonstrating this on certain graphs. We use point-greedy algorithm [11] for solving this problem. In this algorithm, one facility is added at a time that minimizes cost. More formally, start with solution set $S = \emptyset$, and update $S = S \cup f_k$ where $f_k \in F - S$ such that $cost(S)$ is minimized. This algorithm is very fast but it has an approximation ratio $O(n)$, where n is the number of demand locations.

Local Search: In the second algorithm, instead of a step-by-step allocation, we check different combinations and minimize the cost over all combinations. This is the exact solution of k-median and is NP-Hard. Best known polynomial time solution to this problem is local search. In this approach, one random feasible solution is determined, and at each step, at most p facilities are swapped until no more improvement can be obtained, i.e. some facilities are removed and some are utilized, keeping at most k facilities utilized at any time. This approach has approximation ratio $3 + 2/p$ and the running time is $O(n^p)$ [2], where n is the number of facilities.

This approximation ratio is for metric k-median problem, i.e. cost function needs to be symmetric and satisfy triangle inequality. Although the cost function in our problem definition is symmetric, there are cases where it violates triangle inequality. In order to overcome this problem, we define the distance between a node and a robot as the length of the shortest path on the graph.

Local search method is a centralized method, but it is suitable for computing in a distributed fashion. This approach starts with a greedy solution where robots are located near low redundant nodes. Then, each node assigns itself to the closest robot, creating a partitioning of the network. After this step, the algorithm starts running asynchronously. We assume each robot is controlled by the node that is closest to it. Each node, at a random time, computes the cost of the solution if it moves the robot to one of its neighbors. If this solution gives a lower cost, then it actually sends the robot. Otherwise, after waiting for a random amount of time, it checks another node, until all neighbors are tried and no more improvement can be obtained. In a network of m robots and n nodes, assuming random sampling prevents more than one node to swap robots, i.e., $p = 1$, our algorithm finishes in $O(n)$ time, and the

approximation ratio is 5 [2]. The details of the algorithm, as well as its pseudocode can be found at [3].

5.3.2 Discrete Demand and Continuous Facility Sets for Policies II and III

Facility location problem is called Fermat-Weber [23] problem when facilities can be located anywhere in the plane and only 1 facility can be utilized. More formally, let $D = \{a_1, a_2, \ldots, a_m\}$ be the set of m points in $\Re^n$, Fermat-Weber problem is to find a point q which minimizes the sum of the weighted Euclidean distances to the points in D:

$$\min_q d(q) = \sum_{i=1}^{m} w_i \|q - a_i\|_n \tag{2}$$

where $\|.\|_n$ denotes Euclidean distance in $\Re^n$ and w_i is the weight for point a_i.

Proactive repair approach can be transformed into this problem when there is only one robot, by using k-redundancy to define the weights in the formula. In this case, let $D = \{n_1, n_2, \ldots, n_m\}$ be the set of nodes in $\Re^2$, our problem is to locate the robot r to minimize expected repair time:

$$\min_r E(r) = \sum_{i=1}^{m} p_i^{k_i+1} \|r - n_i\|_2 \tag{3}$$

This problem does not have an exact analytical solution even when $m = 5$ [4]. However, if the points in the set D are not collinear, then this function is positive and strictly convex, hence it has a unique minimum. In the case where the points are collinear, at least one of the points in D is the minimum, and it can be found in linear time. For the noncollinear case, one of the most popular algorithms is Weiszfeld's algorithm [24] which is an iterative method. The iteration function is given by:

$$T(r) = \begin{cases} \dfrac{\sum_{i=1}^{m} \dfrac{p_i^{k_i+1} n_i}{\|r - n_i\|}}{\sum_{i=1}^{m} \dfrac{p_i^{k_i+1}}{\|r - n_i\|}} & \text{if } r \neq n_1, \ldots, n_m \\ n_i & \text{if } r = n_i \end{cases} \tag{4}$$

Weiszfeld's algorithms is defined using this function as an iterative scheme:

$$r_{i+1} = T(r_i)\ i = 0, 1, 2 \ldots \tag{5}$$

This function is continuous and differentiable everywhere except the points in D. If no r_i results in a point in D, these points converge to the global optimal solution r_{opt} in a finite number of iterations [16]. However, when any r_i is in D, then the algorithms terminates at r_i without convergence. The set of initial points r_0 that causes any $r_i \in D$ where $i > 0$ is denumerable (countable) [9, 7, 8], if the convex hull of the points in D has full dimension. In our problem, we are working on $\Re^2$, hence the convex hull of the points in D has 2-dimensional convex hull as long as the

points are noncollinear. As a result, the set of starting points is always denumerable when the points in D are noncollinear. We only use this algorithm when the points are noncollinear, so we can claim that the set of starting points that will cause early termination without convergence is denumerable. Thus, by selecting another starting point whenever the algorithm terminates early, r_{opt} can be found in finite number of steps. However, in practice, it is very unlikely that any r_i will exactly land on a point in D with the exact numerical precision.

The negative gradient of the function $E(r)$ is defined as:

$$R(r) = \sum_{i=1}^{m} \frac{p_i^{k_i+1}}{\|r - n_i\|}(n_i - r) \tag{6}$$

Convexity of $E(r)$ implies that the necessary and sufficient condition for optimality is $R(r_{opt}) = 0$. We use this result to generate the stopping criterion. Algorithm stops at iteration i when $\|R(r_i)\|_2 \leq \varepsilon$, where ε is small and positive.

The rate of convergence depends on whether or not r_{opt} is in D. If $r_{opt} \notin D$, then the rate of convergence is linear. On the other hand, if $r_{opt} \in D$, the convergence rate can be linear, superlinear or sublinear [15].

Multi-Facility: When multiple facilities are used, facility location problem with continuous facility set turns into a very hard optimization problem, and it has been shown to be NP-Hard [17]. Although properties of the optimal solution is known [20], there is no known heuristic with performance bound. We have chosen one of best heuristics, sequential location and allocation (SLA) [10, 6]. This method starts with an arbitrary solution, and assigns each demand location to the closest facility which results in a clustering of demand locations. Then, single facility location problem (1-Weber) is solved for each cluster, and new facility locations are found. This process continues until no more improvement can be obtained, and converges to a local optimal solution.

In order to apply this solution to our problem, we assign continuous facility set as the feasible robot locations in the plane, so that $F \subseteq \Re^2$. Discrete demand set are the locations for the robots to move and repair the network in case a node fails and repair is required. The details of the algorithm, as well as its pseudocode can be found at [3]. For solving the 1-Weber problem, we are using the Weiszfeld's algorithm. In this part, we also explore two different policies:

- **Constant Demand Set:** In our second policy (P2), we set demand set as the node locations and apply SLA algorithm. Initially, we set robot locations as their initial locations and find an assignment for nodes, which results in a clustering of nodes. We then solve Weber problem inside each cluster, and continue until robot locations cannot be changed to reduce expected repair time.
- **Updated Demand Set:** In the last policy (P3), we first form the regions that a robot can move and repair connectivity (repair regions whose construction explained in [3]), and pick the closest points inside these regions as the demand locations. The solution of the problem using this policy gives the solution of Equation. 1. We again use SLA algorithm to solve this problem. In this

algorithm, each time robot locations are selected and single facility location is solved, demand set is updated according to the new robot locations.

6 Simulations

In our simulations, we want to determine the characteristics of connectivity in mobile networks and the effect of using robots to reinforce network and repair connectivity. We are interested in observing the effects of increasing k-redundancy, success of the proposed methods in maintaining connectivity, effects of obstacles and node failures. Because of space limitations, we only present some of the results. Our full results can be found at [3]. We implemented and tested our algorithm on the network simulator NS-2 [1]. NS-2 is one of the most commonly used network simulators and it can simulate realistic network conditions including message transfer, network congestion, delay etc. In our experiments, we have selected $k = 2$ (see [3] for evaluation of different k values). We setup communication range to be 45 meters which is around the range of low-power radios. Before presenting simulation results, we discuss properties for measuring connectivity.

6.1 Connectivity Measure

There are two metrics for measuring connectivity in our experiments. The first one is the classical measure which is 1 when network is connected, and 0 if the network is partitioned. The second metric which is called reachability [18] is more useful to measure connectivity on a continuous scale. This metric is defined as the ratio of the total number of node pairs that can communicate among each other to the 2-combination of all nodes. This number reaches 1 when all nodes are connected, and 0 when there is no connection between any nodes.

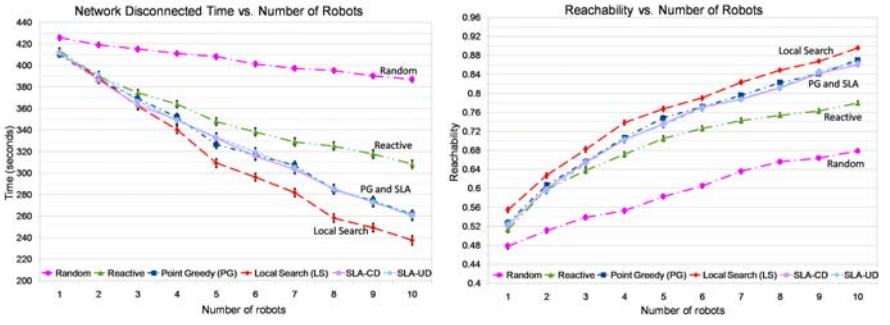
6.2 Simulation Results

Success Rate: These simulations show the success of the proposed methods in maintaining network connectivity when there are different number of robots in the system. For this purpose, we present average total disconnection time during the simulation period and average reachability. We measure these in two different setups. We first let robots move randomly just like nodes, and in the second one, we have the robots controlled by nodes using reactive and proactive methods. We allow robots to move randomly and compare to our technique, instead of simply adding robots to the network because increasing the number of mobile entities alone can help connectivity. As a result, the number of mobile units is same in comparison, only the behavior of robots change. We started with a network of 10 mobile nodes, and added one robot at each experiment. We used 10 different initial graphs for any given number of robots, and the experiment has been repeated by starting with each graph 100 times. Each simulation time is 500 seconds.

Fig. 2(a) shows the time the graph remains disconnected. When there is only one robot in the network, the difference in the times between random and controlled robot motion is about 15 seconds. However, as the number of robots increase, the time difference starts increasing. The difference in disconnected time between random motion and reactive approach reaches 80 seconds when there are 10 robots for repair. The results show that proactive approach performs better than reactive approach. In the proactive approach, we have tested point greedy (PG), local search (LS), sequential location and allocation with constant demand set (SLA-CD), and sequential location and allocation with updated demand set (SLA-UD) methods. As it can be seen in Fig. 2(a), PG, SLA-CD and SLA-UD performs similar, whereas LS outperforms all these three methods. When there are 10 robots, the time difference between LS and PG, SLA-CD and SLA-UD reaches 25 seconds. The time we obtain using LS is 150 seconds better than random motion, and 70 seconds better than reactive approach. We should note that the algorithm used to send robots to repair the network in case a disconnection is detected is the same with reactive and proactive approach. This shows that placing robots using k-redundancy can provide improvement about 70 seconds over 500 seconds.

One interesting observation is that both SLA-CD and SLA-UD perform very similar to PG. Although SLA can place robots anywhere in the 2-D Euclidean space, it can perform much worse than optimal solution on certain graphs. When the initial distribution of the robots is uneven among the different parts of the graph, one part of the graph can have very little expected repair time, whereas the other part has very high because of the uneven distribution of robots. This solution is possible to obtain using SLA because there is no force that can push some of the robots from the dense part of the graph to the sparse part of the graph. PG can also suffer from this problem because of the greedy approach. The greedy approach minimizes cost at each step. So if there are two dense parts of the graph connected by a single bridge and there are two robots, the first robot goes along that bridge, and the second robot goes to one of the dense parts. In an optimal solution though, each robot is placed in one dense part. However, LS can avoid these cases by moving the robots to other parts of the network as long as improvement can be obtained. Finally, we have also seen that there is not statistically significant difference between SLA-CD and SLA-UD. With SLA-CD, all nodes send their own locations to be used in the computation of Weber point, whereas with SLA-UD, nodes send the closest point required to repair network. However, the optimal position of the robot is always among the nodes, so most of the time, the change in the distance computations of the nodes opposite to each other with respect to the robot cancel each other, and the location of the Weber point changes very little. Although the expected repair time with SLA-UD turns out to be smaller than SLA-CD, the optimal robot position is very similar.

Reachability measurements show similar results(Fig. 2(b)). When there is only one robot, the difference in reachability between random motion and reactive approach is 0.03, and the difference between random motion and LS is 0.07. Reachability increases with all methods as the number of robots increase. The reachability of LS reaches up to 0.9 when there are 10 robots, which means that on average, 90% of the nodes in the network are connected to each other over 500 seconds. With 10



(a) (b)

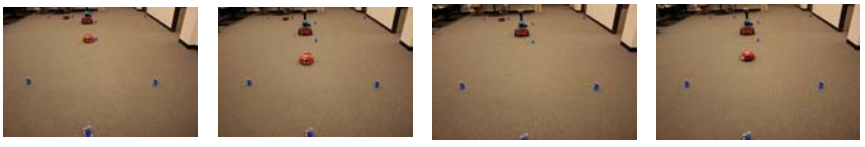
Fig. 2. (a) Average disconnected time as the number of robots increase. (b) Reachability as the number of robots increase.

robots, the difference between random motion and reactive approach reaches 0.1, and the difference between random motion and LS reaches 0.22. This means using LS, 22% more nodes can communicate among each other compared to random motion.

7 Experiments

We show the feasibility of our approach with experiments on real hardware. For this purpose, we experimented on a network formed of 3 robots (2 AmigoBots, 1 Pioneer 3-DX) and 10 Tmote sky motes. Each robot is equipped with a mote, and the other 7 motes are used as static nodes. 1 AmigoBot is used as a mobile node, and the other robots are used as connectivity repair robots. We present the working of the system in an environment of size 8mx8m. In our experiments, we set a communication range of 2.5 meters to imitate radio communication range, so although motes hear all other motes in the environment, they filter out messages from motes who are further away than 2.5m.

Initial experiment setup is shown in Fig. 3(a). The mobile node represented with an AmigoBot is working as a bridge between the upper and lower parts of the network. Two repair robots are located at the two minimum redundant nodes in the upper part. When AmigoBot moves closer to the camera, this causes a disconnectivity



(a) (b) (c) (d)

Fig. 3. Real experiments in a scenario that represents bridge forming and node failure handling

in the network, so the closest idle robot (Pioneer) moves towards that region to repair connectivity (Fig. 3(b)). Then, AmigoBot fails (Fig. 3(c)) so another disconnection occurs in the network. This time, the robot who is supposed to provide connectivity (Pioneer) acts as a mobile node and calls for another robot, and the second AmigoBot reaches the region and maintains connectivity with the neighbors of the failed node (Fig. 3(d)).

8 Conclusion

In this paper, we have presented a new graph property, *k*-redundancy, to define the communication characteristics of a dynamic wireless sensor network. This property provides a way to represent the effects of removing a node from the network on the connectivity. We show that this property can be used to estimate repair time to reconnect a network. We have presented an in-network algorithm that is based on *k*-redundancy to improve the network's connectivity where mobile nodes request mobile robots to repair low connected areas. Finally, we have showed the performance of our algorithm in simulations and real hardware.

Acknowledgements. We would like to thank Steve LaValle, Robert Ghrist, Douglas West, Joseph Mitchell, Robert Pless, Jianer Chen and Jennifer Welch for useful discussions about the novelty of *k*-redundancy who guided us in the development of this graph property. We also would like to thank Parasol Lab and Nancy Amato for providing AmigoBots we used in the experiments.

References

1. NS-2 Network Simulator, http://www.isi.edu/nsnam/ns/
2. Arya, V., Garg, N., Khandekar, R., Munagala, K., Pandit, V.: Local search heuristic for k-median and facility location problems. In: STOC 2001: Proceedings of the thirty-third annual ACM symposium on Theory of computing, pp. 21–29. ACM, New York (2001)
3. Atay, H.N.: Connectivity Maintanence and Task Allocation For Mobile Wireless Sensor Networs. Ph.D. dissertation, Washington University in St. Louis (August 2008)
4. Bajaj, C.: The algebraic degree of geometric optimization problems. Discrete and Computational Geometry 3, 177–191 (1988)
5. Bekris, K.E., Tsianos, K.I., Kavraki, L.E.: A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS), San Diego, CA, pp. 3784–3790 (2007)
6. Brandeau, M., Chiu, S.: Sequential location and allocation: Worst case performance and statistical estimation. Location Science 1, 289–298 (1993)
7. Brimberg, J.: The fermat-weber location problem revisited. Math. Program. 71(1), 71–76 (1995)
8. Brimberg, J.: Further notes on convergence of the weiszfeld algorithm. Yugoslav Journal of Operations Research 13(2), 199–206 (2003)
9. Chandrasekaran, R., Tamir, A.: Open questions concerning weiszfeld's algorithm for the fermat-weber location problem. Mathematical Programming 44(1), 293–295 (1989)

10. Cooper, L.: Heuristic methods for location-allocation problems. SIAM Review 6(1), 37–53 (1964)
11. Cornuejols, G., Fisher, M.L., Nemhauser, G.L.: Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. Management Science 23(8), 789–810 (1977)
12. Esposito, J., Dunbar, T.: Maintaining wireless connectivity constraints for swarms in the presence of obstacles. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, May 15-19, pp. 946–951 (2006)
13. Gerkey, B.P., Matarić, M.J.: Principled communication for dynamic multi-robot task allocation. In: Rus, D., Singh, S. (eds.) Experimental Robotics VII. LNCIS, vol. 271, pp. 353–362. Springer, Berlin (2001)
14. Hsieh, M.A., Cowley, A., Keller, J.F., Chaimowicz, L., Grocholsky, B., Kumar, V., Taylor, C.J., Endo, Y., Arkin, R.C., Jung, B., Wolf, D.F., Sukhatme, G.S., MacKenzie, D.C.: Adaptive teams of autonomous aerial and ground robots for situational awareness: Field reports. J. Field Robot. 24(11-12), 991–1014 (2007)
15. Katz, I.N.: Local convergence in fermat's problem. Mathematical Programming 6(1), 89–104 (1974)
16. Kuhn, H.W.: A note on fermat's problem. Mathematical Programming 4(1), 98–107 (1973)
17. Megiddo, N., Supowit, K.J.: On the complexity of some common geometric location problems. SIAM Journal on Computing 13(1), 182–196 (1984)
18. Perur, S., Iyer, S.: Characterization of a connectivity measure for sparse wireless multihop networks. In: 26th IEEE International Conference on Distributed Computing Systems Workshops, ICDCS Workshops 2006, July 4-7, pp. 80–85 (2006)
19. Rao, N., Qishi, W., Iyengar, S., Manickam, A.: Connectivity-through-time protocols for dynamic wireless networks to support mobile robot teams. In: IEEE International Conference on Robotics and Automation (ICRA), 2003, September 14-19, vol. 2, pp. 1653–1658 (2003)
20. Rosing, K.E., Harris, B.: Algorithmic and technical improvements: Optimal solutions to the (generalized) multi-weber problem. Papers in Regional Science 71(3), 331–352 (1992)
21. Somasundara, A.A., Kansal, A., Jea, D.D., Estrin, D., Srivastava, M.B.: Controllably mobile infrastructure for low energy embedded networks. IEEE Transactions on Mobile Computing 5(8), 958–973 (2006)
22. Spanos, D., Murray, R.: Motion planning with wireless network constraints. In: Proceedings of the 2005 American Control Conference, pp. 87–92 (2005)
23. Weber, A.: Theory of the Location of Industries, translated by Carl J. Friedrich. University Of Chicago Press, Chicago (1965)
24. Weiszfeld, E.: Sur le point pour lequal la somme des distances de n points donnés est minimum. Tohoku Mathematics Journal 43, 355–386 (1937)
25. Yang, G., Chen, L.-J., Sun, T., Zhou, B., Gerla, M.: Ad-hoc storage overlay system (asos): A delay-tolerant approach in manets. In: Proceeding of the IEEE MASS, pp. 296–305 (2006)
26. Zhao, W., Ammar, M., Zegura, E.: A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: MobiHoc 2004: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing, pp. 187–198 (2004)

Part II
Distributed Systems

On Endogenous Reconfiguration in Mobile Robotic Networks

Ketan Savla and Emilio Frazzoli

Abstract. In this paper, our focus is on certain applications for mobile robotic networks, where reconfiguration is driven by factors intrinsic to the network rather than changes in the external environment. In particular, we study a version of the coverage problem useful for surveillance applications, where the objective is to position the robots in order to minimize the average distance from a random point in a given environment to the closest robot. This problem has been well-studied for omni-directional robots and it is shown that optimal configuration for the network is a centroidal Voronoi configuration and that the coverage cost belongs to $\Theta(m^{-1/2})$, where m is the number of robots in the network. In this paper, we study this problem for more realistic models of robots, namely the double integrator (DI) model and the differential drive (DD) model. We observe that the introduction of these motion constraints in the algorithm design problem gives rise to an interesting behavior. For a *sparser* network, the optimal algorithm for these models of robots mimics that for omni-directional robots. We propose novel algorithms whose performances are within a constant factor of the optimal asymptotically (i.e., as $m \to +\infty$). In particular, we prove that the coverage cost for the DI and DD models of robots is of order $m^{-1/3}$. Additionally, we show that, as the network grows, these novel algorithms outperform the conventional algorithm; hence necessitating a reconfiguration in the network in order to maintain optimal quality of service.

1 Introduction

The advent of large scale sensor and robotic networks has led to a surge of interest in reconfigurable networks. These systems are usually designed to reconfigure in a

Ketan Savla
Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
e-mail: ksavla@mit.edu

Emilio Frazzoli
Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
e-mail: frazzoli@mit.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 53–67.
springerlink.com

reactive way, i.e., as a response to changes in external conditions. Due to their importance in sensor network applications, reconfiguration algorithms have attracted a lot of attention, e.g., see [8]. However, there are very few instances in engineering systems, if any, that demonstrate an internal reconfiguration in order to maintain a certain level of performance when certain *intrinsic* properties of the system are changed. However, examples of endogenous reconfiguration or phase transitions are abound in nature, e.g., desert locusts [4] who switch between gregarious and social behavior abruptly, etc. An understanding of the phase transitions can not only provide insight into the reasons for transitions in naturally occurring systems but also identify some design principles involving phase transition to maintain efficiency in engineered systems.

In this paper, we observe such a phenomenon under a well-studied setting that is relevant for various surveillance applications. We consider a version of the so-called Dynamic Traveling Repairperson Problem, first proposed by [11] and later developed in [2]. In this problem, service requests are generated dynamically. In order to fulfill a request, one of the vehicles needs to travel to its location. The objective is to design strategies for task assignment and motion planning of the robots that minimizes the average waiting time of a service request. In this paper, we consider a special case of this problem when service requests are generated sparingly. This problem, also known as coverage problem, has been well-studied in the robotics and operations research community. However, we consider the problem in the context of realistic models of robots: double integrator models and differential drive robots. Some preliminary work on coverage for curvature-constrained vehicles was reported in our earlier work [7]. In this paper, we observe that when one takes into consideration the motion constraints of the robots, the optimal solution exhibits a phase transition that depends on the size of the network.

The contributions of this paper are threefold. First, we identify an interesting characteristic of the solution to the coverage problem for double integrator and differential drive robots, where reconfiguration is necessitated intrinsically by the growth of the network, in order to maintain optimality. Second, we propose novel approximation algorithms for double integrator robots as well as differential drive robots and prove that they are within a constant factor of the optimal in the asymptotic ($m \to +\infty$) case. Moreover, we prove that, asymptotically, these novel algorithms will outperform the conventional algorithms for omni-directional robots. Lastly, we show that the coverage for both, double integrator as well as differential drive robots scales as $1/m^{1/3}$ asymptotically.

Due to space limitations, we do not present all the proofs. Missing proofs and details are presented in the extended version of this article [12].

2 Problem Formulation and Preliminary Concepts

In this section, we formulate the problem and present some preliminary concepts.

Problem Formulation

The problem that we consider falls into the general category of the so called *Dynamic Traveling Repairperson Problem*, originally proposed in [11]. Let $Q \subset \mathbb{R}^2$ be a convex, compact domain on the plane, with non-empty interior; we will refer to Q as the *environment*. For simplicity in presentation, we assume that Q is a square, although all the analysis presented in this paper carries through for any convex and compact Q with non-empty interior in $\mathbb{R}^2$. Let A be the area of Q. A spatio-temporal Poisson process generates *service requests* with finite time intensity $\lambda > 0$ and uniform spatial density inside the environment. In this paper, we focus our attention on the case when $\lambda \to 0^+$, i.e., when the service requests are generated very rarely.

These service requests are identical and are to be fulfilled by a team of m robots. A service request is fulfilled when one of m robots moves to the target point associated with it.

We will consider two robot models: the double integrator model and the differential drive model. The double integrator (DI) model describes the dynamics of a robot with significant inertia. The configuration of the robot is $g = (x, y, v_x, v_y) \subset \mathbb{R}^4$ where (x, y) is the position of the robot in Cartesian coordinates, and (v_x, v_y) is its velocity. The dynamics of the DI robot are given by

$$\begin{aligned}
\dot{x}(t) &= v_x(t), \\
\dot{y}(t) &= v_y(t), \quad v_x(t)^2 + v_y(t)^2 \leq v_{\max}^2 \quad \forall t \\
\dot{v}_x(t) &= u_x(t), \\
\dot{v}_y(t) &= u_y(t), \quad u_x(t)^2 + u_y(t)^2 \leq u_{\max}^2 \quad \forall t,
\end{aligned}$$

where $v_{\max}$ and $u_{\max}$ are the bounds on the speed and the acceleration of the robots.

The differential drive model describes the kinematics of a robot with two independently actuated wheels, each a distance ρ from the center of the robot. The configuration of the robot is a directed point in the plane, $g = (x, y, \theta) \subset \mathrm{SE}(2)$ where (x, y) is the position of the robot in Cartesian coordinates, and θ is the heading angle with respect to the x axis. The dynamics of the DD robot are given by

$$\begin{aligned}
\dot{x}(t) &= \frac{1}{2}(w_l(t) + w_r(t))\cos\theta(t), \\
\dot{y}(t) &= \frac{1}{2}(w_l(t) + w_r(t))\sin\theta(t), \\
\dot{\theta}(t) &= \frac{1}{2\rho}(w_r(t) - w_l(t)), \quad |w_l(t)| \leq w_{\max} \forall t, |w_r(t)| \leq w_{\max} \forall t,
\end{aligned}$$

where the inputs w_l and w_r are the angular velocities of the left and the right wheels, which we assume to be bounded by $w_{\max}$. Here, we have also assumed that the robot wheels have unit radius.

The robots are assumed to be identical. The strategies of the robots in the presence and absence of service requests are governed by their *motion coordination algorithm*. A motion coordination algorithm is a function that determines the

actions of each robot over time. For the time being, we will denote these functions as $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, but do not explicitly state their domain; the output of these functions is a steering command for each vehicle. The objective is the design of motion coordination algorithms that allow the robots to fulfill service requests efficiently. To formalize the notion of efficiency, let $T(j)$ be the time elapsed between the generation of the j-th service request, and the time it is fulfilled and let $T_\pi := \lim_{j\to+\infty} \lim_{\lambda\to 0^+} \mathrm{E}[T(j)]$ be defined as the system time under policy π, i.e., the expected time a service request must wait before being fulfilled, given that the robots follow the algorithm defined by π. We shall also refer to the average system time as the *coverage cost*. Note that the system time T_π can be thought of as a measure of the quality of service collectively provided by the robots.

In this paper, we wish to devise motion coordination algorithms that yield a quality of service (i.e., system time) achieving, or approximating, the theoretical optimal performance given by $T_{\mathrm{opt}} = \inf_\pi T_\pi$. Since finding the optimal algorithm maybe computationally intractable, we are also interested in designing computationally efficient algorithms that are within a constant factor of the optimal, i.e., policies π such that $T_\pi \leq \kappa T_{\mathrm{opt}}$ for some constant κ. Moreover, we are interested in studying the scaling of the performance of the algorithms with m, i.e., size of the network, other parameters remaining constant. Since, we keep A fixed, this is also equivalent to study the scaling of the performance with respect to the density m/A of the network.

We now describe how a solution to this problem gives rise to an endogenous reconfiguration in the robotic network.

Endogenous Reconfiguration

The focus of this paper is on *endogenous* reconfiguration, that is a reconfiguration necessitated by the growth of the network (as the term 'endogenous' implies in biology), as opposed to any external stimulus. We formally describe its meaning in the context of this paper. In the course of the paper, we shall propose and analyze various algorithms for the coverage problem. In particular, for each model of the robot, we will propose two algorithms, π' and π''. The policy π' closely resembles the omni-directional based policy, whereas the novel π'' policy optimizes the performance when the motion constraints of the robots start playing a significant role. We shall show that there exists a constant $c > 1$ such that

$$\lim_{m/\mathrm{A}\to 0^+} \frac{T_{\pi'}}{T_{\mathrm{opt}}} = 1, \quad \lim_{m\to+\infty} \frac{T_{\pi''}}{T_{\pi'}} = 0, \quad \limsup_{m\to+\infty} \frac{T_{\pi''}}{T_{\mathrm{opt}}} \leq c.$$

This shows that, for sparse networks, the omni-directional model based algorithm π' is indeed a reasonable algorithm. However, as the network size increases, there is a phase transition, during which the motion constraints start becoming important and the π'' algorithm starts outperforming the π' algorithm. Moreover, the π'' algorithm performs with a constant factor of the optimal in the asymptotic ($m \to +\infty$) case. Hence, in order to maintain efficiency, one needs to switch away from the π'

policy as the network grows. It is in this sense that we shall use the term endogenous reconfiguration to denote a switch in the optimal policy with the growth of the network.

3 Lower Bounds

In this section, we derive lower bounds on the coverage cost for the robotic network that are independent of any motion coordination algorithm adopted by the robots. Our first lower bound is obtained by modeling the robots as *equivalent* omni-directional robots. Before stating the lower bounds formally, we need to briefly review a related problem from computational geometry which has direct consequences for the case omni-directional robots.

The Continuous m-median Problem

Given a convex, compact set $\mathrm{Q} \subset \mathbb{R}^2$ and a set of points $p = \{p_1, p_2, \ldots, p_m\} \in \mathrm{Q}^m$, the expected distance between a random point q, sampled from a uniform distribution over Q, and the closest point in p is given by

$$\mathscr{H}_m(p,\mathrm{Q}) := \int_{\mathrm{Q}} \frac{1}{\mathrm{A}} \min_{i\in\{1,\ldots,m\}} \|p_i - q\|\, dq = \sum_{i=1}^{m} \int_{\mathscr{V}_i(p)} \frac{1}{\mathrm{A}} \|p_i - q\|\, dq,$$

where $\mathscr{V}(p) = (\mathscr{V}_1(p), \mathscr{V}_2(p), \ldots, \mathscr{V}_m(p))$ is the *Voronoi (Dirichlet) partition* [9] of Q generated by the points in p, i.e.,

$$\mathscr{V}_i(p) = \{q \in \mathrm{Q} : \|q - p_i\| \leq \|q - p_j\|, \forall j \in \{1,\ldots,m\}\}, \qquad i \in \{1,\ldots,m\}.$$

The problem of choosing p to minimize $\mathscr{H}_m$ is known in geometric optimization [1] and facility location [6] literature as the (continuous) m-median problem. The m-median of the set Q is the global minimizer

$$p_m^*(\mathrm{Q}) = \mathrm{argmin}_{p\in\mathrm{Q}^m}\, \mathscr{H}_m(p,\mathrm{Q}).$$

We let $\mathscr{H}_m^*(\mathrm{Q}) = \mathscr{H}_m(p_m^*(\mathrm{Q}),\mathrm{Q})$ be the global minimum of $\mathscr{H}_m$. The solution of the continuous m-median problem is hard in the general case because the function $p \mapsto \mathscr{H}_m(p,\mathrm{Q})$ is not convex for $m > 1$. However, gradient algorithms for the continuous multi-median problem can be designed [5]. We would not go further into the details of computing these m-median locations and assume that these locations are given or that a computationally efficient algorithm for obtaining them is available.

This particular problem formulation, with demand generated independently and uniformly from a continuous set, is studied thoroughly in [10] for square regions and [13] for more general compact regions. It is shown in [13] that, in the asymptotic ($m \to +\infty$) case, $\mathscr{H}_m^*(\mathrm{Q}) = c_{\mathrm{hex}}\sqrt{\frac{\mathrm{A}}{m}}$ almost surely, where $c_{\mathrm{hex}} \approx 0.377$ is the first moment of a hexagon of unit area about its center. This optimal asymptotic value

is achieved by placing the m points on a regular hexagonal network within Q (the honeycomb heuristic). Working towards the above result, it is also shown in [13] that for any $m \in \natural$:

$$\frac{2}{3}\sqrt{\frac{\mathrm{A}}{\pi m}} \leq \mathscr{H}_m^*(\mathrm{Q}) \leq c(\mathrm{Q})\sqrt{\frac{\mathrm{A}}{m}}, \tag{1}$$

where $c(\mathrm{Q})$ is a constant depending on the shape of Q. In particular, for a square Q, $c(\mathrm{Q}) \approx 0.38$.

We use two different superscripts on T_{opt} for the two models of the robots, i.e., we use $T_{\mathrm{opt}}^{\mathrm{DI}}$ for the DI robot and $T_{\mathrm{opt}}^{\mathrm{DD}}$ for the DD robot. Finally, we state a lower bound on these quantities as follows.

Lemma 3.1. *The coverage cost satisfies the following lower bound.*

$$T_{\mathrm{opt}}^{\mathrm{DI}} \geq \frac{\mathscr{H}_m^*(\mathrm{Q})}{v_{max}}, \quad T_{\mathrm{opt}}^{\mathrm{DD}} \geq \frac{\mathscr{H}_m^*(\mathrm{Q})}{w_{max}}.$$

Remark 3.1. Since from Equation (1), $\mathscr{H}_m^*(\mathrm{Q}) \in \Omega(1/\sqrt{m})$, Lemma 3.1 implies that $T_{\mathrm{opt}}^{\mathrm{DI}}$ and $T_{\mathrm{opt}}^{\mathrm{DD}}$ also belong to $\Omega(1/\sqrt{m})$.

This lower bound will be particularly useful for proving the optimality of algorithms for sparse networks. We now proceed towards deriving a tighter lower bound which will be relevant for dense networks. The reachable sets of the two models of robots will play a crucial role in deriving the new lower bound. We study them next.

Reachable Sets for the Robots

In this subsection, we state important properties of the reachable sets of the double integrator and differential drive robots that are useful in obtaining tighter lower bound.

Let $\tau : G \times \mathbb{R}^2 \to \mathbb{R}^+$ be the minimum time required to steer a robot from initial configuration g in G to a point q in the plane. For the DI robot, $G = \mathbb{R}^4$ and for DD robots $G = \mathrm{SE}(2)$. With a slight abuse of terminology, we define the reachable set of a robot, $\mathscr{R}_t(g)$, to be the set of all points q in Q that are reachable in time $t > 0$ starting at configuration g. Note that, in this definition, we do not put any other constraint (e.g., heading angle, etc.) on the terminal point q. Formally, the reachable set is defined as

$$\mathscr{R}_t(g) = \{q \in \mathbb{R}^2 \mid \tau(g,q) \leq t\}.$$

We now state a series of useful properties of the reachable sets.

Lemma 3.2 (Upper bound on the small-time reachable set area). *The area of the reachable set for a DI robot starting at a configuration $g = (x, y, \dot{x}, \dot{y}) \in \mathbb{R}^4$, with $\dot{x}^2 + \dot{y}^2 = v_0$, satisfies the following upper bound.*

$$\mathrm{Area}(\mathscr{R}_t(g)) \leq \begin{cases} 2v_0 u_{max} t^3 + o(t^3), & \text{as } t \to 0^+ \text{ for } v_0 > 0, \\ u_{max}^2 t^4 & \text{for } v_0 = 0. \end{cases}$$

The area of the reachable set for a DD robot starting at any configuration $g \in \mathrm{SE}(2)$ satisfies the following upper bound.

$$\mathrm{Area}(\mathscr{R}_t(g)) \leq \frac{5}{6\rho} w_{max}^3 t^3 + o(t^3), \quad as\ t \to 0^+.$$

Lemma 3.3 (Lower bound on the reachable set area). *The area of the reachable set of a DI robot starting at a configuration $g = (x, y, \dot{x}, \dot{y}) \in \mathbb{R}^4$, with $\dot{x}^2 + \dot{y}^2 = v_0$, satisfies the following lower bound.*

$$\mathrm{Area}(\mathscr{R}_t(g)) \geq \frac{v_0 u_{max}}{3} t^3 \quad \forall t \leq \frac{\pi}{2} \frac{v_0^2}{u_{max}}.$$

The area of the reachable set of a DD robot starting at any configuration $g \in \mathrm{SE}(2)$ satisfies the following lower bound.

$$\mathrm{Area}(\mathscr{R}_t(g)) \geq \frac{2}{3\rho} w_{max}^3 t^3.$$

Lemma 3.4. *The travel time to a point in the reachable set for a DI robot starting at a configuration $g = (x, y, \dot{x}, \dot{y}) \in \mathbb{R}^4$, with $\dot{x}^2 + \dot{y}^2 = v_0$, satisfies the following property.*

$$\int_{\mathscr{R}_t(g)} \tau(g, q) dq \geq \frac{v_0 u_{max}}{12} t^4 \quad \forall t \leq \frac{\pi}{2} \frac{v_0^2}{u_{max}}.$$

The travel time to a point in the reachable set for a DD robot starting at any configuration $g \in \mathrm{SE}(2)$ satisfies the following property.

$$\int_{\mathscr{R}_t(g)} \tau(g, q) dq \geq \frac{w_{max}^3}{6\rho} t^4.$$

We are now ready to state a new lower bound on the coverage cost.

Theorem 3.1 (Asymptotic lower bound on the coverage cost). *The coverage cost for a network of DI or DD robots satisfies the following asymptotic lower bound.*

$$\liminf_{m \to +\infty} T_{\mathrm{opt}}^{\mathrm{DI}} m^{1/3} \geq \frac{1}{24} \left(\frac{\mathrm{A}}{2 v_{max} u_{max}} \right)^{1/3}, \quad \textit{and}$$
$$\liminf_{m \to +\infty} T_{\mathrm{opt}}^{\mathrm{DD}} m^{1/3} \geq \frac{1}{5 w_{max}} \left(\frac{6\rho \mathrm{A}}{5} \right)^{1/3}.$$

Proof. We state the proof for the double integrator robot. The proof for the differential drive robot follows along similar lines.

In the following, we use the notation $\mathrm{A}_i = \mathrm{Area}(\mathscr{DV}_i(g))$, where $\mathscr{DV}_i(g) := \{q \in \mathrm{Q} \mid \tau(g_i, q) \leq \tau(g_j, q) \quad \forall j \neq i\}$. We begin with

$$T^{\mathrm{DI}}_{\mathrm{opt}} = \inf_{g\in\mathbb{R}^{4m}} \sum_{i}^{m} \int_{\mathscr{V}_i(g)} \frac{1}{\mathrm{A}} \tau(g_i,q)\, dq$$
$$\geq \inf_{g\in\mathbb{R}^{4m}} \int_{\mathrm{Q}} \frac{1}{\mathrm{A}} \min_{i\in\{1,\ldots,m\}} \tau(g_i,q) dq. \tag{2}$$

Let $\bar{R}_{\mathrm{A}_i}(g_i)$ be the reachable set starting at configuration g and whose area is A_i. Using the fact that, given an area A_i, the region with the minimum integral of the travel time to the points in it is the reachable set of area A_i, one can write Equation (2) as

$$T^{\mathrm{DI}}_{\mathrm{opt}} \geq \inf_{g\in\mathbb{R}^{4m}} \sum_{i}^{m} \int_{\bar{R}_{\mathrm{A}_i}(g_i)} \frac{1}{\mathrm{A}} \tau(g_i,q)\, dq. \tag{3}$$

Let t_i be defined such that $\mathrm{Area}(\mathscr{R}_{t_i}(g_i)) = \mathrm{A}_i$. Lets assume that as $m \to +\infty$, $\mathrm{A}_i \to 0^+$ (this point will be justified later on). In that case, we know from Lemma 3.2 that, t_i can be lower bounded as $t_i \geq \left(\frac{\mathrm{A}_i}{2v_i u_{\max}}\right)^{1/3}$, where v_i is the speed associated with the state g_i. Therefore, from Equation (3) and Lemma 3.4, one can write that

$$T^{\mathrm{DI}}_{\mathrm{opt}} \geq \inf_{g\in\mathbb{R}^{4m}} \sum_{i}^{m} \int_{\mathscr{R}_{t_i}(g_i)} \frac{1}{\mathrm{A}} \tau(g_i,q)\, dq$$
$$\geq \inf_{g\in\mathbb{R}^{4m}} \sum_{i}^{m} \frac{1}{\mathrm{A}} \frac{v_i u_{\max}}{12} t_i^4. \tag{4}$$

Using the above mentioned lower bound on t_i, Equation (4) can be written as

$$T^{\mathrm{DI}}_{\mathrm{opt}} \geq \frac{1}{24\sqrt[3]{2}\mathrm{A}} \frac{1}{u_{\max}^{1/3}} \inf_{g\in\mathbb{R}^{4m}} \frac{\mathrm{A}_i^{4/3}}{v_i^{1/3}}$$
$$\geq \frac{1}{24\sqrt[3]{2}\mathrm{A}} \frac{1}{u_{\max}^{1/3} v_{\max}^{1/3}} \min_{\{\mathrm{A}_1,\mathrm{A}_2,\ldots,\mathrm{A}_m\}\in\mathbb{R}^m} \sum_{i}^{m} \mathrm{A}_i^{4/3}$$
$$\text{subject to} \quad \sum_{i}^{m} \mathrm{A}_i = \mathrm{A} \quad \text{and } \mathrm{A}_i \geq 0 \quad \forall i \in \{1,\ldots,m\}.$$

Note that the function $f(x) = x^{4/3}$ is continuous, strictly increasing and convex. Thus by using the Karush-Kuhn-Tucker conditions [3], one can show that the quantity $\sum_i^m \mathrm{A}_i^{4/3}$ is minimized with an equitable partition, i.e., $\mathrm{A}_i = \mathrm{A}/m$, $\forall i$. This also justifies the assumption that for $m \to +\infty$, $\mathrm{A}_i \to 0^+$.

Remark 3.2. Theorem 3.1 shows that both $T^{\mathrm{DI}}_{\mathrm{opt}}$ and $T^{\mathrm{DD}}_{\mathrm{opt}}$ belong to $\Omega(1/m^{1/3})$.

4 Algorithms and Their Analyses

In this section, we propose novel algorithms, analyze their performance and explain how the size of the network plays a role in selecting the right algorithm.

We start with an algorithm which closely resembles the one for omni-directional robots. Before that, we first make a remark relevant for the analysis of all the algorithms to follow.

Remark 4.1. Note that if n_0 is the number of outstanding service requests at initial time, then the time required to service all of them is finite (Q being bounded). During this time period, the probability of appearance of a new service requests appearing is zero (since we are dealing with the case when the rate of generation of targets λ is arbitrarily close to zero). Hence, after an initial transient, with probability one, all the robots will be in their stationary locations at the appearance of the new target. Moreover, the probability of number of outstanding targets being more than one is also zero. Hence, in the analysis of the algorithms, without any loss of generality, we shall implicitly assume that there no outstanding service requests initially.

The Median Stationing (MS) Algorithm

Place the m robots at rest at the m-median locations of Q. In case of the DD robot, its heading is chosen arbitrarily. These m-median locations will be referred to as the *stationary locations* of the robots. When a service request appears, it is assigned to the robot whose stationary location is closest to the location of the service request. In order to travel to the service location, the robots use the fastest path with no terminal constraints at the service location. In absence of outstanding service tasks, the robot returns to its stationary location. The stationary configurations are depicted in Figure 1.

Let $T_{\mathrm{MS}}^{\mathrm{DI}}$ and $T_{\mathrm{MS}}^{\mathrm{DD}}$ be the coverage cost as given by the above policy for the DI and DD robots, respectively.

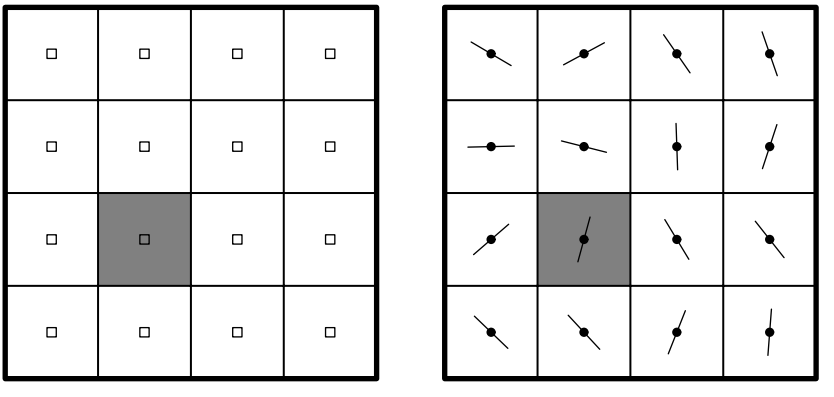


Fig. 1. Depiction of typical stationary configurations for the Median Stationing Algorithm. On the left: DI robots at rest at their stationary locations. On the right: DD robots with arbitrary headings at their stationary locations. In both the figures, the shaded cell represents a typical *region of responsibility* for a robot.

Theorem 4.1 (Analysis of the MS algorithm). *The coverage cost for a network of DI and DD robots with the MS algorithm satisfies the following bounds.*

$$\frac{\mathscr{H}_m^*(\mathrm{Q})}{v_{max}} \leq T_{\mathrm{MS}}^{\mathrm{DI}} \leq \frac{\mathscr{H}_m^*(\mathrm{Q})}{v_{max}} + \frac{v_{max}}{2u_{max}} + \sqrt{\frac{2\sqrt{2\mathrm{A}}}{u_{max}}}.$$

$$\frac{\mathscr{H}_m^*(\mathrm{Q})}{w_{max}} \leq T_{\mathrm{MS}}^{\mathrm{DD}} \leq \frac{\mathscr{H}_m^*(\mathrm{Q})}{w_{max}} + \frac{\rho\pi}{2w_{max}}.$$

Proof. The lower bounds on $T_{\mathrm{MS}}^{\mathrm{DI}}$ and $T_{\mathrm{MS}}^{\mathrm{DD}}$ follow trivially from Lemma 3.1.

For a double integrator robot, the minimum travel time from rest at location p to a point q is given by

$$\tau((p,0),q) \leq \begin{cases} \sqrt{\frac{2\|p-q\|}{u_{\max}}} & \text{for } \|p-q\| \leq \frac{v_{\max}^2}{2u_{\max}}, \\ \frac{\|p-q\|}{v_{\max}} + \frac{v_{\max}}{2u_{\max}} & \text{otherwise.} \end{cases}$$

Therefore, for any q, $\tau((p,0),q)$ can be upper bounded as

$$\begin{aligned} \tau((p,0),q) &\leq \frac{\|p-q\|}{v_{\max}} + \frac{v_{\max}}{2u_{\max}} + \sqrt{\frac{2\|p-q\|}{u_{\max}}} \\ &\leq \frac{\|p-q\|}{v_{\max}} + \frac{v_{\max}}{2u_{\max}} + \sqrt{\frac{2\sqrt{2\mathrm{A}}}{u_{\max}}}. \end{aligned}$$

The upper bound on $T_{\mathrm{MS}}^{\mathrm{DI}}$ is then obtained by taking the expected value over all $q \in \mathrm{Q}$ while taking into consideration the assignment policies for the services to robots. For a differential drive robot, the travel time from any initial configuration (p,θ) to a point q can be upper bounded by $\frac{\|p-q\|}{w_{\max}} + \frac{\pi\rho}{2w_{\max}}$. The result follows by taking expected value of the travel time over all points in Q.

Remark 4.2. Theorem 4.1 and Lemma 3.1 along with Equation (1) imply that,

$$\lim_{m/\mathrm{A}\to 0^+} \frac{T_{\mathrm{MS}}^{\mathrm{DI}}}{T_{\mathrm{opt}}^{\mathrm{DI}}} = 1 \quad \text{and} \quad \lim_{m/\mathrm{A}\to 0^+} \frac{T_{\mathrm{MS}}^{\mathrm{DD}}}{T_{\mathrm{opt}}^{\mathrm{DD}}} = 1.$$

This implies that the MS algorithm is indeed a reasonable algorithm for sparse networks, where the travel time for a robot to reach a service location is almost the same as that for an omni-directional vehicle.

However, as the density of robots increases, the assigned service locations to the robots start getting relatively closer. In that case, the motion constraints start having a significant effect on the travel time of the robots and it is not obvious in that case that the MS algorithm is indeed the best one.

In fact, for dense networks, one can get a tighter lower bound on the performance of the MS algorithm:

Theorem 4.2. *The coverage cost of a dense network of DI and DD robots with the MS algorithm satisfies the following bounds:*

$$\liminf_{m\to\infty} T_{\mathrm{MS}}^{\mathrm{DI}} m^{1/4} \geq 0.321 \left(\frac{\mathscr{A}}{u_{max}^2}\right)^{1/4}.$$

$$\liminf_{m\to\infty} T_{\mathrm{MS}}^{\mathrm{DD}} \geq \frac{\pi\rho}{4w_{max}}.$$

In other words, for DI robots, the MS algorithm requires them to stay stationary in absence of any outstanding service requests. Once a service request is assigned to a robot, the amount of time spend in attaining the maximum speed $v_{\max}$ becomes significant as the location of assigned service requests start getting closer. Similar arguments hold for DD robots.

An alternate approach, as proposed in the next algorithm, is to keep the robots moving rather than waiting in absence of outstanding service requests. The algorithm assigns dynamic regions of responsibility to the robots.

The Strip Loitering (SL) Algorithm

This algorithm is an adaptation of a similar algorithm proposed in [7] for Dubins vehicle, i.e., vehicles constrained to move forward with constant speeds along paths of bounded curvature.

Let the robots move with constant speed $v^* = \min\{v_{\max}, \frac{\sqrt{\sqrt{\mathrm{A}}u_{\max}}}{3.22}\}$ and follow a loitering path which is constructed as follows. Divide Q into strips of width w where $w = \min\left\{\left(\frac{4}{3\sqrt{\rho^*}}\frac{\mathrm{A}+10.38\rho^*\sqrt{\mathrm{A}}}{m}\right)^{2/3}, 2\rho^*\right\}$, where $\rho^* := \frac{v^{*2}}{u_{\max}}$. Orient the strips along a side of Q. Construct a closed path which can be traversed by a double integrator robot while always moving with constant speed v^*. This closed path runs along the longitudinal bisector of each strip, visiting all strips from top-to-bottom, making U-turns between strips at the edges of Q, and finally returning to the initial configuration. The m robots loiter on this path, equally spaced, in terms of path length. A depiction of the Strip Loitering algorithm can be viewed in Figure 2. Moreover, in Figure 3 we define two distances that are important in the analysis of this algorithm. Variable d_2 is the length of the shortest path departing from the loitering path and ending at the target (a circular arc of radius ρ^*). The robot responsible for visiting the target is the one closest in terms of loitering path length (variable d_1) to the point of departure, at the time of target-arrival. Note that there may be robots closer to the target in terms of the actual distance. However, we find that the assignment strategy described above lends itself to tractable analysis.

After a robot has serviced a target, it must return to its place in the loitering path. We now describe a method to accomplish this task through the example shown in Fig. 3. After making a left turn of length d_2 to service the target, the robot makes a right turn of length $2d_2$ followed by another left turn of length d_2, returning it to the loitering path. However, the robot has fallen behind in the loitering pattern. To rectify this, as it nears the end of the current strip, it takes its U-turn early.

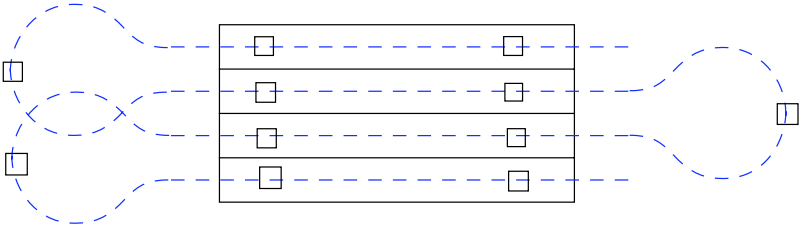


Fig. 2. Depiction of the loitering path for the double integrator robots. The segment providing closure of the loitering path (returning the robots from the end of the last strip to the beginning of the first strip) is not shown here for clarity of the drawing.

Let $T_{\mathrm{SL}}^{\mathrm{DI}}$ be the coverage cost as given by the above algorithm for DI robots. We now state an upper bound on $T_{\mathrm{SL}}^{\mathrm{DI}}$.

Theorem 4.3 (Analysis of the SL algorithm). *The coverage cost for a team of DI robots implementing the SL algorithm satisfies the following asymptotic upper bound.*

$$\limsup_{m\to+\infty} T_{\mathrm{SL}}^{\mathrm{DI}} m^{1/3} \leq \frac{1.238}{v^*}\left(\rho^* \mathrm{A} + 10.38\rho^{*2}\sqrt{\mathrm{A}}\right)^{1/3}.$$

Remark 4.3. Theorem 4.3 and Theorem 3.1 imply that $T_{\mathrm{opt}}^{\mathrm{DI}}$ belongs to $\Theta(1/m^{1/3})$. Moreover, Theorem 4.3 and Theorem 4.2 together with Equation 1 imply that $T_{\mathrm{SL}}^{\mathrm{DI}}/T_{\mathrm{MS}}^{\mathrm{DI}} \to 0^+$ as $m \to +\infty$. Hence, asymptotically, the SL algorithm outperforms the MS algorithm and is within a constant factor of the optimal.

We now present the second algorithm for DD robots.

The Median Clustering (MC) Algorithm

Form as many teams of robots with $k := \left\lceil 4.09\left(\frac{\rho}{\sqrt{A}}\right)^{2/3} m^{1/3} \right\rceil$ robots in each team. If there are additional robots, group them into one of these teams. Let $n := \left\lfloor \frac{m}{k} \right\rfloor$ denote the total number of teams formed. Position these n teams at the n-median locations of Q, i.e., all the robots in a team are co-located at the median location of its team. Within each team j, $j \in \{1,\ldots,n\}$, the headings of the robots belonging to that team are selected as follows. Let $\ell_j \geq k$ be the number of robots in team j. Pick a direction randomly. The heading of one robot is aligned with this direction. The heading of the second robot is selected to be along a line making an angle $\frac{\pi}{\ell}$, in the counter-clockwise direction, with the first robot. The headings of the remaining robots are selected similarly along directions making $\frac{\pi}{\ell}$-angle with the previous one (see Figure 4). These headings will be called the *median headings* of the robots. Each robot in a team is assigned a *dominance region* which is the region formed by the intersection of double cone making half angle of $\frac{\pi}{2\ell}$ with its median heading and the Voronoi cell belonging to the team (see Figure 4). When a service request appears, it is assigned to the robot whose dominance region contains its location.

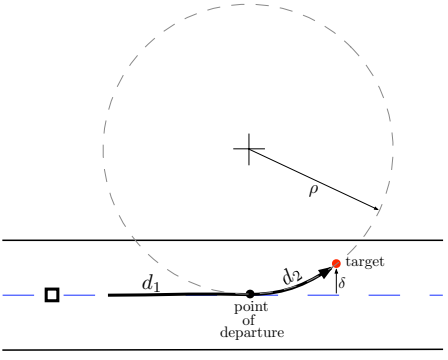


Fig. 3. Close-up of the loitering path with construction of the point of departure and the distances δ, d_1, and d_2 for a given target, at the instant of appearance.

The assigned robot travels to the service location in the fastest possible way and, upon the completion of the service, returns to the median location of its team and aligns itself with its original median heading.

Let $T_{\mathrm{MC}}^{\mathrm{DD}}$ be the coverage cost as given by the above policy. We now state an upper bound on $T_{\mathrm{MC}}^{\mathrm{DD}}$ in the following theorem.

Theorem 4.4 (Analysis of the MC algorithm). *The coverage cost for a team of DD robots while implementing the MC algorithm satisfies the following asymptotic upper bound.*

$$\limsup_{m\to+\infty} T_{\mathrm{MC}}^{\mathrm{DD}} m^{1/3} \leq \frac{1.15}{w_{max}} (\rho \mathrm{A})^{1/3}.$$

Proof. The travel time for any robot from its median location p to the location q of a service request is upper bounded by $\frac{\|p-q\|}{w_{\max}} + \frac{\pi\rho}{2w_{\max}k}$. Taking the expected value of this quantity while taking into consideration the assignment policy of the service requests gives us that

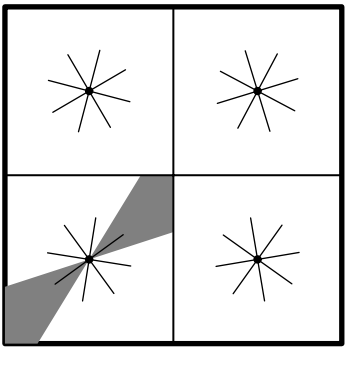


Fig. 4. Depiction of the Median Clustering Algorithm with teams of 4 robots each. The shaded region represents a typical *region of responsibility* for a robot.

$$T^{\mathrm{DD}}_{\mathrm{MC}} \leq \frac{\mathscr{H}^*_n(\mathrm{Q})}{w_{\max}} + \frac{\pi\rho}{2w_{\max}k}. \tag{5}$$

From Equation (1), $\mathscr{H}^*_n(\mathrm{Q}) \leq 0.38\sqrt{\frac{\mathrm{A}}{n}}$. Moreover, for large m, $n \approx m/k$. This combined with Equation (5), one can write that, for large m,

$$T^{\mathrm{DD}}_{\mathrm{MC}} \leq \frac{0.38}{w_{\max}}\sqrt{\frac{\mathrm{A}}{m/k}} + \frac{\pi\rho}{2w_{\max}k}. \tag{6}$$

The right hand side of Equation (6) is minimized when $k = 4.09\left(\frac{\rho}{\sqrt{A}}\right)^{2/3} m^{1/3}$. Substituting this into Equation (6), one arrives at the result.

Remark 4.4. Theorem 4.4 and Theorem 3.1 imply that $T^{\mathrm{DD}}_{\mathrm{opt}}$ belongs to $\Theta(1/m^{1/3})$. Moreover, Theorem 4.4 and Theorem 4.2 together with Equation 1 imply that $T^{\mathrm{DD}}_{\mathrm{MC}}/T^{\mathrm{DD}}_{\mathrm{MS}} \to 0^+$ as $m \to +\infty$. Hence, asymptotically, the MC algorithm outperforms the MS algorithm and is within a constant factor of the optimal.

5 Conclusion

In this paper, we considered a coverage problem for a mobile robotic network modeled as double integrators and differential drives. We observe that the optimal algorithm for omni-directional robots is a reasonable solution for sparse networks of double integrator or differential drive robots. However, these algorithms do not perform well for large networks because they don't take into consideration the effect of motion constraints. We propose novel algorithms that are within a constant factor of the optimal for the DI as well as DD robots and prove that the coverage cost for both of these robots is of the order $1/m^{1/3}$.

In future, we would like to obtain sharper bounds on the coverage cost so that we can make meaningful predictions of the onset of reconfiguration in terms of system parameters. It would be interesting to study the problem for non-uniform distribution of targets and for higher intensity of arrival. Also, this research opens up possibilities of reconfiguration due to other constraints, like sensors (isotropic versus anisotropic), type of service requests (distributable versus in-distributable), etc. Lastly, we plan to apply this research in understanding phase transition in naturally occurring systems, e.g., desert locusts [4].

References

1. Agarwal, P.K., Sharir, M.: Efficient algorithms for geometric optimization. ACM Computing Surveys 30(4), 412–458 (1998)
2. Bertsimas, D.J., van Ryzin, G.J.: A stochastic and dynamic vehicle routing problem in the Euclidean plane. Operations Research 39, 601–615 (1991)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

4. Buhl, J., Sumpter, D.J., Couzin, I.D., Hale, J., Despland, E., Miller, E., Simpson, S.J.: From disorder to order in marching locusts. Science 312, 1402–1406 (2006)
5. Cortés, J., Martínez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation 20(2), 243–255 (2004)
6. Drezner, Z. (ed.): Facility Location: A Survey of Applications and Methods. Springer Series in Operations Research. Springer, New York (1995)
7. Enright, J., Savla, K., Frazzoli, E.: Coverage control for teams of nonholonomic agents. In: IEEE Conf. on Decision and Control, pp. 4250–4256 (2008)
8. Kansal, A., Kaiser, W., Pottie, G., Srivastava, M., Sukhatme, G.: Reconfiguration methods for mobile sensor networks. ACM Transactions on Sensor Networks 3(4), 22:1–22:28 (2007)
9. Okabe, A., Boots, B., Sugihara, K., Chiu, S.: Spatial tessellations: Concepts and Applications of Voronoi diagrams, 2nd edn. Wiley Series in Probability and Statistics. John Wiley & Sons, Chichester (2000)
10. Papadimitriou, C.H.: Worst-case and probabilistic analysis of a geometric location problem. SIAM Journal on Computing 10(3) (August 1981)
11. Psaraftis, H.: Dynamic vehicle routing problems. In: Golden, B., Assad, A. (eds.) Vehicle Routing: Methods and Studies. Studies in Management Science and Systems. Elsevier, Amsterdam (1988)
12. Savla, K., Frazzoli, E.: On endogenous reconfiguration for mobile robotic networks. Technical report (2008), http://arxiv.org/abs/0807.2648
13. Zemel, E.: Probabilistic analysis of geometric location problems. Annals of Operations Research 1(3), 215–238 (1984)

Simultaneous Control of Multiple MEMS Microrobots

Bruce R. Donald, Christopher G. Levey, Igor Paprotny, and Daniela Rus

Abstract. We present control algorithms that implement a novel planar microassembly scheme using groups of stress-engineered microrobots controlled through a single global control signal. The global control signal couples the motion of the devices, causing the system to be highly underactuated. Despite the high degree of underactuation, it is desirable that each robot be independently maneuverable. By exploiting differences in the designs and the resulting electromechanical interaction with the control signal, the behavior of the individual robots can be differentiated. We harness this differentiation by designing the control signal such that some devices remain confined in small circular orbits (limit cycles), while the non-orbiting robots perform work by making progress towards the goal. The control signal is designed to minimize the number of independent control voltage levels that are used for independent control, allowing us to maximize the number of simultaneously controllable devices.

Our algorithms were tested on systems of fabricated untethered stress-engineered MEMS microrobots. The robots are 240–280 μm $\times$ 60 μm $\times$ 7–20 μm in size and are controlled in parallel (simultaneously) within the same operating environment. We demonstrated the feasibility of our control algorithms by accurately assembling 5 different types of planar microstructures.

Bruce R. Donald
Department of Computer Science, Duke University, Durham, NC, USA
Department of Biochemistry, Duke University Medical Center, Durham, NC, USA
e-mail: brd+wafr08@cs.duke.edu

Christopher G. Levey
Thayer School of Engineering, Dartmouth College, Hanover, NH, USA

Igor Paprotny
Department of Computer Science, Dartmouth College, Hanover, NH, USA

Daniela Rus
Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA

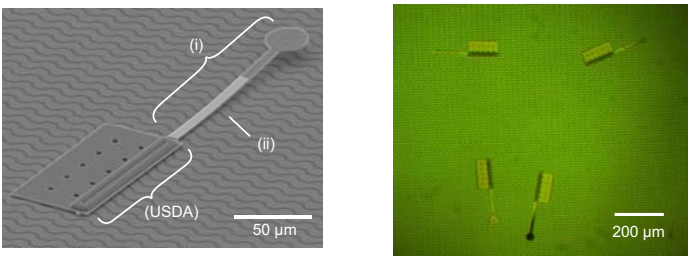
G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 69–84.
springerlink.com

1 Introduction

The operation of multiple untethered microrobotic devices have many potential applications in medicine, surveillance, and assembly [13]. In this work we use the term *microrobot* to denote an untethered robot that fits strictly within a 1 mm^3 volume.

In [6], we presented a globally controllable 240 μm $\times$ 60 μm $\times$ 10 μm mobile stress-engineered MEMS microrobot. A scanning-electron micrograph of a stress-engineered MEMS microrobot, and an optical micrograph of four microrobots during an assembly experiment, are shown in Figure 1. The robot has a *steering arm* (Fig. 1a(i)), and operates on a planar substrate (its *operating environment*), which couples an external control and power delivery signal to the device regardless of its pose. Simultaneous control of multiple microrobots within a single operating environment is desirable, but presents a significant challenge when only a single, global control signal can be used to control all the devices: the resulting system is highly underactuated.

We show that designing microrobots with different steering arms that respond to different voltages allows us to independently maneuver multiple microrobots. Each steering arm has two *transition voltages* (voltage levels at which the steering arm is either pulled down or released from the substrate). We show that it is sufficient that each steering arm has a unique transition voltage pair, i.e. the combination of both transition voltages is unique. Details of fabrication, designs and testing were reported in [9]. In this work, we present the theory for designing the control signal to enable simultaneous control of multiple microrobots for microassembly. Enabling our control algorithms are new theorems that minimize the control bandwidth requirements. These theorems were essential for controlling multiple untethered microrobots to move and assemble independently, and are not covered in [9].



(a) (b)

Fig. 1. Scanning-electron micrograph of a stress-engineered MEMS microrobot (a), and optical micrograph of four microrobots (b). **a:** The microrobot consists of an untethered scratch-drive actuator (USDA) that provides forward motion, and a curved steering-arm actuator (i) that determines whether the robot moves in straight-line motion or turns. A lithographically-patterned chrome-layer defines the curvature of the steering arm (ii). **b:** Four different stress-engineered microrobots on their operating environment. The robots are differentiated by the design of their steering-arm actuators.

2 Related Work

Small, often completely autonomous mobile robotic devices, with size on the order of centimeters [12] are called *miniature robots*. Miniature robots containing microfabricated components [11] are called MEMS robots. The components of MEMS robots are often manufactured separately and then assembled to form complete systems. The size of such MEMS robots ranges from hundreds of micrometers to several centimeters. In [6, 9] and this paper, we use the term *microrobot* to denote mobile untethered MEMS robots with their dimensions strictly confined within a 1 mm^3 cube. Other microrobotic systems include a magnetic micro-needle that is actuated using magnetic fields [15].

Microassembly is one potential application for cooperating microrobotic systems. Currently, the assembly of micro-components is performed either using serial robotic manipulators with micro-scaled end-effectors [3], using a distributed manipulation approach, [1], or through parallel but less controllable self-assembly (SA) [14]. In contrast with SA, our microassembly scheme relies on intersecting trajectories, rather than component affinity, to promote structure aggregation. The concept of selective response of individual units to a global, broadcasted, control signal (Global Control, Selective Response (GCSR) [5]) is common in micro- or nanoscale biological systems. Bretl [2] presented a related theoretical motion planning approach for systems of robots with limited controllability, showing that even simple devices controlled through a global signal can perform useful tasks.

3 Stress-Engineered MEMS Microrobot

All the control and assembly algorithms presented in this paper are implemented using groups of parallel-actuated stress-engineered MEMS microrororobots [6]. A stress-engineered microrobot has two actuated internal degrees of freedom (DOF); an untethered scratch-drive actuator (USDA) [7] that provides forward locomotion, and a steering-arm actuator that determines whether the robot moves in a straight-line or turns. The steering-arm consists of a cantilever beam with a circular pad and a .75–1.2 μm deep dimple. The cantilever beam is curved (out-of-plane) using a stress-engineering process [9], which determines the deflection of the steering arm. The microrobot operates on fields of zirconia-insulated interdigitated electrodes. When a voltage is applied across these electrodes, the electrodes and the conductive microrobot chassis form a capacitive circuit inducing an electric potential on the microrobot body. This voltage (waveform) is varied over time to provide power to the untethered scratch-drive actuator and to control the state of the steering-arm. This waveform is called the *control waveform*. Figure 2(a) illustrates one cycle of the control waveform. The waveform is divided into a *control cycle*, containing one or more *control pulses* ($V_{a,j}$), that sets the state of the steering-arm actuator, and a *power-delivery cycle* that provides power to the scratch-drive actuator. For a single robot, a specific control waveform, defined through the voltage triple $(V_{a,1}, V_b, V_s)$, is called a *control primitive*; one control primitive causes the robot to turn, while another causes it to move in a straight line.

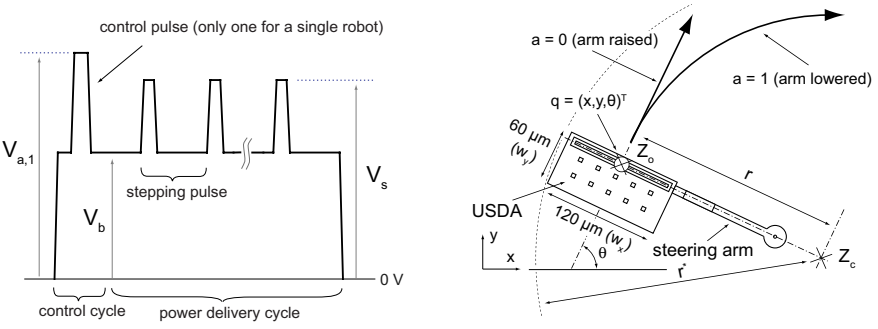


Fig. 2. The control and power delivery waveform for a single stress-engineered MEMS microrobot (a), and kinematics of the stress-engineered MEMS microrobot (b).

Similar to an electrostatic cantilever beam, the steering arm of the stress-engineered MEMS microrobot has two distinct voltage-levels at which it abruptly changes state. These voltages are called the *transition voltages*. While clearly the state of our microrobot includes the state of the steering arm and the state of the scratch-drive actuator, for the purpose of this section it suffices to consider only the states of the steering-arm actuators, which we call the robots' *control states*. Consequently, a single actuated stress-engineered microrobot can be in one of only two control states; the steering-arm can be either raised (0) or lowered (1). When the voltage supplied to the robot reaches the steering-arm's *snap-down* transition voltage (V_d), the arm is pulled into contact with the substrate. When the voltage is reduced past the *release* transition voltage (V_u), the arm is released from the substrate. The transition voltages are a function of the design of the individual steering-arm actuators: for example, smaller air gaps or larger steering pads reduce both V_u and V_d. Microrobots with identical steering-arms are classified as belonging to the same *microrobot species*. The difference between the snap-down and release voltage of a steering-arm is called the *hysteresis gap*.

The kinematics of our robot is illustrated on Fig. 2(b). The configuration of the robot is given by the vector $q = (x, y, \theta)^T$ in configuration space (C-space). The configuration of the robot is measured at the point Z_o in the middle of its bushing. The robot moves like a Dubins car that can turn in one direction only. The velocity of the robot is $\dot{q} = v\left(\sin\theta, \cos\theta, \frac{ah}{r}\right)^T$, where $h \in \{-1, 1\}$ and denotes whether the steering arm is on the right or the left side, v is the velocity of the scratch-drive actuator, r is the turning radius and $a \in \{0, 1\}$ is the state of the steering arm (0 = up, 1 = down). The velocity v can be varied by changing the frequency of the stepping pulses, however for the remainder of this paper we will consider v to be a positive constant (positive because the robot can not back up). It follows that the robot is not *small-time locally controllable* (STLC).

4 STRING Theory: Control Signal Design for Parallel Actuation

Multiple microrobots operating simultaneously within the same environment receive the same, single, global control signal. Here we show how to design the control signal, constructing a set of control primitives that allows us to independently maneuver multiple microrobots when implementing microassembly.

Suppose we have a set of m control primitives that can be applied through the operating environment to control a set of n microrobots. A mapping between the control primitives and the motion of the individual devices is expressed through a *control matrix* M of size $(n \times m)$, where each entry (i,j) contains the control state of microrobot j during the application of the control primitive i. The control matrix describes the coupling of the microrobot motion as a function of the control signal, linking the electromechanical functionality of the steering-arms with the application of the control primitives. An example of the control matrix is shown in Eq. (3) on p. 76. Our control strategies (described in sec. 5) require that the control matrix is structured such that the robots progressively start turning as the control primitives with higher index i are applied. We prove that such a control matrix can be constructed with the smallest number of independent voltage levels in the control signal necessary to achieve independent control, allowing us to maximize the number of simultaneously controllable microrobots.

For a system of n microrobots, let $V_{d,i}$ and $V_{u,i}$ denote the snap-down and release voltages of microrobot i. Let V_Ω be the breakdown voltage of the operating environment, V_{flx} be the minimum voltage at which the backplate of the USDA generates enough flexure to produce a forward step of the scratch-drive actuator, and V_{rel} be the maximum voltage at which the backplate of the USDA relaxes, allowing it to take a step forward (each stepping pulse must cycle through V_{flx} and V_{rel} in order to supply power to the USDA). The snap-down and release voltages of each device must satisfy to the following constraints:

1. $V_{d,i} < V_\Omega$: snap-down voltage cannot exceed the break-down voltage of the operating environment.
2. $V_{d,i} > V_{u,i}$: dictated by the electromechanics of cantilever beams.
3. $V_{rel} > V_{u,i}$ for all i : ensures the microrobot can receive power during all the control states.
4. $V_{flx} \leq_i^{min} (V_{d,i})$: Ensures that the USDA flexes sufficiently to produce forward motion during the power delivery cycle.

The *control voltage bandwidth* ξ of a microrobot system is the number of independent electromechanically-addressable transition voltage levels that can be used for control. ξ depends on four parameters: the break-down voltage of the operating environment, V_Ω, the inherent variability of the power coupling between the robot and the underlying substrate, the precision of the fabrication process, and the minimum range of voltages required to reliably power the USDA, V_{SDA} ($V_{SDA} = V_{flx} - V_{rel}$). The variability in the power coupling causes deviations in the potential induced between the steering arm and the substrate, while inaccuracies

in the fabrication process cause deviations in the transition voltages of the steering arms. Let δ_v be the maximum deviation of the transition voltage manifested during the microrobot operation, determined by these two parameters. We define two transition voltages to be *significantly independent* if they are separated by at least $2\delta_v$. Note that, although in general, $V_{SDA} = V_{flx} - V_{rel}$, it is possible for the stepping pulse to overlap with the lowest snap-down voltage, $\min_{i \in Z_n} V_{d,i}$. Consequently, we define V'_{SDA} as the *additional* control voltage bandwidth that is required by the power delivery cycle to ensure reliable actuation of the USDA, as follows:

$$V'_{SDA} = \begin{cases} 0, & \text{if } V_{flx} - V_{rel} \leq 2\delta_v, \\ V_{flx} - V_{rel} - 2\delta_v, & \text{otherwise.} \end{cases} \quad (1)$$

The control voltage bandwidth of a microrobot system is then $\xi = \left\lfloor \frac{V_\Omega - V'_{SDA}}{2\delta_v} \right\rfloor$, assuming V_{flx} and V_{rel} can vary by at least $2\delta_v$ (otherwise $\xi = \left\lfloor \frac{V_{rel}}{2\delta_v} \right\rfloor + \left\lfloor \frac{V_\Omega - V_{flx}}{2\delta_v} \right\rfloor$). How much of ξ is actually used to control the microrobotic system is related to the number of accessible control states of the steering arm actuators. Define the *control voltage bandwidth requirement*, ξ_n, of a n-microrobot system as the number of independent voltage levels necessary to achieve independent control during our microassembly application. Clearly it must hold that $\xi_n \leq \xi$. In general, a microrobotic system with fewer accessible control states has a lower control voltage bandwidth requirement. More specifically, the accessibility of the control states depends on the relation between the hysteresis gaps of the individual robots.

Consider a system of two microrobots, D_1 and D_2, with steering arms that have *Nested Hysteresis Gaps* (NHG) (first proposed in [6]). Fig. 3(a) shows the relation between the transition voltages for such a system. The snap-down and release voltages are shown as circles and rectangles, respectively. Without loss of generality, we consider only significantly independent voltage levels of the control signal (labeled as V_α-V_δ on Fig. 3). Fig. 3(b) shows the programming cycles for the four control primitives that access the four control states (11), (10), (01) and (00) (we assume $V_b = V_{rel}$). More generally, we classify the system of n steering arms, sorted according to ascending $V_{d,i}$, as having NHG when $(V_{d,i} + 2\delta_v < V_{d,j})$ and $(V_{u,i} - 2\delta_v > V_{u,j})$, for all $i < j$. NHG systems can access all 2^n control states. However each device requires two unique control voltage levels, and so the control voltage bandwidth requirement of this system is $\xi_n = 2n$.

NHG are sufficient but not necessary, to control multiple devices during assembly. Consider a two-robot system where the hysteresis gaps of the robots are not nested, as shown in Fig. 4(a). In this particular system, only three control states are electromechanically accessible. The control cycles (control pulses only) that access all three control states, (00), (10) and (11), are shown in Fig. 4(b). Control state (01) can not be accessed, because pulling down the steering-arm of D_2 also pulls down the steering-arm of D_1, and the steering-arm of D_1 can not be released without also releasing the arm of D_2.

In general, an n-microrobotic system, first sorted according to ascending values of V_d, and then sorted according to ascending values of V_u, has non-nested hysteresis

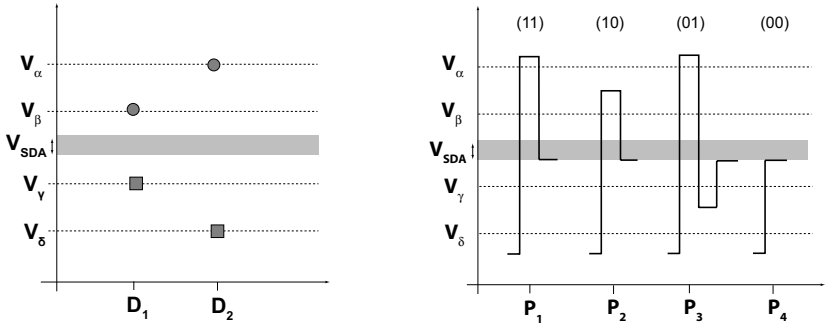


Fig. 3. Transition voltages for a system of two microrobots with nested hysteresis gaps (NHG) (**a**), and the control cycles for the four control primitives that access all control states for the system (**b**).

gaps if $(V_{d,i} \leq V_{d,j})$ and $(V_{u,i} \leq V_{u,j})$, for all $i < j$. However, in the case when $|V_{d,i} - V_{d,j}| < 2\delta_v$ and $|V_{u,i} - V_{u,j}| < 2\delta_v$, the behavior of robots i and j is indistinguishable, and such two devices cannot be controlled independently. We call such two robots a *degenerate pair*. Let a *STRIctly Non-nested hysteresis Gaps* (STRING) system be a non-nested hysteresis gap system with no degenerate pairs of devices.

Lemma 4.1. *An n-robot STRING system has exactly* $n+1$ *accessible control states.*

The complete proof of Lemma 4.1 is provided in Supplementary Material (**SM**) **Section 1** available online in Ref. [10]. We now construct the control primitives and corresponding control matrix that can access the $n+1$ control states of a n-robot STRING system. The ordering of the robots is determined by the transition voltages of the steering arms, i.e., the robots must be primarily sorted according to

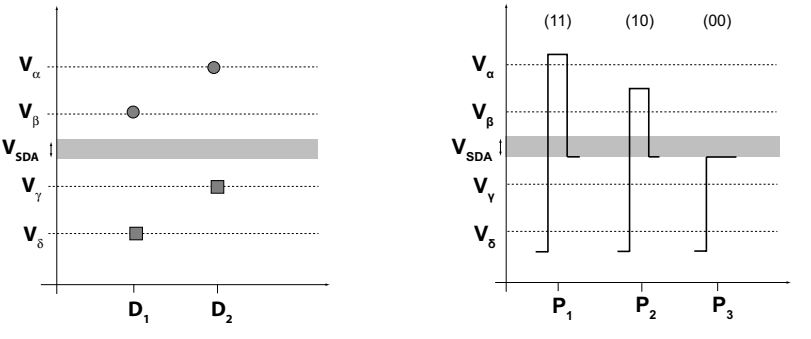


Fig. 4. Transition voltages for a system of two microrobots with non-nesting hysteresis gaps (**a**), and the control cycles for the control primitives that access three of the control states for the system.

increasing order of $V_{u,i}$ and secondarily sorted according to increasing order of $V_{d,i}$. We construct the control primitive P_j such that it snaps down the arms of devices D_i for $i \leq j$, and releases the arms of devices D_i for all $i > j$. P_j is defined by a control cycle containing two control pulses, $P_j = (V_{a,1}, V_{a,2})$, assume $V_b = V_{rel}$ and $V_s = V_{flx} \leq V_{d,1}$. Consider the STRING system shown on Fig. 5(a), where $V_\alpha, \cdots, V_\varepsilon$ represent significantly independent control voltage levels. We define P_j as:

$$P_j = \begin{cases} (V_0, V_0), & \text{if } j = 0; \\ (V_{d,j}, V_{u,j+}), & \text{if } j \in Z_{n-1}; \\ (V_{d,n}, V_{d,n}), & \text{if } j = n, \end{cases} \tag{2}$$

where $V_{u,j+} = V_{u,j} + 2\delta_v$. In practice, $V_{u,j+}$ is the next significantly independent release voltage above $V_{u,j}$. Also, note that in order for $V_{d,j}$ to cause *reliable* snap down, it must be δ_v *above* the designed (nominal) $V_{d,j}$ level. Correspondingly, $V_{u,j}$ must be δ_v *below* the designed (nominal) $V_{u,j}$ level to ensure reliable steering arm release.

The first control pulse ($V_{a,1}$) snaps down the steering arms of all the devices D_i, $i \in Z_j$, as well as any devices D_k, $k > j$ with $V_{d,k} = V_{d,j}$. The second control pulse ($V_{a,2}$) releases all the devices D_k, $k > j$ that were snapped down by the first control pulse, because in the case when $V_{d,k} = V_{d,j}$, it must hold that $V_{u,j} < V_{u,j+} \leq V_{u,k}$. An example control cycle is also shown in Fig. 5(a).

The $n+1$ control primitives generated by P_j form a $(n+1) \times n$ control matrix M. An example of such control matrix for four devices is:

$$M = \begin{pmatrix} 0\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 1\,1\,1\,0 \\ 1\,1\,1\,1 \end{pmatrix}. \tag{3}$$

We refer to M as the *STRING control matrix*, the $n+1$ control primitives contained in M as the *STRING control primitives*, and the $n+1$ control states accessible using these control primitives as the *STRING control states*. Note that because adding a new control state to a STRING system requires the addition of another independent voltage level (per Lemma 4.1), the control bandwidth requirement for a STRING system is $\xi_n = n+1$.

Lemma 4.2. *Any stress-engineered n-microrobotic system with no degenerate pairs of robots can be sorted such that all $n+1$ STRING control states are accessible.*

Proof. By construction. Consider a microrobot system with k independent snap-down voltages, and ℓ independent release voltages. Assuming no degenerate pairs of devices, it follows that $n \leq k\ell$. In the case when $n = k\ell$, the n steering arms encode all the possible $k\ell$ combinations of snap-down and release voltages. We call such system for *electromechanically saturated* (ESat). We can enumerate the hysteresis gaps for an ESat system given both k and ℓ. Consider an ESat system, sorted primarily according to increasing release voltage $V_{u,i}$ and secondarily sorted according

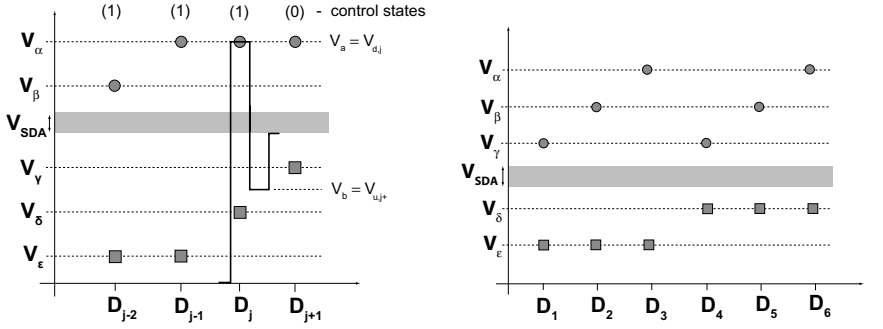


Fig. 5. Construction of a STRING control matrix (a), An example of an ESat system with $k = 3$ and $l = 2$ (b).

to increasing snap-down voltage $V_{d,i}$. Fig. 5(b) shows such system when $k = 3$ and $l = 2$. Note that sorting ensures a monotonic increase of $V_{u,i}$ with increasing index i. For such an order, there exists a recursive formula, shown in Eq. (4), that generates all $n+1$ STRING control primitives. The control cycle for each control primitive defined by Eq. (4) contains a sequence of up to $2n$ control pulses (in contrast with 2 control pulses in Eq. (2)). Again, we assume $V_b = V_{rel}$ and $V_s = V_{flx} \leq V_{d,1}$.

$$P_j = \begin{cases} (V_0, V_0), & \text{if } j = 0; \\ \left(P_{j-1}, V_{d,j}, V_{u,j+}\right), & \text{if } j \in Z_n. \end{cases} \tag{4}$$

P_j generates $n+1$ control primitives that form a STRING control matrix, by causing devices D_i $(i \leq j)$ to be in state 1, while robots D_i $(i > j)$ are in state 0. Consider the base case P_0, where all D_j, $(j \in Z_n)$ is in state 0. We make the inductive argument that after application of the recursive part of P_j, P_{j-1}, all $D_1, \cdots, D_{j-1}$ robots are in control state 1. It is clear that $V_{d,j}, V_{u,j+}$, $(j \in Z_n)$, will snap down D_j. The $V_{u,i}$-then $V_{d,i}$-sorting implies that, for a device D_k, $k > j$, only two case are possible with respect to its transition voltages: (a) $V_{d,j} < V_{d,k}$ (e.g. $j = 2$ and $k = 3$ in Fig. 5(b)), or (b) $V_{u,j} < V_{u,j+} \leq V_{u,k}$ (e.g. $j = 3$ and $k = 5$ in Fig. 5(b)). It is clear that in case (a), $V_{d,j}$ sets D_j to state 1, while D_k, $k > j$ is in state 0. The sorting ensures that any previously applied control primitive P_i, $i < j$ with $V_{a,1} \geq V_{d,k}$ (which also inadvertently snaps down the arm of D_k) must have have been followed by a control pulse $V_{a,2} \leq V_{u,k} - 2\delta_v$ (which would release the arm of D_k). In case (b), $V_{u,j+}$ releases any devices D_k, $k > j$. □

Note that because the devices are sorted according to $V_{i,u}$ and $V_{d,i}$, Eq. (4) also holds for any microrobotic system, even one that is not ESat.

Theorem 4.1. *A system of n STRING microrobots contains the minimum number $(n+1)$ of electromechanically accessible control states of any stress-engineered microrobot system without degeneracy.*

Proof. Per Lemma 4.1, an n-microrobot STRING system has exactly $n+1$ accessible control control states, and by Lemma 4.2, any n stress-engineered microrobotic system without degenerate pairs of robots contains at least $n+1$ control states. □

Theorem 4.2. *An algorithm that solves the gross motion planning problem (i.e. finds the control sequence S) for a STRING system can be applied to solve the gross motion planning problem for any non-degenerate system of stress-engineered microrobots.*

Proof. A consequence of Lemma 4.2: a STRING control matrix can be constructed for any n-stress-engineered microrobotic system. □

Theorem 4.2 allows us to further reduce the control bandwidth requirements for a microrobotic system. The control voltage bandwidth requirement for a microrobot system with k independent snap-down voltage levels and ℓ independent release voltage levels is $\xi_n = k+\ell$. In an ESat system, $n = k\ell$. It follows that n is maximized when $\ell = k = \xi_n/2$, and $n = \xi_n^2/4$. We call such system *symmetric electromechanically saturated*, or SESat. As a consequence, the control bandwidth requirement for an ESat system is $\xi_n \geq 2\lceil\sqrt{n}\rceil$, while it is $\xi_n = 2\lceil\sqrt{n}\rceil$ for an SESat system. It follows that an SESat system maximizes the number of simultaneously controllable microrobots.

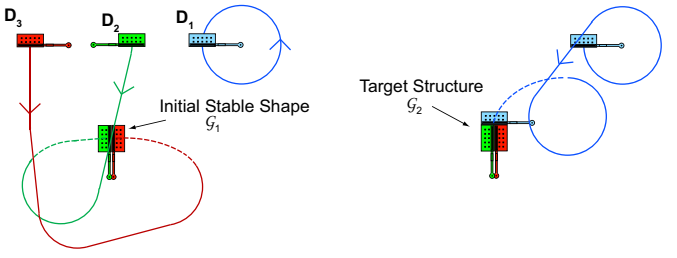
5 STRING Parallel Control for Microroassembly

We now use the structure of the STRING matrix M to decouple the motion of the individual robots by reducing the problem of controlling n microrobots in parallel to the problem of controlling only two robots in parallel, followed by sequential control of single devices. We can perform this reduction whenever M has the form of a STRING control matrix by constraining some robots to orbit along limit cycles without making progress towards the goal. Note that our planning algorithms are not fully general and require a minimum separation between the orbiting robots.

The assembly is performed in $n-1$ steps. Fig. 6(a) and (b) show the assembly for a system of $n = 3$ microrobots assembling the shape $\mathscr{G}_2$ (from Fig. 9). The STRING control matrix M for this system is shown on Eq. (3) p. 76. Further details of the reduction are provided in **SM** [10] **Section 2**.

5.1 Microassembly Step 1

The two microrobots with the highest index, D_{n-1} and D_n are maneuvered to assemble an initial stable shape, $\mathscr{G}_1$. Only control primitives P_{n-2}, P_{n-1} and P_n are used to control D_{n-1} and D_n. P_{n-2}, P_{n-1} and P_n cause only turning motion in robots D_i, ($i < n-1$, e.g. robot D_1 in Fig. 6(a)), and consequently these robots remain in circular orbits. The assembly of $\mathscr{G}_1$ takes place in two stages, because even though both D_{n-1} and D_n move simultaneously, error correction is only performed on a single microrobot at any given time.



(a) (b)

Fig. 6. Example assembly of three microrobots in two steps. **a:** Step 1, the assembly of the initial stable shape. **b:** Step 2, docking of D_1 to form the target shape.

5.2 Microassembly Step $2, \cdots, n-1$

A single robot, D_i, is maneuvered to dock with the assembling shape (See Fig. 6(b)). Only control primitives P_i and P_{i-1} are used to maneuver D_i, causing the devices D_j, $(j < i)$, to follow circular orbits. Because the target shape ($\mathcal{G}_k$ out of a set of p available target shapes, $\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_P\}$) is assembled in the descending order of the device index i, the devices D_j, $(j > i)$ are immobilized by compliance within the assembling shape.

5.3 EDR-Based Control

We used the theory of Error Detection and Recovery (EDR) [4] to construct control strategies that allow us to reliably maneuver the microrobots to target regions in the presence of control error. Our control strategies are based on progressive execution and replanning of the microrobot trajectories during each of the steps described above. However, in order to increase the precision of the assembly, we switch to a fine-motion control strategy as the robots approach their docking configurations (the dashed trajectories in Fig. 6). The fine-motion trajectory is based on interpolated turning, and allows us to sacrifice precise control of the incident angle for the docking robot in favor of precise control of its docking location. The accumulating error in rotation is later reduced using compliance. Details regarding the control strategies are provided in **SM** [10] **Section 3** for the interested reviewer.

6 Experimental Results

The control strategies presented in this paper have been tested experimentally on groups of fabricated stress-engineered microrororobots. This section uses experimental data that has been previously reported in [9], but describes how this data validates the algorithms above, and gives the explicit construction of the control matrices necessary to replicate the results.

6.1 Fabrication of STRING Microrobots

We fabricated 15 microrobots classified into five microrobot *species*. The microrobot species are differentiated through the designs of their steering arm actuators. Fig. 7 shows a scanning-electron micrograph of a microrobot from each species. The species are designed such that multiple copies can be reproducibly fabricated despite the inherent variability of our fabrication process.

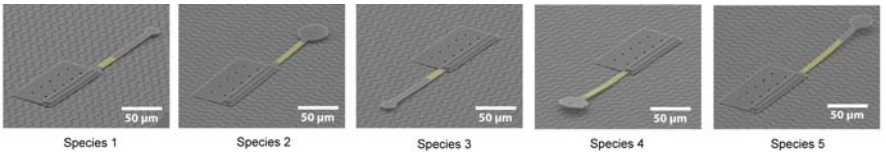


Fig. 7. *Reprinted with permission from [9].* Scanning-electron micrographs of the five microrobot species used to implement microassembly. Yellow color is used to highlight the areas of the steering-arms covered by the chrome layer.

The steering-arm designs were determined based on closed-form equations [6], finite-element analysis and empirical data, such that the transition voltage, ($V_{d,i}$ and $V_{u,i}$), for all the robots are reproducibly confined to the ranges shown in Fig. 8(a). Snap-down voltages ($V_{d,i}$) are marked by circles, while the rectangles denote the release voltages ($V_{u,i}$). The ranges are marked by a vertical bar, with two dots signifying that the lower or upper bound is not fixed or measured. Groups of robots from species 1,3,4,5 and 2,3,4,5 form STRING systems, while species 1 and 2 form a degenerate pair. The exact parameters of the steering-arms defining all five species are described in [9]. The waveforms (control pulse and three power delivery pulses only) of the five control primitives used to control the four-robot STRING groups are show in Fig. 8(b). Average V_a, V_b and V_s voltage levels are shown. The actual voltage-levels used to control the individual groups of microrobots could vary by up to $\pm$ 10 V.

6.2 Microassembly

We applied the control algorithms to groups of four STRING microbots, generating a total of 14 planar structures. The assembled structures belong to five types of target shapes, labeled $\mathscr{G}_1 - \mathscr{G}_5$. Optical micrographs of microstructures for each type of target shape are shown on Fig. 9.

The robots were operated on a 2 mm^2 environment, and their position was registered using a digital video-camera attached to an optical microscope with a 6.7 $\times$ objective lens. Table 1 shows the average match (portion of the target structure covered by the assembled shape) for the five assembled shapes, $\mathscr{G}_1 - \mathscr{G}_5$. The assembly experiments were conducted starting from two different classes of initial configurations: $\mathscr{R}_1$ – robots are arranged along the corners of a rectangle with sides 1 by

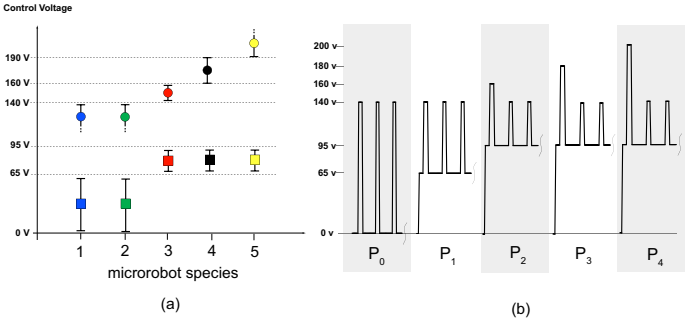


Fig. 8. Transition voltage ranges (a) and corresponding control primitives (b) used to control the five microrobot species.

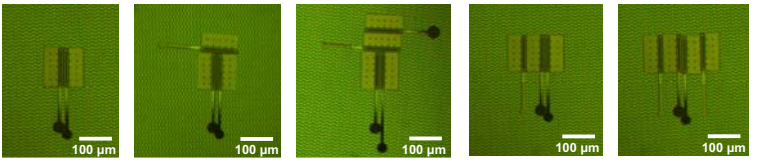


Fig. 9. *Reprinted with permission from [9].* Optical micrographs of five types of target shapes assembled using our microrobots.

0.9 mm, all devices oriented along the y-axis (see Fig. 10(a) for a representative example), and $\mathscr{R}_2$ – robots are arranged in a line with average separation of 360 μm, and with variable orientation. The initial position of the microrobots was set using batch-transfer structures called *transfer-frames* [9] and microprobes. We used common geometric shapes (a line and a rectangle) to demonstrate the ability to achieve successful assembly from arbitrary different initial configurations.

The results in Table 1 do not include completely failed assemblies. We recorded a 11% failure rate during the consecutive assembly of nine structures over the course

Table 1. Precision of Microassembly.

	Goal Configurations					
Initial Configurations	$\mathscr{G}_1$	$\mathscr{G}_2$	$\mathscr{G}_3$	$\mathscr{G}_4$	$\mathscr{G}_5$	**Average**
$\mathscr{R}_1$	$96 \pm 4\%$ (3 runs)	$98 \pm 3\%$ (2 runs)	$96 \pm 2\%$ (2 runs)	96% (1 run)	93% (1 run)	$96 \pm 3\%$ (3 runs)
$\mathscr{R}_2$	$99 \pm 2\%$ (2 runs)	98% (1 run)	93% (1 run)	89% (1 run)	na	$95 \pm 4\%$ (2 runs)
Average	$97 \pm 3\%$ (5 runs)	$98 \pm 2\%$ (3 runs)	$95 \pm 2\%$ (3 runs)	$93 \pm 5\%$ (2 runs)	93% (1 run)	$96 \pm 3\%$ (5 runs)

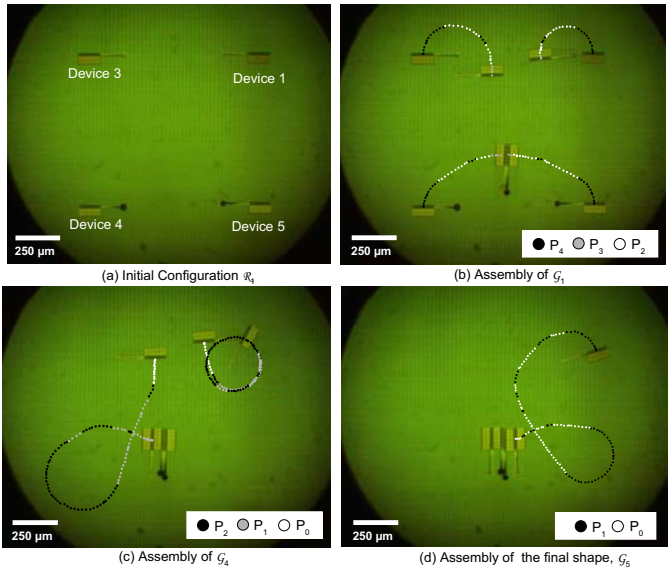


Fig. 10. *Reprinted with permission from [9].* Composite optical micrograph of experimental assembly-data using devices from species 1,3,4 and 5. The devices are labeled according to the number of their respective species.

of three assembly experiments. This reflects that the assembly of one of the nine structures failed due to the loss of stability of an intermediate structure, which was attributed to an initial unfortunate misalignment between the microrobots forming the intermediate assembly. Fig. 10 illustrates a representative assembly experiment. In this experiment, the target shape $\mathcal{G}_5$ is generated via the assembly of $\mathcal{G}_1$ and $\mathcal{G}_4$. The experiment terminated when all four microrobots were successfully incorporated in the assembled structure. A movie of this assembly experiment is available online at [8].

7 Conclusions

We presented novel control algorithms for implementing planar microassembly using groups of stress-engineered microrobots. The experimental data, reprinted from [9], indicates the feasibility of our algorithms, and represents the first implementation of a multi-microrobotic system. This work presents the planning and control challenges to achieve independent microrobot control and implement the microassembly scheme.

Our control scheme minimizes the control voltage bandwidth requirements of an n-microrobotic system. The sub-linear ($O(\sqrt{n})$) control voltage bandwith requirement is a large improvement over $O(n)$ in our previously proposed approach [6]. Reducing the control voltage bandwidth requirement was the enabling technology

that allowed us to experimentally demonstrate simultaneous control of four devices. We used the STRING control matrix to implement planning and control algorithms that reduce parallel motion of n robots to parallel motion of only two robots, followed by sequential motion of single devices.

We believe the concept of selective response to a global control signal (GCSR [5]) will be important for controlling future multi-microrobotic systems. GCSR is common in biological, micro, and nano-scale systems, and may be the paradigm of choice for controlling groups of micro, and nano-robots.

Acknowledgements

This work was supported by grant number GM-65982 to B.R.D. from NIH, and 2000-DT-CX-K001 to B.R.D., from the Office for Domestic Preparedness, Department of Homeland Security, USA. The electron micrographs were taken at the Dartmouth Ripple Electron Microscopy Laboratory, with the help of C. P. Daghlian. We thank U. J. Gibson for the use of equipment in her lab, and for many helpful discussions. We further thank D. Balkcom, C. Bailey-Kellogg, A. Lebeck and K. Böhringer for their advice and suggestions.

References

1. Böhringer, K.-F., Donald, B.R., Mihailovich, R.R., MacDonald, N.C.: A theory of manipulation and control for microfabricated actuator arrays. In: Proceedings of the IEEE Workshop on Micro Electro Mechanical Systems MEMS (January 1994)
2. Bretl, T.: Control of many agents using few instructions. In: Proceedings of Robotics: Science and Systems 2007 conference, Atlanta, GA (2007)
3. Dechev, N., Cleghorn, W.L., Mills, J.K.: Tether and joint design for micro-components used in microassembly of 3D microstructures. In: Proceedings of SPIE Micromachining and Microfabrication, Photonics West 2004 (January 2004)
4. Donald, B.R.: Error Detection and Recovery in Robotics. LNCS, vol. 336. Springer, Heidelberg (1989)
5. Donald, B.R.: Building very small mobile micro robots. Inaugural Lecture, Nanotechnology Public Lecture Series, MIT (Research Laboratory for Electronics, EECS, and Microsystems Technology, Laboratories), Cambridge, MA (April 2007), http://mitworld.mit.edu/video/463/
6. Donald, B.R., Levey, C.G., McGray, C., Paprotny, I., Rus, D.: An untethered, electrostatic, globally-controllable MEMS micro-robot. Journal of Microelectromechanical Systems 15(1), 1–15 (2006)
7. Donald, B.R., Levey, C.G., McGray, C., Rus, D., Sinclair, M.: Power delivery and locomotion of untethered micro-actuators. Journal of Microelectromechanical Systems 10(6), 947–959 (2003)
8. Donald, B.R., Levey, C.G., Paprotny, I.: Planar microassembly by parallel actuation of MEMS microrobots – supplementary videos (2008), http://www.cs.duke.edu/donaldlab/Supplementary/jmems08/
9. Donald, B.R., Levey, C.G., Paprotny, I.: Planr microassembly by parallel actuation of MEMS microrobots. Journal of Microelectromechanical Systems 17(3) (2008), doi:10.1109/JMEMS.2008.924251

10. Donald, B.R., Levey, C.G., Paprotny, I., Rus, D.: Simultaneous control of multiple MEMS microrobots – supplementary material. Technical report, Department of Computer Science, Duke University (July 2008), http://www.cs.duke.edu/donaldlab/Supplementary/wafr08/
11. Hollar, S., Flynn, A., Bellew, C., Pister, K.S.J.: Solar powered 10 mg silicon robot. In: Proceedings of the The Sixteenth Annual International Conference on Micro Electro Mechanical Systems, MEMS 2003, Kyoto, Japan, January 19-23, pp. 706–711 (2003)
12. Jianghao, L., Zhenbo, L., Jiapin, C.: An omni-directional mobile millimeter-sized microrobot with 3-mm electromagnetic micromotors for a micro-factory. Advanced Robotics 21(12) (December 2007)
13. Popa, D., Stephanou, H.E.: Micro and meso scale robotic assembly. SME Journal of Manufacturing Processes 6(1), 52–71 (2004)
14. Whitesides, G.M., Grzybowski, B.: Self-assembly at all scales. Science 295, 2418–2421 (2002)
15. Yesin, K.B., Vollmers, K., Nelson, B.J.: Modeling and control of untethered boimicrorobots in fluid environment using electromagnetic fields. The International Journal of Robotics Research 25(5-6), 527–536 (2006)

Simultaneous Coverage and Tracking (SCAT) of Moving Targets with Robot Networks

Luciano C.A. Pimenta, Mac Schwager, Quentin Lindsey, Vijay Kumar,
Daniela Rus, Renato C. Mesquita, and Guilherme A.S. Pereira

Abstract. We address the problem of simultaneously covering an environment and tracking intruders (SCAT). The problem is translated to the task of covering environments with time-varying density functions under the locational optimization framework. This allows for coupling the basic subtasks: task assignment, coverage, and tracking. A decentralized controller with guaranteed exponential convergence is devised. The SCAT algorithm is verified in simulations and on a team of robots.

1 Introduction

Three important problems in the field of cooperative robotics are: (i) environment coverage, (ii) task assignment, and (iii) target (intruder) tracking. These three problems are the basic subtasks that must be solved in the global problem of tracking multiple intruders in a pre-specified region. When the number of intruders and their locations is unknown, the need for environment coverage is important to maximize the probability of detecting them. Task assignment is necessary to define when a given agent should perform a coverage behavior and when it should do intruder tracking. For multiple intruders it is also necessary to allocate intruders to tracking agents. After assigning an intruder-tracking task to an agent, it is desirable to use control laws with guaranteed convergence.

A distributed and asynchronous approach for optimal coverage of a domain with identical mobile sensing agents is proposed in [3] based on a framework for

Luciano Pimenta, Quentin Lindsey, and Vijay Kumar
GRASP Lab, University of Pennsylvania, USA
e-mail: {pimenta,quentinl}@seas.upenn.edu, kumar@grasp.upenn.edu

Mac Schwager and Daniela Rus
CSAIL, MIT, USA, e-mail: schwager@mit.edu, rus@csail.mit.edu

Luciano Pimenta, Guilherme Pereira, and Renato Mesquita
Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, Brazil
e-mail: {lucpim,renato,gpereira}@cpdee.ufmg.br

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 85–99.
springerlink.com

optimized quantization derived in [7]. Each agent (robot) follows a gradient descent control algorithm that minimizes a functional encoding the quality of the sensing coverage. Further, this control law depends only on the information about the position of the robot and that of its immediate neighbors, which are the robots in neighboring Voronoi cells. These control laws are computed without the requirement of global synchronization. The functional also uses a *distribution density function*, which weights points or areas in the environment according to importance. This adaptive technique can cope with changing environments, tasks, and network topology.

Several extensions of the framework devised in [3] have been proposed. In [1] the problem of limited-range interaction between agents was addressed. In [10] different distance metrics were used to address heterogeneous sensor footprints and nonconvex environments. In addition, constraints were added to the minimization problem to deal with finite sized robots. The problem of learning the distribution density function online while moving toward the optimal locations was addressed in [11].

The most relevant paper to this work is [2], in which the authors presented the problem of tracking an intruder in time-varying environments. In this case, a dynamic target was defined and its position used in the density function computation. The problem of covering environments with time-varying density functions was addressed by using a control law where convergence guarantees were difficult to establish.

In this paper, we translate the problem of optimally covering an environment while tracking intruders into the problem of covering environments with time-varying density functions. The proposed framework allows for the coupling of the three previously mentioned basic subtasks: target assignment, coverage, and tracking in an elegant distributed manner. For the first time, a decentralized controller with guaranteed exponential convergence is also derived. Simulations and actual robots experiments are also presented to verify the proposed approach.

2 Locational Optimization Framework

Locational optimization addresses how to place facilities or resources to minimize some cost [12, 4]. The canonical example is placing retail facilities to minimize the aggregate travel time of a population of customers. The locational optimization framework has been recently applied to the optimal placement of mobile sensors to provide sensor coverage of an environment [3]. In this section, we give a brief introduction to the locational optimization results relevant to our problem.

Let there be n mobile sensing agents in a bounded environment represented by $Q \subset \mathbb{R}^N$, and the location of the ith sensor is given by $\mathbf{p}_i \in Q$. Then $\mathbf{P} = [\mathbf{p}_1^T, \ldots, \mathbf{p}_n^T]^T \in Q \times \ldots \times Q$ is a vector representing the configuration of the mobile sensor network. Consider the cost function

$$\mathscr{H}(\mathbf{P}) = \int_Q \min_{i \in \{1,\ldots,n\}} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q})\, d\mathbf{q}, \tag{1}$$

where $d : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}_{\geq 0}$ is a function that measures distances between points $\mathbf{q} \in Q$ and sensor positions $\mathbf{p}_i \in Q$. The function $\phi : Q \to \mathbb{R}_{\geq 0}$ is a distribution density function, which defines a weight for each point in Q. The density function may reflect the probability of occurrence of events in different regions, or a measure of relative importance of different regions in Q. Therefore, points with greater weight values should be better covered than points with smaller values. The function $f : \mathbb{R} \to \mathbb{R}$ is a continuously differentiable, strictly increasing function over the range of d that measures the degradation of sensing performance with distance. The minimum over sensors reflects the fact that a point $\mathbf{q}$ should be the responsibility of the sensor that has the best sensing performance at $\mathbf{q}$. Covering the environment, Q, becomes an optimization to minimize the cost function (1). In the following subsection the necessary conditions to minimize the function in (1) are established, which will give the basis for a control strategy.

2.1 Computations with Voronoi Tessellations

Consider the minimization of (1)

$$\min_{\mathbf{P}} \mathscr{H}(\mathbf{P}) = \min_{\mathbf{P}} \int_Q \min_i f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q})\, d\mathbf{q}.$$

The minimum inside the integral induces a partition of Q into non-overlapping cells, V_i, to give

$$\min_{\mathbf{P}} \mathscr{H}(\mathbf{P}) = \min_{\mathbf{P}} \sum_{i=1}^{n} \int_{V_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q})\, d\mathbf{q}, \tag{2}$$

where $V_i = \{\mathbf{q} \in Q \mid f(d(\mathbf{q}, \mathbf{p}_i)) \leq f(d(\mathbf{q}, \mathbf{p}_j)), \forall j \neq i\}$. Since f is strictly increasing, this is equivalent to

$$V_i = \{\mathbf{q} \in Q \mid d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j), \forall j \neq i\}. \tag{3}$$

The region V_i is the Voronoi cell of $\mathbf{p}_i$. The collection of Voronoi cells is called the *Voronoi tessellation* of Q [9][1].

We now define a number of quantities relating to Voronoi cells. Let ∂V_i and ∂Q be the boundary of V_i and Q, respectively. By $\mathbf{q}_{\partial V_i}(\mathbf{P})$ we mean a point $\mathbf{q} \in \partial V_i$, and $\mathbf{n}_{\partial V_i}$ is the outward facing unit normal of ∂V_i. Given a robot i, we define $\mathscr{N}_i$ as the index set of robots that share Voronoi boundaries with V_i, $\mathscr{N}_i = \{j \mid V_i \cap V_j \neq \emptyset\}$. We denote the set of points on the Voronoi boundary shared by agents i and j as $l_{ij} = V_i \cap V_j$ as shown in Fig. 1. Then $\mathbf{q}_{l_{ij}}(\mathbf{p}_i, \mathbf{p}_j)$ is a point on that shared boundary, and $\mathbf{n}_{l_{ij}}$ is the unit normal of l_{ij} from $\mathbf{p}_i$ to $\mathbf{p}_j$. By the definition of the Voronoi cell (3), we know the following facts:

[1] It is convenient for us to use $\leq$ in the definition of Voronoi cell, rather than the more common definition with $<$.

$$\partial V_i = (\cup_{j\in\mathcal{N}_i} l_{ij}) \cup (\partial V_i \cap \partial Q), \tag{4}$$

$$l_{ij} = l_{ji}, \tag{5}$$

$$\mathbf{n}_{l_{ij}} = -\mathbf{n}_{l_{ji}}. \tag{6}$$

When d is the Euclidean distance, the shared boundaries l_{ij} are hyperplanes, and the Voronoi cells are convex.

The following lemma states an important fact about the cost function (2).

Lemma 2.1 (Distributed Gradient)

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \int_{V_i} \frac{\partial}{\partial \mathbf{p}_i} f(d(\mathbf{q},\mathbf{p}_i))\phi(\mathbf{q})\, d\mathbf{q}$$

Proof. Differentiating under the integral sign [5], we have

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = &\int_{V_i} \frac{\partial}{\partial \mathbf{p}_i} f(d(\mathbf{q},\mathbf{p}_i))\phi(\mathbf{q})\, d\mathbf{q} \\ &+ \int_{\partial V_i \cap \partial Q} f(d(\mathbf{q},\mathbf{p}_i))\phi(\mathbf{q}) \frac{\partial \mathbf{q}_{\partial V_i}(\mathbf{P})}{\partial \mathbf{p}_i} \mathbf{n}_{\partial V_i}\, d\mathbf{q} \\ &+ \sum_{j\in\mathcal{N}_i} \int_{l_{ij}} (f(d(\mathbf{q},\mathbf{p}_i) - f(d(\mathbf{q},\mathbf{p}_j)))\phi(\mathbf{q}) \frac{\partial \mathbf{q}_{l_{ij}}(\mathbf{p}_i,\mathbf{p}_j)}{\partial \mathbf{p}_i} \mathbf{n}_{l_{ij}}\, d\mathbf{q}.\end{aligned}$$

where $\frac{\partial \mathbf{q}_{\partial V_i}(\mathbf{P})}{\partial \mathbf{p}_i}$ and $\frac{\partial \mathbf{q}_{l_{ji}}(\mathbf{p}_i,\mathbf{p}_j)}{\partial \mathbf{p}_i}$ are $N \times N$ matrices. By definition of l_{ij}, $d(\mathbf{q},\mathbf{p}_i) = d(\mathbf{q},\mathbf{p}_j)\ \forall \mathbf{q} \in l_{ij}$, so the last sum vanishes. Since points on the boundary of the environment do not change position as a function of $\mathbf{p}_i$, we have

$$\frac{\partial \mathbf{q}_{\partial V_i}}{\partial \mathbf{p}_i} = 0 \quad \forall \mathbf{q} \in \partial V_i \cap \partial Q$$

and the second term vanishes. □

Lemma 2.1 means that the gradient

$$\frac{\partial \mathcal{H}}{\partial \mathbf{P}} = \left[\cdots \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}^T \cdots \right]^T$$

is distributed among the agents in the sense that an agent i can compute its own gradient component, $\partial \mathcal{H}/\partial \mathbf{p}_i$, using only information relevant to its Voronoi cell. This allows the design of *distributed* gradient algorithms, as was shown in [3].

2.2 Euclidean Setting

For the remainder of the paper, we will restrict ourselves to agents living on a plane ($N = 2$), where d is the Euclidean distance, $f(x) = x^2$, and the environment Q is

convex. For a discussion of controllers for the general setting, please see [10]. In this restricted setting, the cost function becomes

$$\mathscr{H} = \sum_{i=1}^{n} \int_{V_i} \|\mathbf{q} - \mathbf{p}_i\|^2 \phi(\mathbf{q})\, d\mathbf{q}. \tag{7}$$

Define the quantities

$$M_{V_i} = \textstyle\int_{V_i} \phi(\mathbf{q})\, d\mathbf{q},\ \mathbf{L}_{V_i} = \int_{V_i} \mathbf{q}\phi(\mathbf{q})\, d\mathbf{q},\ \text{ and }\ \mathbf{C}_{V_i} = \mathbf{L}_{V_i}/M_{V_i}. \tag{8}$$

Then Lemma 2.1 simplifies to

$$\frac{\partial \mathscr{H}}{\partial \mathbf{p}_i} = -\int_{V_i} 2(\mathbf{q} - \mathbf{p}_i)\phi(\mathbf{q}) d\mathbf{q} = -2M_{V_i}(\mathbf{C}_{V_i} - \mathbf{p}_i). \tag{9}$$

Clearly, critical points of $\mathscr{H}$ (configurations where the gradient is zero) are those in which every agent is at the centroid of its Voronoi cell, $\mathbf{p}_i = \mathbf{C}_{V_i}\ \forall i$. The resulting partition of the environment is commonly called a *Centroidal Voronoi Tessellation* (CVT). It is known that the critical points are local minima of $\mathscr{H}$[2]. Therefore, we desire that every agent drives to the generalized centroid of its Voronoi region to locally minimize the cost function (7). We do not attempt to find global optima as that is known to be difficult (NP-hard for a given discretization of Q) even in the fully centralized case. Next, we present a distributed control law that is used in [3] to converge to a CVT.

2.3 Continuous-Time Lloyd Algorithm

A classic discrete-time method to compute CVTs is Lloyd's algorithm [7]. In each iteration this method executes three steps: (i) compute the Voronoi regions; (ii) compute the centroids; (iii) move each point site to the corresponding centroid.

In [3] a continuous-time version of this approach is proposed for kinematic models

$$\dot{\mathbf{p}}_i = \mathbf{u}_i. \tag{10}$$

The control law

$$\mathbf{u}_i = \gamma(\mathbf{C}_{V_i} - \mathbf{p}_i) \tag{11}$$

guarantees that the system converges to a CVT, where γ is a positive gain. This control law is a gradient-descent approach ($\mathbf{u}_i \propto -\partial\mathscr{H}/\partial\mathbf{p}_i$ from (9)).

3 Tracking Moving Targets

Consider the *time-dependent* coverage function

[2] To prove this, one may consider the positive definiteness of the Hessian of $\mathscr{H}$ at such points.

$$\mathscr{H}(\mathbf{P},t) = \sum_{i=1}^{n} \int_{V_i} \|\mathbf{q} - \mathbf{p}_i\|^2 \phi(\mathbf{q},t) d\mathbf{q}. \tag{12}$$

More specifically, we are interested in tracking targets that invade a given region Q. In our approach, we do not limit the number of intruders. Furthermore, the location where they will appear and the instant of time of invasion are unknown. Therefore, we define the problem of tracking intruders while also keeping uniform coverage of Q. The uniform coverage is important to increase the probability of detecting new intruders. In order to provide a fully decentralized approach, the assignment of tasks should also be decentralized. There are $l(t)+1$ tasks, where $l(t)$ is the number of intruders at a given time t, ($l(t)$ tracking tasks and 1 uniform coverage task) to be accomplished. The number of tracking tasks, $l(t)$, can change over time as intruders enter and leave the environment. An intruder is only tracked while it is inside the environment Q.

We assume that each agent, i, is able to estimate the position and velocity of an intruder that is located inside the corresponding Voronoi cell V_i. In addition, we assume that the position and velocity of an intruder that is located at a distance R_t from the Voronoi cell V_i can be obtained by means of estimation or communication. This tuning parameter, R_t, represents the *maximum tracking distance*. This means that intruders located at distances greater than R_t from V_i will not be considered for the control law of agent i. In other words, the task of tracking this intruder will not be assigned to i.

We model the task assignment problem by composing radial basis functions, ϕ_i, which represent each intruder. Thus, we define the time-varying density function

$$\phi(\mathbf{q},t) = \sum_{i=1}^{l} \alpha_i \phi_i(\mathbf{q},t) + \beta, \tag{13}$$

where α_i and β are positive tuning constants that define the importance of a given task. The parameter α_i defines the importance of tracking intruder i, while the parameter β defines the importance of uniform coverage.

We consider radial basis functions that are centered at the intruder position, reach a maximum value at this position, and decreases to a negligible value at a distance R_t from the center. Therefore, the local computation of ϕ, performed by agent i, considers only the intruders within tracking distance. An option is the Gaussian function in [2]:

$$\phi_i(\mathbf{q},t) = A \exp\left(-\frac{(x-x_i(t))^2}{2\sigma^2} - \frac{(y-y_i(t))^2}{2\sigma^2}\right). \tag{14}$$

In this case, R_t is related to the value of σ. We let $R_t = \lambda\sigma$, where λ is a positive constant such that the values of ϕ at points out of this $\lambda\sigma$ range can be neglected.

One could also think of a probabilistic model of intruders and also use Gaussian density functions which could even have a different standard deviation for each axis. Such a modelling and procedures to find optimal tuning parameters α_i, β, A, and σ

are out of the scope of this paper. In the next subsection we discuss a decentralized controller that minimizes $\mathscr{H}(\mathbf{P},t)$.

3.1 Exponential Controller

The optimal deployment occurs when each agent is located at its Voronoi centroid, which is computed according to (8), (see section 2.1). Optimal deployment for time-varying density functions requires each agent to track its corresponding centroid over time. If the centroids were perfectly tracked, we would be able to guarantee that the function in (12) remains at a local minimum. Based on this, we define the performance function

$$E = \sum_{i=1}^{n} M_{V_i} \|\mathbf{C}_{V_i} - \mathbf{p}_i\|^2, \tag{15}$$

which yields a controller with exponential performance.

To maintain or improve tracking of the centroid according to the function in (15), we desire $\dot{E} \leq 0, \forall t$. Computing the time derivative leads to

$$\dot{E} = \sum_{i=1}^{n} \left(\frac{\partial E}{\partial \mathbf{p}_i}^T \dot{\mathbf{p}}_i + F_i \right) \tag{16}$$

where

$$F_i = (\mathbf{C}_{V_i} - \mathbf{p}_i)^T \int_{V_i} (2\mathbf{q} - \mathbf{C}_{V_i} - \mathbf{p}_i) \dot{\phi}(\mathbf{q},t)\, d\mathbf{q} \tag{17}$$

is the term resulting from the fact that $\phi(\mathbf{q},t)$ is time varying. After some algebraic effort, we can calculate the partial derivatives of E as

$$\frac{\partial E}{\partial \mathbf{p}_i} = -2(\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i), \tag{18}$$

where

$$\mathbf{R}_i = \sum_{j \in \mathscr{N}_i} \left[\frac{1}{2} \mathscr{M}_{ij} (\mathbf{C}_{V_i}^T \mathbf{C}_{V_i} - \mathbf{C}_{V_j}^T \mathbf{C}_{V_j}) - \mathscr{L}_{ij} (\mathbf{C}_{V_i} - \mathbf{C}_{V_j}) \right], \tag{19}$$

with

$$\mathscr{M}_{ij} = \int_{l_{ij}} \phi(\mathbf{q}) \frac{\partial \mathbf{q}_{l_{ij}}}{\partial \mathbf{p}_i} \mathbf{n}_{l_{ij}}\, d\mathbf{q}, \quad \text{and} \tag{20}$$

$$\mathscr{L}_{ij} = \int_{l_{ij}} \phi(\mathbf{q}) \mathbf{q} \left(\frac{\partial \mathbf{q}_{l_{ij}}}{\partial \mathbf{p}_i} \mathbf{n}_{l_{ij}} \right)^T d\mathbf{q}. \tag{21}$$

The vector $\mathbf{R}_i$ is due to the moving boundaries of the Voronoi regions and can be thought as a disturbance to the centroid position.

Collecting these terms together results in

$$\dot{E} = \sum_{i=1}^{n} \left(-2(\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i)^T \dot{\mathbf{p}}_i + F_i \right) . \tag{22}$$

We propose to use the controller

$$\mathbf{u}_i = \frac{(\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i)}{2\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|^2} (kM_{V_i}\|\mathbf{C}_{V_i} - \mathbf{p}_i\|^2 + F_i). \tag{23}$$

With the integrator dynamics $\dot{\mathbf{p}}_i = \mathbf{u}_i$, this controller gives the result $\dot{E} = -kE$, by substituting (23) back into (22). This means that our controller results in an *exponential* decay of the performance function

$$E(t) = E(0)e^{-kt} . \tag{24}$$

There is still a question as to the boundedness of the control signal $\mathbf{u}_i$ for the control law (23). We will discuss this point in subsection 3.3.

It is important to emphasize the decentralized aspect of the controller in (23). In order to compute all the terms in (23), each agent relies only on local information. Due to the properties of the intruder functions ϕ_i, previously discussed, each agent only needs to obtain the positions and velocities of intruders that are within a certain tracking distance from the corresponding Voronoi cell. Besides, each agent only needs the centroids of neighbor cells.

3.2 Computation of Boundary Terms

There remains a significant practical difficulty in computing the boundary terms $\mathbf{R}_i$. We will exploit the geometry of the Voronoi cell to derive a parametric formula for the computation of $(\partial \mathbf{q}_{l_{ij}}/\partial \mathbf{p}_i)\mathbf{n}_{l_{ij}}$. Then computing $\mathscr{M}_{ij}$ and $\mathscr{L}_{ij}$ will be a matter of integrating an explicit formula over a single parameter, and $\mathbf{R}_i$ can be readily computed. Define l_{ij} parametrically by

$$l_{ij} = \{\mathbf{q} \mid \mathbf{q} = v_{ij_1}\alpha + v_{ij_2}(1-\alpha), \quad \alpha \in [0,1]\}, \tag{25}$$

where $v_{ij_1}, v_{ij_2} \in \mathbb{R}^2$ are the end points of the line segment l_{ij}. Now we can write $\mathbf{q}(\alpha)$ for $\mathbf{q} \in l_{ij}$. By the definition of the Voronoi cell, any point $\mathbf{q}(\alpha)$ must satisfy

$$\mathbf{n}_{l_{ij}}^T \left(\mathbf{q}(\alpha) - \frac{\mathbf{p}_j + \mathbf{p}_i}{2} \right) = 0 \quad \forall \mathbf{q} \in l_{ij}, \tag{26}$$

where

$$\mathbf{n}_{l_{ij}} = \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|}. \tag{27}$$

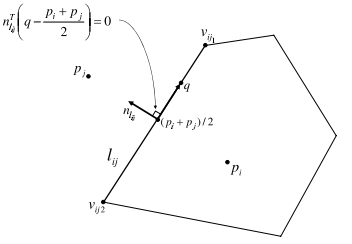


Fig. 1. This figure shows the quantities and constraints associated with the Voronoi boundary shared between neighboring agents.

Figure 1 shows a schematic of the quantities and constraints associated with the boundary shared between neighboring agents.

Differentiate (26) implicitly with respect to $\mathbf{p}_i$ to find the relation

$$\frac{\partial \mathbf{n}_{l_{ij}}}{\partial \mathbf{p}_i}\left(\mathbf{q}(\alpha)-\frac{\mathbf{p}_j+\mathbf{p}_i}{2}\right)+\frac{\partial \mathbf{q}_{l_{ij}}}{\partial \mathbf{p}_i}\mathbf{n}_{l_{ij}}-\frac{\mathbf{n}_{lij}}{2}=0 \tag{28}$$

where

$$\frac{\partial \mathbf{n}_{l_{ij}}}{\partial \mathbf{p}_i}=\frac{\mathbf{n}_{l_{ij}}\mathbf{n}_{l_{ij}}^T-I_2}{\|\mathbf{p}_j-\mathbf{p}_i\|}. \tag{29}$$

Simplify and substitute using $\mathbf{q}=v_{ij_1}\alpha+v_{ij_2}(1-\alpha)$ to find the desired formula

$$\frac{\partial \mathbf{q}_{l_{ij}}}{\partial \mathbf{p}_i}\mathbf{n}_{l_{ij}}=\frac{(I_2-\mathbf{n}_{l_{ij}}\mathbf{n}_{l_{ij}}^T)((v_{ij_1}-v_{ij_2})\alpha+v_{ij_2}-\mathbf{p}_i)}{\|\mathbf{p}_j-\mathbf{p}_i\|}+\frac{\mathbf{n}_{l_{ij}}}{2}, \tag{30}$$

where I_2 is the 2×2 identity matrix. Notice (30) is linear in α and can be computed readily given the known quantities $\mathbf{p}_i$, $\mathbf{p}_j$, v_{ij_1}, and v_{ij_2}. Then the integrals $\mathscr{M}_{ij}$ and $\mathscr{L}_{ij}$ potentially can be computed analytically over $\alpha\in[0,1]$ if $\phi(\mathbf{q}(\alpha))$ is in a form that allows for this (e.g. if it is polynomial), yielding closed form equations for the boundary terms. If $\phi(\mathbf{q}(\alpha))$ is not in a form for which the integrals can be performed analytically, they can be approximated numerically without much computational burden. This gives a method to compute the term $\mathbf{R}_i$ (19).

3.3 Asymptotic Analysis

In this subsection we address the asymptotic behavior of the control signal $\mathbf{u}_i$. One must take care that a singularity does not appear in the control signal as the centroidal configuration is reached. Firstly we formalize the requirement that the environment cannot change *too* quickly.

Assumption 1 (Bound). $\|\int_{V_i}(2\mathbf{q}-\mathbf{C}_{V_i}-\mathbf{p}_i)\dot{\phi}(\mathbf{q},t)\,d\mathbf{q}\|\leq B_i\ \forall i$ *and* $\forall t$.

This is a design requirement that means that $\phi(\mathbf{q},t)$ cannot change at an unbounded rate (e.g. the intruders have a bounded velocity).

Now we evaluate the limit to determine the asymptotic behavior of $\mathbf{u}_i$

$$\lim_{\mathbf{p}_j \to \mathbf{C}_{V_j}, \forall j} \|\mathbf{u}_i\| =$$

$$\lim_{\mathbf{p}_j \to \mathbf{C}_{V_j}, \forall j} \frac{\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|}{2\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|^2} |kM_{V_i}\|\mathbf{C}_{V_i} - \mathbf{p}_i\|^2 + F_i|$$

$$\leq \lim_{\mathbf{p}_j \to \mathbf{C}_{V_j}, \forall j} \frac{kM_{V_i}^{-1}\|\mathbf{L}_{V_i} - M_{V_i}\mathbf{p}_i\|^2 + |F_i|}{2\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|}$$

$$\leq \lim_{\mathbf{p}_j \to \mathbf{C}_{V_j}, \forall j} \frac{k\|\mathbf{L}_{V_i} - M_{V_i}\mathbf{p}_i\|^2 + \|\mathbf{L}_{V_i} - M_{V_i}\mathbf{p}_i\|B_i}{2M_{V_i}\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|}$$

$$= \lim_{\mathbf{p}_j \to \mathbf{C}_{V_j}, \forall j} \frac{\|\mathbf{L}_{V_i} - M_{V_i}\mathbf{p}_i\|B_i}{2M_{V_i}\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|}$$

where B_i is the bound from Assumption 1.

Conjecture 3.1 (Singularity)

$$(\mathbf{L}_{V_i} - M_{V_i}\mathbf{p}_i) \neq -\mathbf{R}_i \quad \forall i \text{ and } \forall t \text{ except when } \mathbf{C}_{V_i} = \mathbf{p}_i.$$

Conjecture 3.2 (Finite Limit)

$$\lim_{\mathbf{p}_j \to \mathbf{C}_{V_j}, \forall j} \frac{\|\mathbf{L}_{V_i} - M_{V_i}\mathbf{p}_i\|}{2M_{V_i}\|\mathbf{L}_{V_i} + \mathbf{R}_i - M_{V_i}\mathbf{p}_i\|} \text{ is bounded.}$$

Remark 3.1. Conjecture 3.1 states that the control signal will never become unbounded in finite time due to a cancellation of vectors in the denominator. It would be difficult to prove whether our system has this property *a priori* for a given set of initial conditions, but one can easily enforce a saturation bound on the control input to avoid this problem. Indeed, we do not see this problem occurring in simulation or hardware experiments.

Remark 3.2. Conjecture 3.2 states that the control input does not blow-up asymptotically as an agent converges to its centroid. One straightforward way to prove this conjecture would be to show that $\mathbf{R}_i$ does not converge to zero. Unfortunately, it can be proved that for bounded $\mathscr{M}_{ij}$ and $\mathscr{L}_{ij}$

$$\mathbf{p}_i = \mathbf{C}_{V_i} \quad \forall i \quad \text{implies} \quad \mathbf{R}_i = 0 \quad \forall i \tag{31}$$

Again, we were unable to prove this conjecture, but we rely upon simulations and experiments which show that the input does not blow-up. In practice one can impose saturation constraints to prevent this from happening.

4 Numerical Simulations

In this section we present ideal simulations that verify the proposed approach. The simple model in (10) is used. In these simulations and all the other simulations and experiments presented in this paper, we used parameters $\alpha_i = 50$, $\beta = 1$, and

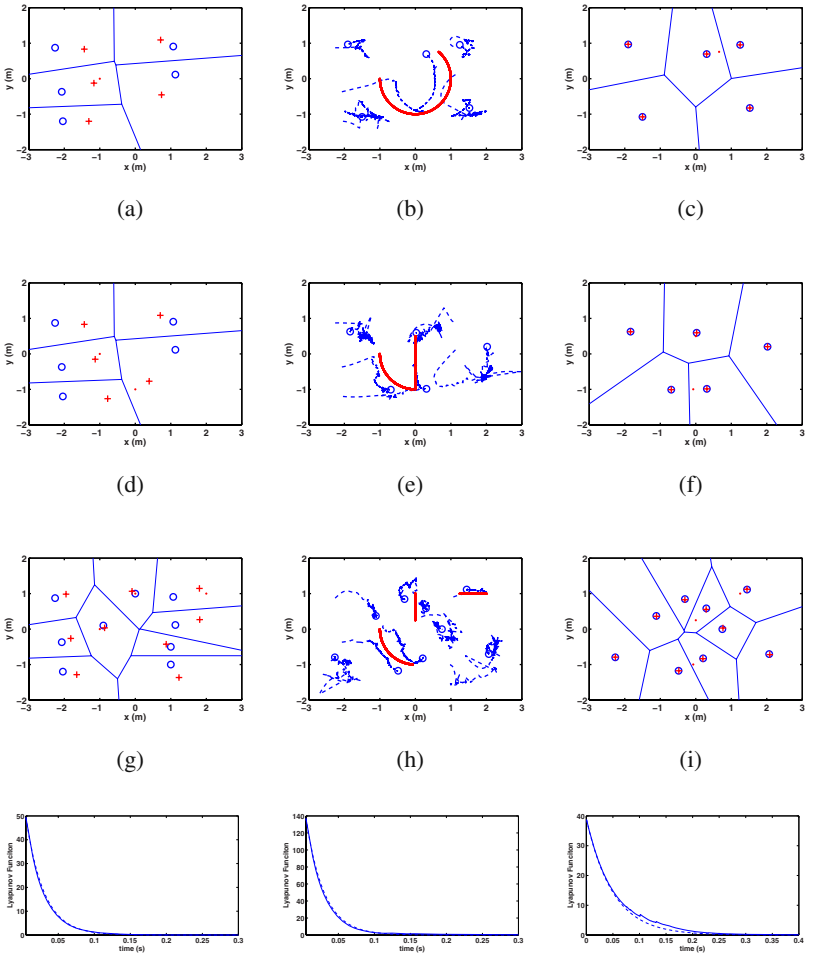


Fig. 2. Simulation results for three scenarios: (i) five robots and one intruder, $k = 40$, (Figs. 2a, 2b, 2c, and 2j), (ii) five robots and two intruders, $k = 40$, (Figs. 2d, 2e, 2f, and 2k), (iii) nine robots and three intruders, $k = 20$, (Figs. 2g, 2h, 2i, and 2l). The positions of the agents are given by the circles while the positions of the intruders are given by the dots. The crosses show the positions of the centroids. For each scenario it is shown initial configuration, trajectories, final configuration, and Lyapunov function evolution (solid line) along with the theoretical bound in (24)(dashed line).

Gaussian density function (eq. (14)) with parameters $\sigma = 0.5, A = 0.64$. We implemented the controller in (23) which is related to the Lyapunov function in (15). Figs. 2a, 2b, and 2c present the initial configuration, executed path, and final configuration, respectively, for a simulation with a team of five robots and one intruder. The intruder moves with constant speed of $0.1m/s$ in a uniform circular motion centered at the origin with unitary radius. Figs. 2d, 2e, and 2f present simulation results with five robots and two intruders. The second intruder moves with constant velocity vector $[0, 0.1]^T$. In both cases we used $k = 40$. A more complicated scenario is shown in Figs. 2g, 2h, and 2i where we have a team of nine robots and three intruders. Once again, one intruder moves according to the uniform circular motion. The other two intruders move with constant velocity vectors $[0, -0.05]^T$ and $[-0.05, 0]^T$. In this last simulation we used $k = 20$.

The Lyapunov function exponential decay is verified in Figures 2j, 2k, and 2l where we also show the bound given in (24). One can observe that the Lyapunov functions follow the expected bound as desired, even though we noticed some "jumps" in some simulations as in Fig. 2l. These "jumps" were caused by numerical errors related to the resolution in the numerical computation of integrals and also the simulation time step used. In fact, we verified that the system is very sensitive to these numbers.

5 Experiments

In this section, we verify the proposed approach in Gazebo [6], a simulated robotic environment, and on actual robots. These experiments were based on five Scarabs, differential drive robots, shown in Figure 3. For more information refer to [8]. We assumed an 6 m $\times$ 4 m environment.

The first scenario is the one presented in the last section with five robots and one intruder. The initial configuration of the robots are the same as in Figure 2a. First, we present the Gazebo simulation results. In Figures 4a and 4b, the final configuration of the robots are given along with the Lyapunov function E. One can see that this function is not bounded by the theoretical bound in (24) when $k = 40$. This inconsistency with the theoretical results comes from many factors including the saturation

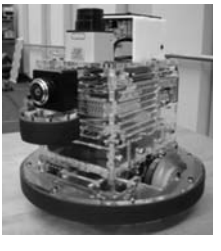


Fig. 3. Scarab, differential drive robotic platform, outfitted with a camera, Urg laser unit and tracking beacon.

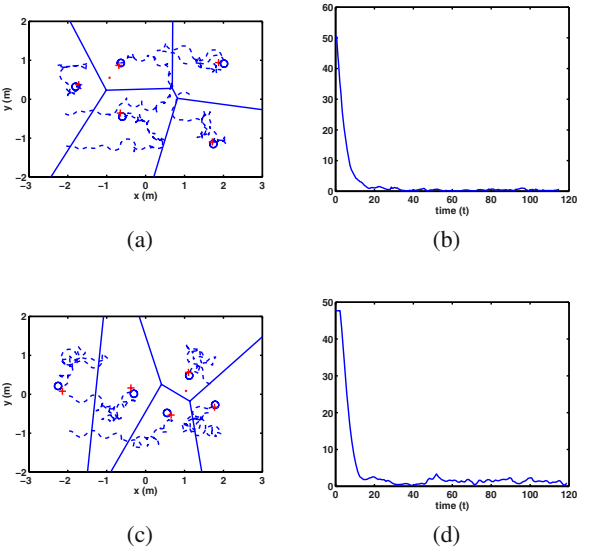


Fig. 4. Results for one moving intruder and five robots. Fig. 4a The final configuration and trajectories for a Gazebo simulation. Fig. 4b Evolution of the Lyapunov function for the Gazebo simulation. Figs. 4c and 4d Corresponding results for actual robots experiment.

of the acceleration caused by the limitations of a simulated robot, the nonholonomic constraints, and the repulsion term added to $\mathbf{u}_i$ to avoid collisions:

$$\mathbf{u}_{i,expt} = \mathbf{u}_i + \sum_{i \neq j} \frac{\kappa}{\|\mathbf{p}_i - \mathbf{p}_j\|^2} \mathbf{n}_{l_{ji}}, \tag{32}$$

where κ is a small positive collision avoidance gain.

The experiment was conducted using actual robots. Each robot localizes using an overhead tracking system and velocity information from its motor controllers. Their pose and velocities are sent to a single computer running Player [6], which calculates the new controls and sends these commands to the individual robots. Each robot is kinematically controlled using feedback linearization.

In Figures 4c and 4d, the results of the robot experiment are presented. We observed the same decay seen in Gazebo simulations as shown in Figure 4b along with the inconsistency with the theoretical bound.

We also conducted gazebo simulations for the scenario where we have nine robots and three intruders. The trajectories and configurations for different instants of time are presented in Figs. 5a, 5b, and 5c. Fig. 5d presents the Lyapunov function evolution. As expected, once again the exponential decay does not match the bound, with $k = 20$ in this case.

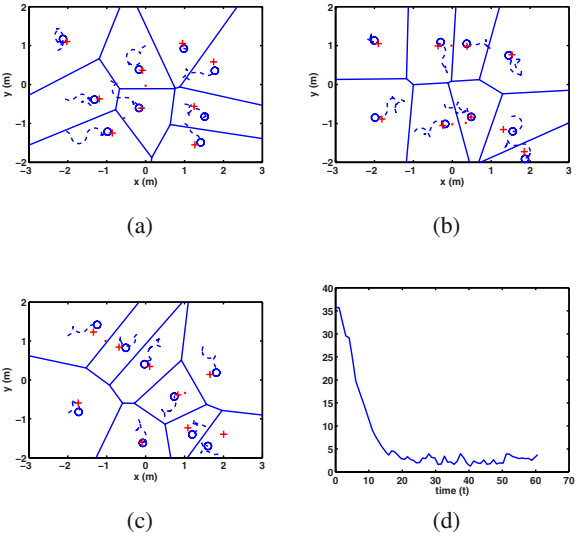


Fig. 5. Results for three moving intruders and nine robots. Figs. 5a, 5b, and 5c show trajectories for $t = 0.00$ to 20.20 sec, $t = 20.20$ to 40.44 sec, and $t = 40.44$ to 60.60 sec. Fig. 5d Evolution of the Lyapunov function for the Gazebo simulation.

6 Conclusion

This work proposes an extension of the framework proposed in [3] to solve the problem of tracking multiple intruders in a pre-specified region. The basic subtasks: task assignment, coverage, and tracking are coupled in an elegant manner. The proposed decentralized controller guarantees exponential convergence. A complete analysis of the singularities that are present in the controller is difficult. However, after conducting simulations and experiments we believe that such singularities do not pose a problem in practice.

Acknowledgements. We gratefully acknowledge support from NSF grant IIS-0427313, ONR grant N00014-08-1-0696, ARO grants W911NF-05-1-0219 and W911NF-08-2-0004, and CNPq – Brazil.

References

1. Cortés, J., Martínez, S., Bullo, F.: Spatially-distributed coverage optimization and control with limited-range interactions. ESIAM: Control, Optimisation and Calculus of Variations 11, 691–719 (2005)
2. Cortés, J., Martínez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks: Variations on a theme. In: Proceedings of the 10th Mediterranean Conf. on Control and Automation, Lisbon, Portugal, pp. 1–9 (2002)

3. Cortés, J., Martínez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation 20(2), 243–255 (2004)
4. Drezner, Z.: Facility Location: A Survey of Applications and Methods. Springer Series in Operations Research. Springer, New York (1995)
5. Flanders, H.: Differentiation under the integral sign. American Mathematical Monthly 80(6), 615–627 (1973)
6. Gerkey, B., Vaughan, R.T., Howard, A.: The Player/Stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics, pp. 317–323 (2003)
7. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inform. Theory 28(2), 129–137 (1982)
8. Michael, N., Fink, J., Kumar, V.: Experimental testbed for large multirobot teams. IEEE Robotics and Automation Magazine 15(1), 53–61 (2008)
9. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd edn. Wiley Series in Probability and Statistics. Wiley, New york (2000)
10. Pimenta, L.C.A., Kumar, V., Mesquita, R.C., Pereira, G.A.S.: Sensing and coverage for a network of heterogeneous robots. In: Proc. 47th IEEE Conf. on Decision and Control, Cancun, Mexico, pp. 3947–3952 (2008)
11. Schwager, M., Slotine, J.-J.E., Rus, D.: Consensus learning for distributed coverage control. In: Proc. IEEE Int. Conf. Robot. Automat., Pasadena, USA, pp. 1042–1048 (2008)
12. Weber, A.: Theory of the Location of Industries. The University of Chicago Press, Chicago (1929); Translated by Carl. J. Friedrich

Cooperative Towing with Multiple Robots

Peng Cheng, Jonathan Fink, Soonkyum Kim, and Vijay Kumar

Abstract. In this paper, we address the cooperative towing of payloads by multiple mobile robots that move in the plane. Robots pull via cables attached to an object or a pallet carrying a payload and they coordinate their motion to manipulate the payload through a planar, warehouse-like environment. We formulate a quasi-static model for manipulation and derive equations of motion that yield the motion of the payload for a prescribed motion of the robots in the presence of dry friction and tension constraints. We derive conditions of stable equilibrium for one, two, and three or more robots towing the payload and propose abstractions for the task that lend themselves to simple algorithms for motion planning. We present simulation and experimental results that demonstrate the basic concepts.

1 Introduction

There are many important applications in which vehicles are used to tow payloads using cables or chains. Conventional oceanographic data collection is largely dependent on towed systems. Tugboats are used to maneuver large boats and ships through rivers and canals. Tow boats are generally used for pushing while tug boats are used for towing a barge or multiple barges tied together. Helicopters are often used to carry suspended payloads.

In this paper we are primarily interested in cooperative towing of payloads by multiple vehicles. Cooperative lifting of large payloads by multiple helicopters has great benefits in humanitarian or military field operations. There are advantages to using multiple tugboats for towing. Smaller tugboats which have better maneuverability can be reconfigured to tow large barges depending on their payloads. For warehousing operations, automatic guided vehicles (AGVs) are generally used for carrying pallets. But pallets can also be towed by AGVs. And if pallets of different

Peng Cheng, Jonathan Fink, Soonkyum Kim, and Vijay Kumar
Grasp Laboratory, University of Pennsylvania
e-mail: chpeng@mathworks.com,
{jonfink,soonkyum,kumar}@grasp.upenn.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 101–116.
springerlink.com

sizes and weights are used, multiple robots can be reconfigured to tow pallets based on the size and payload.

From the standpoint of robotics, towing is an important manipulation process [12]. However, there are few studies of robotic towing, especially tasks involving cooperation between multiple robots. The kinematics and dynamics of cable-actuated, parallel manipulators, which have been studied extensively [14, 2, 16, 17], are relevant to this work. However, this body of literature primarily addresses the control of the cable extensions or forces in order to manipulate the payload. In contrast, towing involves cables of fixed length where manipulation is accomplished by controlling the motions of the "pivot points" in the parallel manipulators. While manipulation using cables has been studied in the context of distributed manipulation [8, 6, 7], these papers do not address the mechanics or control of the cooperative manipulation task.

Because we address the quasi-static manipulation of objects on a plane, the body of work on the mechanics of objects sliding on a rough plane is very relevant. Because assumptions of rigidity and planar contact are not compatible in practice, planar surface contact is difficult to model. It has been shown that any planar contact with dry friction can be modeled by three point contacts [12, 15]. Quasi-static analysis of a planar rigid body on a rough, horizontal plane reduces to first determining the force distribution in the vertical direction using a three-point-contact support, and then determining the frictional forces from the motion of the rigid body. However, there is no canonical three-point-contact support for a rigid body. Further, dry friction is very difficult to model [10].

Finally, the unilateral constraints imposed by cables that can only admit positive forces introduce another degree of complexity into the problem. Because the tension on a cable can only be positive and the distance between the two end-points of a cable cannot be more than its free length, and further the tension is non zero only when the distance is equal to the free lengths – each cable introduces a *complementarity constraint* of the type:

$$s \geq 0, \;\; \lambda \geq 0, \;\; \lambda\, s = 0 \tag{1}$$

where s is the slack in the cable and λ is the tension in the cable. The solutions to systems of linear equations subject to such constraints, the so-called *Linear Complementarity Problem* (LCP), have been studied extensively [5]. Even though all the terms in the LCP are linear in the unknowns, the system can have multiple solutions or no solutions at all. In addition to having complementarity constraints, the planar towing problem considered here has frictional constraints which are nonlinear.

In this paper, we first formulate a quasi-static model of towing planar objects with multiple robots in Section 2. In [4], we establish the quasi-static towing problem with n cables has a unique solution under certain conditions. In other words, if the robot motions are known, there is instantaneously a unique object motion. These results are briefly summarized in Section 3. However, because we can never know the exact distribution of forces at the contact surface, predicting the correct object motion is difficult. In Section 4, we show that there are stable equilibrium configurations for single-robot and multi-robot towing, independent of the support

distribution. In Section 5, we develop abstractions for towing that are independent of the uncertainties in support forces. In Section 6, we show how these abstractions can be used to develop an algorithm to follow a specified motion plan as well as simulation and experimental results.

2 The Quasi-static Model for Cooperative Towing

Our task is to control multiple robots so they can cooperatively tow or carry an object subject to gravity and frictional forces from any initial part configuration to a desired goal part configuration as depicted in Fig. 1. The position and orientation of the payload is given by (X, Y, θ) in the world frame. The velocity of the payload is a planar twist in the body-fixed frame, $x_b - y_b$, attached to the payload: $\xi = [\dot{x}, \dot{y}, \dot{\theta}]^T$. Let R_j denote the position of a reference point on robot j. There are m robots, each of which is attached via an inextensible cable that connects the point R_j on robot j and the point P_j on the payload. If $\left|\overrightarrow{P_j R_j}\right|$ equals the free length of the cable, then the dot product of the unit vector u_j with the relative velocity of the point P_j to the point R_j is non negative, which can be written as unilateral kinematic constraints[1]:

$$A\xi \geq b \quad \text{where} \quad A = \begin{bmatrix} A_1 \\ \dots \\ A_m \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ \dots \\ b_m \end{bmatrix}, \tag{2}$$

A_j is a function of the unit vector u_j, showing the direction of the cable j, and the position vector $\rho_j = \overrightarrow{O_b P_j}$:

$$A_j^T = \begin{bmatrix} u_j \cdot \mathbf{i} \\ u_j \cdot \mathbf{j} \\ (\rho_j \times u_j) \cdot \mathbf{k} \end{bmatrix}, \tag{3}$$

$\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ are respectively unit vectors along x_b, y_b, and z_b axis, and b_j is a function of u_j and the velocity $\mathbf{v}_{R,j}$ of the towing robot j:

$$b_j = u_j^T \mathbf{v}_{R,j}. \tag{4}$$

Note that $b_j \in [-v_R^{\max}, v_R^{\max}]$ because the robot velocity $\mathbf{v}_{R,j}$ is bounded by $v_R^{\max}$.

The set of *twists of freedom* [11] is defined with respect to the tuple (A, b) as follows:

$$\Sigma_{(A,b)} = \{\, \xi \,|\, A\xi \geq b \,\}. \tag{5}$$

Note that this set is determined by the towing configuration of the robots and the payload, specified by the matrix A, and the velocities of the robots, given by the vector b.

The kinematics-statics duality is evident in this problem. λ_c, the m-vector of cable tensions is non negative and is non zero only when the equality in (2) is satisfied.

[1] The vector inequality denotes that each element of the vector satisfies the inequality.

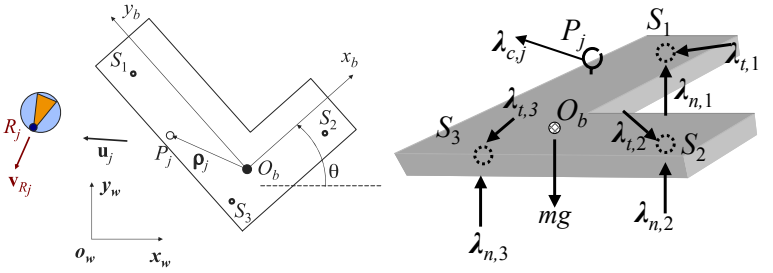


Fig. 1. Quasi-static manipulation: The object is supported by three support points, S_i, with normal forces (out of the plane), $\lambda_{n,i}$ and tangential frictional forces, $\lambda_{t,i}$. It is pulled by m cables, each exerting a force $\lambda_{c,j}$. Note the robot R_j pulls by moving the object with a prescribed (given) velocity.

Thus we write complementarity constraints:

$$0 \leq \lambda_c \perp A\xi - b \geq 0, \tag{6}$$

where "$\perp$" implies $\lambda_{c,j}(A\xi - b)_j = 0$.

In order to model the dry friction between the object and the support surface, we assume that the object is supported by a finite number of frictional point contacts. For three non collinear support points, the support forces $\lambda_{n,i} \geq 0, i = 1,2,3$ can be uniquely obtained from the following equation

$$\begin{bmatrix} 1 & 1 & 1 \\ y_{s,1} & y_{s,2} & y_{s,3} \\ x_{s,1} & x_{s,2} & x_{s,3} \end{bmatrix} \begin{bmatrix} \lambda_{n,1} \\ \lambda_{n,2} \\ \lambda_{n,3} \end{bmatrix} = \begin{bmatrix} mg \\ 0 \\ 0 \end{bmatrix}, \tag{7}$$

where $(x_{s,i}, y_{s,i})$ for $i = 1,2,3$ are the coordinates of the support points in the body-fixed frame.

If the object undergoes quasi-static motion, the tensions associated with m cables and the frictional forces $(\lambda_{t,i,x}, \lambda_{t,i,y})$ at each of the three support points $(i = 1,2,3)$ must add to zero. Thus, we have the equilibrium equations:

$$B^T \lambda_t + A^T \lambda_c = 0, \quad \lambda_c \geq 0 \tag{8}$$

where B is a full rank 6×3 matrix:

$$B^T = \begin{bmatrix} B_1^T & B_2^T & B_3^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ -y_{s,1} & x_{s,1} & -y_{s,2} & x_{s,2} & -y_{s,3} & x_{s,3} \end{bmatrix}, \quad B_i = \begin{bmatrix} 1 & 0 & -y_{s,i} \\ 0 & 1 & x_{s,i} \end{bmatrix}, \tag{9}$$

and λ_t is an unknown 6-vector with compoents in the body-fixed frame:

$$\lambda_t = [\lambda_{t,1}^T, \lambda_{t,2}^T, \lambda_{t,3}^T]^T, \quad \lambda_{t,i} = [\lambda_{t,i,x}, \lambda_{t,i,y}]^T. \tag{10}$$

We use FC_i to denote the friction cone at the ith support point defined by Coulomb friction:

$$0 \leq \|\lambda_{t,i}\| \leq \mu \lambda_{n,i}, \quad \|\lambda_{t,i}\| = \sqrt{\lambda_{t,i,x}^2 + \lambda_{t,i,y}^2}. \tag{11}$$

Note that FC_i is the friction cone with a known $\lambda_{n,i}$.

For a given object twist, the velocity vector of the support point can be written in the body-fixed frame:

$$v_{t,i} = B_i \, \xi = \begin{bmatrix} 1 & 0 & -y_{s,i} \\ 0 & 1 & x_{s,i} \end{bmatrix} \xi. \tag{12}$$

From Coulomb's law, the friction forces are equal to $\mu \lambda_{n,i}$ and are opposite to the direction of slip, except if the slip is zero when the magnitude is indeterminate. This can be written as

$$\lambda_t(\xi) \in \underset{\lambda_i \in FC_i}{\arg\min} \; \xi^T B^T \lambda. \tag{13}$$

It is not too hard to verify that this is equivalent to the Coulomb friction law [1].

3 Existence and Uniqueness of Solutions to (6), (8), and (13)

In this section, we will summarize the results in [4] about the existence and uniqueness of solutions to (6), (8), and (13), which will be used in this paper. We will omit the proofs. Please see [4] for more details.

The existence and uniqueness are established by considering the following max-min problem:

$$\underset{\xi \in \Sigma_{(A,b)}}{\text{maximize}} \quad \underset{\lambda_{t,i} \in FC_i}{\text{minimize}} \quad L(\xi, \lambda_t) \tag{14}$$

with the saddle function $L(\xi, \lambda_t) \equiv \xi^T B^T \lambda_t$ being bilinear in its arguments. The minimization part corresponds to the Coulomb friction law, which means that the support frictional forces will try to minimize the dissipation power with respect to a given object twist. The maximization part means that the part will move with a twist which maximizes the dissipation power due to support frictional forces. We say that a pair (ξ, λ_t) is *feasible* if ξ satisfies the kinematic constraint (2) and λ_t satisfies the friction constraints: $\lambda_{t,i} \in FC_i$ for all $i = 1, 2, 3$. By definition, a feasible pair $(\bar{\xi}, \bar{\lambda}_t)$ is a *saddle point* of L if for all feasible pairs (ξ, λ):

$$L(\bar{\xi}, \lambda_t) \geq L(\bar{\xi}, \bar{\lambda}_t) \geq L(\xi, \bar{\lambda}_t). \tag{15}$$

It is known [9, Theorem 1.4.1] that the following three statements are equivalent:

(a) $(\bar{\xi}, \bar{\lambda}_t)$ is a saddle point of L over the feasible pairs of admissible twists and friction forces;

(b) $\bar{\xi} \in \underset{\xi \in \Sigma_{(A,b)}}{\text{argmax}} \; \varphi(\xi)$ and $\bar{\lambda}_t \in \underset{\lambda_{t,i} \in FC_i}{\text{argmin}} \; \psi(\lambda_t)$, where

$$\varphi(\xi) \equiv \min_{\lambda_{t,i} \in FC_i} L(\xi, \lambda_t) \quad \text{and} \quad \psi(\lambda_t) \equiv \max_{\xi \in \Sigma_{(A,b)}} L(\xi, \lambda_t);$$

(c) $\varphi(\bar{\xi}) = \psi(\bar{\lambda}_t) = L(\bar{\xi}, \bar{\lambda}_t)$.

The existence and uniqueness of a solution to the quasi-static towing problem formulated in (6), (8), and (13) are established in two steps: (a) such a solution is characterized as a saddle point of L in (14); and (b) such a saddle point exists and is unique. The result below establishes the first step.

Theorem 3.1. [4] A feasible pair $(\bar{\xi}, \bar{\lambda}_t)$ is a saddle point of L if and only if $\bar{\lambda}_c$ exists such that the triple $(\bar{\xi}, \bar{\lambda}_t, \bar{\lambda}_c)$ satisfies conditions (6), (8), and (13).

To show that a unique saddle point exists, notice that

$$\begin{aligned} \varphi(\xi) &= \sum_{i=1}^{3} \underset{\lambda_{t,i} \in FC_i}{\text{minimize}} \left\{ (B_i \xi)^T \lambda_{t,i} \right\} \\ &= -\sum_{i=1}^{3} \mu \, \lambda_{n,i} \sqrt{\left(\dot{x} - y_{s,i}\dot{\theta}\right)^2 + \left(\dot{y} + x_{s,i}\dot{\theta}\right)^2}. \end{aligned}$$

$\varphi(\xi)$ is the sum of the dissipation power caused by the frictional forces at the three support points. The first equality in the above equation is from the definition of $\varphi(\xi)$. For the second equality, the dissipation power at support point i is $|\lambda_{t,i}| \|B_i \xi\|$ because of the Coulomb friction law, the tangential frictional force $|\lambda_{t,i}|$ equals $\mu \lambda_{n,i}$ for the fixed three support points, and the sliding velocity $\|B_i \xi\|$ equals $\sqrt{\left(\dot{x} - y_{s,i}\dot{\theta}\right)^2 + \left(\dot{y} + x_{s,i}\dot{\theta}\right)^2}$. We next consider the maximization of $\varphi(\xi)$ over $\Sigma_{(A,b)}$.

Theorem 3.2. [4] Suppose that $\Sigma_{(A,b)}$ is not empty (*i.e.*, there exists a twist that satisfies the kinematic constraint in (2)) and that $B \in \mathbb{R}^{6\times 3}$ is full rank. There exists a unique $\widehat{\xi}$ and possibly non unique $(\widehat{\lambda}_t, \widehat{\lambda}_c)$ such that $(\widehat{\xi}, \widehat{\lambda}_t, \widehat{\lambda}_c)$ satisfies the conditions (6), (8), and (13); moreover, $\widehat{\xi}$ is the unique solution of the problem:

$$\underset{\xi = (\dot{x}, \dot{y}, \dot{\theta})}{\text{minimize}} \; -\varphi(\xi) \quad \text{subject to } A\xi \geq b \tag{16}$$

and for $i = 1, 2, 3$, $\widehat{\lambda}_{t,i} \in \underset{\lambda_{t,i} \in FC_i}{\arg\min} \; v_{t,i}(\widehat{\xi})^T \lambda_{t,i}$.

Remark: The optimization problem (16) is a convex program. Indeed, it is equivalent to a non-standard linear program over a second-order cone:

$$\begin{aligned} &\underset{\xi = (\dot{x}, \dot{y}, \dot{\theta}), \sigma}{\text{minimize}} \; \sum_{i=1}^{3} \mu \, \lambda_{n,i} \, \sigma_i \\ &\text{subject to } A\xi \geq b \\ &\text{and} \qquad \sqrt{\left(\dot{x} - y_{s,i}\dot{\theta}\right)^2 + \left(\dot{y} + x_{s,i}\dot{\theta}\right)^2} \leq \sigma_i, \quad \text{for } i = 1, 2, 3. \end{aligned} \tag{17}$$

where σ_i is a positive upper bound for the sliding velocity at the support point i. This is a problem that has been extensively studied in the recent mathematical programming literature; see *e.g.* [3]. We use CVX[2] and Matlab© functions to solve for the object twist to generate the results shown in Section 6.

4 Steady-State Analysis

In this section, we will study the steady-state of the towing system when one or two robots move along straight lines as depicted in Fig. 2.

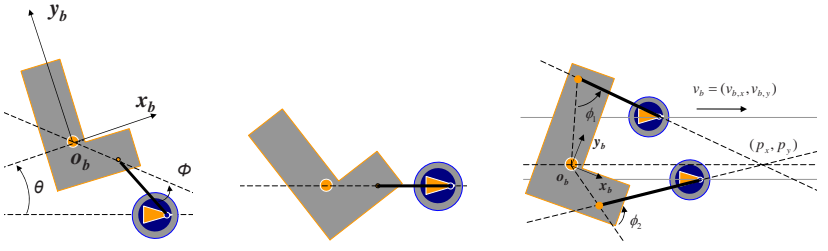


Fig. 2. (Left) Arbitrary initial configuration. (Center) Stable equilibrium configuration. (Right) The equilibrium of two-robot towing.

4.1 One-Robot Towing Case

The single robot towing system will converge to an equilibrium in which the system exhibits pure translation and the cable, the robot, and the center of mass of the part will be aligned as shown in Fig. 2. The result is stated in the following theorem.

Theorem 4.1. If a single robot tows the part with positive cable tension and moves in the invariant direction along a straight line, the angle ϕ will converge to zero.

Proof. Because the cable will have positive tension, the kinematic constraint will be an equality constraint and the part twist will be the result of the following optimization problem:

$$\xi^*(\phi) = \begin{bmatrix} \dot{x}(\phi) \\ \dot{y}(\phi) \\ \dot{\theta}(\phi) \end{bmatrix} = \begin{array}{l} \arg\min \quad -\varphi(\xi) \\ \xi \\ \text{subject to } A(\phi) = b \end{array}. \tag{18}$$

We will prove this by showing that $\dot{\theta}\phi < 0$ and $\dot{\theta} = 0$ if $\phi = 0$.

Step 1: We will first prove that $\xi^*(\phi)$ is well-defined and thus continuous.

Because there is only one robot, the matrix $A(\phi)$ is full rank and it is easy to check that the statement

$$\exists \lambda_c > 0, A^T \lambda_c = 0 \tag{19}$$

[2] http://www.stanford.edu/∼boyd/cvx/

is false because the unique solution to the equality is $\lambda_c = 0$ which does not satisfy the inequality. By Stiemke's lemma, (19) false implies that

$$\exists \xi : A\,\xi > 0 \tag{20}$$

For any given b, if elements $b_j \leq 0$, (20) implies the constraint $A_j\xi \geq b_j$ is satisfied. If there are elements $b_j > 0$, then we can always scale ξ with a positive scalar so that a feasible ξ that satisfies $A_j\xi \geq b_j$ can be found. Thus, $\Sigma_{(A,b)}$ is not empty. Theorems 3.1 and 3.2 guarantee that there always exists a unique part twist $\widehat{\xi}$ for (6), (8), and (13). Therefore, $\xi^*(\phi)$ is a well-defined function.

Because the convex objective function in (16) and constraint function $A(\phi)\xi = b$ are continuous, using the implicit function theorem, it can be easily shown that $\xi^*(\phi)$ is a continuous function of ϕ.

Step 2: When $\dot{\theta} = 0$, the part exhibits pure translation and does not rotate around any of the three support points. Therefore, the objective function is differentiable at $\dot{\theta} = 0$ and we can compute the following necessary KKT conditions.

$$\sum_{i=1}^{3} \alpha_i(\dot{x} - y_{s,i}\dot{\theta}) - \lambda\cos\phi = 0,\ \sum_{i=1}^{3} \alpha_i(\dot{y} + x_{s,i}\dot{\theta}) - \lambda\sin\phi = 0, \tag{21}$$

$$\sum_{i=1}^{3} \alpha_i(-y_{s,i}(\dot{x} - y_{s,i}\dot{\theta}) + x_{s,i}(\dot{y} + x_{s,i}\dot{\theta})) - \lambda\rho\sin\phi = 0, \tag{22}$$

in which

$$\alpha_i = \frac{\mu\lambda_{n,i}}{\sqrt{(\dot{x} - y_{s,i}\dot{\theta})^2 + (\dot{y} + x_{s,i}\dot{\theta})^2}}. \tag{23}$$

Solving these equations, we can see that $\dot{\theta} = 0$ if and only if $\phi = 0$, π, or $-\pi$

Step 3: It can also be checked that when $\phi > 0$, the resulting $\dot{\theta}$ is negative. Similarly, when $\phi < 0$, the resulting $\dot{\theta}$ is positive. □

Please see Fig. 4 for simulation and experimental results supporting the stable equilibrium of one-robot towing.

4.2 *Two-Robot Towing Case*

An equilibrium of two-robot towing is stated in the following theorem.

Theorem 4.2. When two robots tow the part by moving in the same direction and velocity (therefore maintaining fixed relative positions), the part can have zero angular velocity if the line passing through the center of mass and the intersection point of two cables is parallel to the robot direction of motion as shown in Fig. 2.

Proof. The position vector $\rho_i = [\rho_i\cos(\alpha_i), \rho_i\sin(\alpha_i)]^T$ in the body fixed frame, in which ρ_i is the length of the vector ρ_i. Let ϕ_j to be the angle between the anchor point position vector and the cable direction. Then the direction of cable will be $u_j = [\cos(\alpha_j + \phi_j), \sin(\alpha_j + \phi_j)]^T$ in the body fixed frame.

Assuming both robots and the part have a pure translation with the velocity of $v_b = [v_{b,x}, v_{b,y}]^T$ in the body fixed frame, the wrench balance equation is

$$\begin{bmatrix} \cos(\alpha_1+\phi_1) & \cos(\alpha_2+\phi_2) \\ \sin(\alpha_1+\phi_1) & \sin(\alpha_2+\phi_2) \\ \rho_1\sin(\phi_1) & \rho_2\sin(\phi_2) \end{bmatrix} \begin{bmatrix} \lambda_{c,1} \\ \lambda_{c,2} \end{bmatrix} = \begin{bmatrix} \mu mg\frac{v_{b,x}}{\|v_b\|} \\ \mu mg\frac{v_{b,y}}{\|v_b\|} \\ 0 \end{bmatrix}, \tag{24}$$

which is true only when

$$\frac{\rho_2\sin(\phi_2)\sin(\alpha_1+\phi_1) - \rho_1\sin(\phi_1)\sin(\alpha_2+\phi_2)}{\rho_2\sin(\phi_2)\cos(\alpha_1+\phi_1) - \rho_1\sin(\phi_1)\cos(\alpha_2+\phi_2)} = \frac{v_{b,y}}{v_{b,x}}. \tag{25}$$

It can be easily checked that (25) is true when the line of action of the first cable, the line of action of the second cable, and the line passing through the center of mass and parallel to the robot moving direction intersect at a single point. □

Exhaustive simulation and experimental results show that the two robot system converges to the equilibrium state shown in Fig. 2, given by (24). However, we have not been able to prove this analytically. Thus the result of a stable equilibrium under two-robot towing remains a conjecture.

5 Abstractions for Towing

Because of the uncertainties in the support force distribution, it is difficult to predict the object twist for a given robot motion. In this section, we will study geometric and kinematic abstractions for single and multi-robot towing that allow us to develop controls to guide the towed object along a planned path within specified tolerances.

5.1 Geometric Abstractions

The inextensible cable ensures that the distance between the anchor points on the robot and the part is no larger than the free length of the cable no matter what is the support distribution. We can use this property to derive the set of reachable configurations for the towed object.

One-robot towing: When the anchor point of the robot i is at (x_i, y_i), the inextensible cable of length l_i constraint will ensure that the configuration of the part will be in the following set (allowing the cable to be slack):

$$\mathscr{R}_i = \{(x,y,\theta) \,|\, \sqrt{(x-x_i)^2+(y-y_i)^2} \leq l_i + \rho_i, \theta \in [0, 2\pi)\}. \tag{26}$$

Two-robot towing: When the part is attached via cables to the robots i and j and the cables are allowed to be slack, then the part configuration will be in $\mathscr{R}_i \cap \mathscr{R}_j$.

If both cables have positive tension, then the system can be modeled as a four bar linkage assuming the positions of the two robots are fixed. The set of all possible

configurations of the part can be derived considering the admissible configuration space of a four bar linkage system.

5.2 Kinematic Abstractions

When there are no less than three robots, three cables are in tension, and their lines of action are independent, the twist of the towed object will be uniquely determined by the kinematic constraint $A\xi = b$, *i.e.*, the velocity and configuration of the robots. This enables us to achieve any object twist by maintaining the relative position of the robots with respect to the object.[3]

Three robot towing: It is worth noting that there exist object twists that cannot be achieved instantaneously by three robots as shown in the following theorem.

Theorem 5.1. [4] If A is full rank and there are $m \leq 3$ robots, not all twists can be produced.

See [4] for proof.

It is important to recognize that even when the kinematic constraints of a twist ξ are satisfied with a particular configuration A, the assumption that all three cables are in tension only holds if positive cable tensions can balance the possibly unknown support friction wrenches. We shall see in the following section that this can be difficult to guarantee for unknown support distributions.

Towing with more than three robots: We can also show in the following theorem that with one more robot any twist can be achieved by a fixed towing configuration of four towing robots.

Theorem 5.2. [4] Given four towing robots and any part twist ξ^{des}, we can find $\beta \in (0,1)$ and (A,b) such that $\widehat{\xi} = \beta\xi^{des}$ is the unique solution to (14).

See [4] for proof.

5.3 Incorporating Uncertainty for Abstractions

In our model of quasistatic towing presented above, we rely on a three-point model of the support friction. However, in real systems, it may be impossible to know the locations of these virtual support points or even the location of the center of mass for an arbitrary part. By using the following model of support friction uncertainty, we can characterize the set of possible wrenches W_s applied to the part by friction resulting from a particular twist ξ.

Our approximation of $W_s(\xi)$ is computed by first assuming that the center of mass lies within some ε-inset area of the part and then considering the effect of a mass distribution concentrated at a single point. For a given part twist ξ with

[3] In reality, since there will be errors in robot positioning, one or more cables can become slack. In this case, the object can be thought to be "caged" with a finite reachable set that contracts to point when the robot errors are zero.

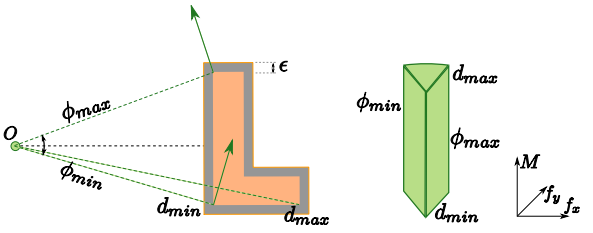


Fig. 3. On the left is the depiction of a specific center of rotation and the resulting components of the bounds for W_s. On the right is a visualization of W_s wth $\mu_{min} = 0$ and arbitrary μ_{max}.

center of rotation O, the minimum and maximum angles ϕ_{min}, ϕ_{max} that subtend the ε-inset geometry from the point O provide two bounds on the set $W_s(\xi)$ as depicted in Fig 3. Magnitudes of these frictional forces will be in the set $[\mu_{min}mg, \mu_{max}mg]$. The minimum and maximum frictional moments are determined based on the points closest and farthest from the center of rotation O with distances $d_{min}(\xi), d_{max}(\xi)$.

Now is it possible for a given towing configuration A to test if, for a given part twist ξ, the cables can generate wrenches to equilibritate any possible frictional wrenches and thus satisfy the quasi-static assumption while maintaining tension in each cable. In the next section, we will show how a convex over-approximation of $W_s(\xi)$ can be used to efficiently find a towing configuration for a desired twist ξ^{des}.

This characterization of the set of possible frictional wrenches allows for alternate explanations of the one, two, and three-robot towing results we have presented above. For one-robot towing, it is clear that the towing wrench is one-dimensional and thus can only balance the wrench set W_s when it reduces to a one-dimensional subset and they have the same alignment. This will occur only when the center of rotation moves to infinity (for pure translation) and the only-allowable center of mass aligns with the towing cable. With two-robot towing, the towing wrench spans a two-dimensional set which can balance W_s when it reduces to a two-dimensional set corresponding to center of rotation at infinity. Finally, three-robot towing yields a three-dimensional towing wrench space which can be chosen to span the frictional wrench space $W_s(\xi)$.

5.4 Summary

In conclusion, we have three levels of abstraction for quasi-static towing. Geometric models allow us to model the set of all reachable configurations. Kinematic models allow us to predict and control the instantaneous object twist. Additionally, quasi-static models predict stable configurations that are eventually reached when robot(s) move along (parallel) straight line trajectories. Finally, we provide a characterization of the uncertain wrenches due to friction which allows for the design of robust towing configurations that are invariant to the mass distribution of the part.

6 Motion Planning for Towing and Experimental Results

In this section, we will provide algorithms, simulation and experimental results on motion planning and control of systems that consist of several SCARAB mobile robots towing an L-shaped object. Details on the SCARAB platform as well as the accompanying architecture and tracking system (which provides localization information for the robots) can be found in [13]. Since the SCARAB robot uses a differential drive with nonholonomic constraints, we are able to attach the towing cable at a feedback linearization point which can be kinematically controlled.

6.1 Motion Planning

Motion planning for the towing system we have described can proceed in two ways. We can generate trajectories that, for a given towing configuration, consist solely of feasible twists.[4] Alternatively, we can use existing motion planning techniques to generate arbitrary trajectories for the part (expanded to include towing robots) in $SE(2)$ and then post-process the trajectories to find towing configurations that satisfy the required twist at each point. In this work, we shall pursue this second option.

In order to provide a tractable method for determining if a particular configuration A can positively span all wrenches in W_s, we rely on a convex over-approximation of W_s that consists of eight points. If A positively spans this over-approximation, then it can balance any wrench in W_s and equilibriate the system. We can easliy solve this test via a linear program as detailed in Algorithm 1.

Algorithm 1 CheckAForXi(A, ξ): Check that wrench matrix A can balance frictional wrenches due to ξ

Require: $\hat{W}_s(\xi)$, $|\hat{W}_s(\xi)| = 8$ {Convex over-approximation of $W_s(\xi)$}
  for $w \in \hat{W}_s$ **do**
    if $\sim$(SolveLP($find\ \lambda\ s.t.\ A^T\lambda = w\ and\ \lambda \geq 0$)) **then**
      return false {Fail if A cannot satisfy any point in W_s}
    end if
  end for
  return true

Now, suppose we have a desired part twist ξ^{des} for which the current towing configuration A^{cur} cannot span $W_s(\xi^{des})$. The problem of finding a towing configuration A which can span $W_s(\xi^{des})$ (let alone one which is close to A^{cur}) is difficult - both non-linear and non-convex. While we have experimented with some hueristics to

[4] Consider feasible twists to be those for which we can balance all possible support friction wrenches in W_s and for which we can satisfy the kinematic constraint $A^T\xi > 0$ so that each robot is doing positive work.

Algorithm 2 FindAForXi(ξ, A^{cur}): Search for towing configuration A *near* the current configuration A^{cur}

Require: $\theta_1^{min,max}$, $\theta_2^{min,max}$, $\theta_3^{min,max}$ {Limits on cable directions}
  $\theta_{1,2,3} = \hat{\theta}_{1,2,3} \leftarrow A_{cur}$ {Initialize θ_i to current configuration}
  $r = 0$
  while $\theta_{1,2,3}^{min} < \theta_{1,2,3} < \theta_{1,2,3}^{max}$ **do**
    for $\theta_{1,2,3}$ s.t. $|\theta_{1,2,3} - \hat{\theta}_{1,2,3}| = r$ **do**
      if CheckAForXi($A(\theta_{1,2,3})$, ξ) **then**
        return $A(\theta_{1,2,3})$
      end if
    end for
    $r+ = \Delta r$
  end while

find A given ξ, the search strategy detailed in Algorithm 2 has proven to be the most reliable.[5]

Given a desired part trajectory $q(t), \dot{q}(t)$, one can consider several design goals when attempting to post-process and generate a suitable configuration trajectory $A(t)$ which in turn dictates individual robot trajectories. In this work, we are simply focusing on feasible trajectories but will look into optimizations in future work. Our current strategy is inheriently greedy; we procees the trajectory at discrete values t_k and simply find $A(t_{k+1})$ that is closest to $A(t_k)$.

6.2 One-Robot Towing Experiments

While it is not possible to predict the exact motion for one-robot towing, we have proved convergence properties above for straight-line trajectories. Fig. 4 provides a plot of the ϕ angle for a series of experiments and matching simulations. Note that the rate of convergence depends on the uncertain support force distribution and that this accounts for some errors between the experimental and simulated data shown here.

6.3 Two-Robot Towing Experiments

As in the one-robot case, we have proved that there exists an equilibrium configuration for two-robot towing. Fig. 4 depicts experimental verification of this equlibrium where a number of towing configurations (with fixed distance between towing robots) are executed from different initial conditions.

[5] In implementation, we utilize a lookup table to improve the performance of this search algorithm.

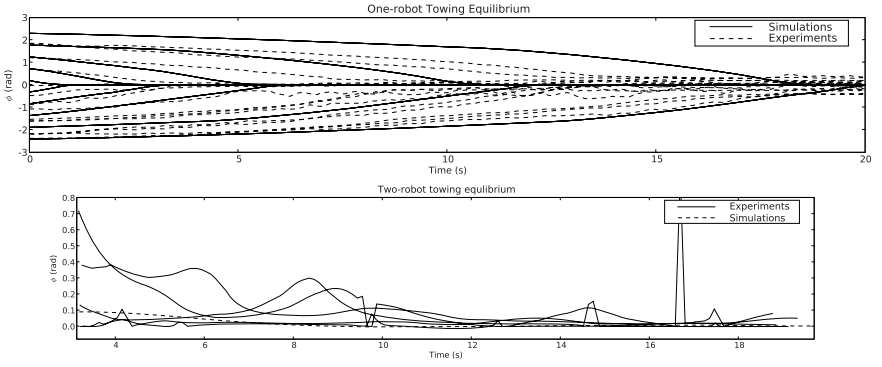


Fig. 4. (Top) Experimental and simulated trajectories of the ϕ angle showing that it approaches an equilibrium from several different initial conditions for a straight-line trajectory. (Bottom) Experimental verification of two-robot towing equilibrium.

6.4 Three-Robot Towing Experiments

With three robots, we are able to post-process the part trajectory to generate individual robot trajectories so that the part's motion is constrained to follow the desired trajectory. Fig. 5 demonstrates the use of three robots to tow a part along a complicated part trajectory to demonstrate this ability. Of course, a disadvantage here is that we must carefully coordinate between the three robots - a task that is often quite difficult in the presence of sensing uncertainty.

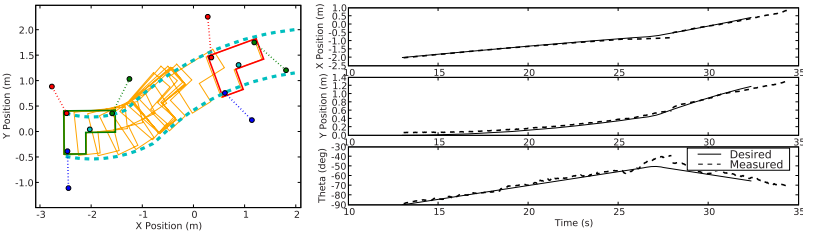


Fig. 5. (a) Plot of experimental part trajectory while being towed along a desired twist by three robots. Snapshots from the measured part configuration are overlayed on the desired envelope. (b) Position error relative to the desired trajectory

7 Conclusion and Discussion

In this paper, we studied the mechanics of planar, multi-robot towing in which multiple robots tow a planar payload subject to friction. The problem formulation incorporates complementarity constraints which are necessary to allow for cables becoming slack during a towing maneuver. We established the uniqueness and existence for the solutions to the forward dynamics problems using a max-min

problem formulation. We also showed that a payload towed by one robot driving along a straight line, or two robots driving along parallel straight lines, converges to an equilibrium configuration independent of the uncertainty in the support force distribution. Next we constructed a characterization of the uncertain support force distribution that allows us to find three-robot towing configurations that are invariant to even the center of mass location. Finally, we developed an algorithm by which allows a team of three-robots to cooperatively tow an object along an arbitrary trajectory within tolerances dictated only by the ability to kinematically control the robots.

Acknowledgement

We gratefully acknowledge the support of NSF grants DMS01-39747, DMS-075437, IIS02-22927, and IIS-0413138, and ARO grant W911NF-04-1-0148.

References

1. Anitescu, M., Potra, F.A.: A time-stepping method for stiff multibody dynamics with contact and friction. International Journal of Numerical Methods Engineering, 753–784 (July 2002)
2. Bosscher, P., Ebert-Uphoff, I.: Wrench based analysis of cable-driven robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, April 2004, pp. 4950–4955 (2004)
3. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
4. Cheng, P., Fink, J., Kumar, V., Pang, J.S.: Cooperative towing with multiple robots. ASME Journal of Mechanism and Robotics (2008)
5. Cottle, R.W., Pang, J., Stone, R.E.: The Linear Complementarity Problem. Academic Press, London (1992)
6. Donald, B.R.: On information invariants in robotics. Artificial Intelligence Journal 72, 217–304 (1995)
7. Donald, B.R.: Information invariants for distributed manipulation. International Journal of Robotics Research 16 (1997)
8. Donald, B.R., Gariepy, L., Rus, D.: Distributed manipulation of multiple objects using ropes. In: Proceedings IEEE International Conference on Robotics & Automation (2000)
9. Facchinei, F., Pang, J.S.: Finite-dimensional Variational Inequalities and Complementarity Problems. Springer, New York (2003)
10. Goyal, S., Ruina, A., Papadopolous, J.: Limit surface and moment function descriptions of planar sliding. In: Proc. IEEE Intl. Conf. on Robotics and Automation, May 1989, pp. 794–799 (1989)
11. Lipkin, H., Duffy, J.: The elliptic polarity of screws. Journal of Mechanisms, Transmissions, Automation and Design 107, 377–387 (1985)
12. Mason, M.T.: Mechanics of robotic manipulation. MIT Press, Cambridge (2001)
13. Michael, N., Fink, J., Kumar, V.: Experimental testbed for large multi-robot teams: Verification and validation. IEEE Robotics and Automation Magazine (March 2008)

14. Oh, S., Agrawal, S.: Cable-suspended planar parallel robots with redundant cables: Controllers with positive cable tensions. In: Proc. of IEEE International Conference on Robotics and Automation, pp. 3023–3028 (2003)
15. Peshkin, M.A.: Planning Robotic Manipulation Strategies for Sliding Objects. PhD thesis, Carnegie Mellon University Department of Physics (November 1986)
16. Stump, E., Kumar, V.: Workspaces of cable-actuated parallel manipulators. ASME Journal of Mechanical Design 128 (January 2006)
17. Verhoeven, R.: Analysis of the workspace of tendon-based Stewart platforms. PhD thesis, University of Duisburg-Essen (2004)

Part III
Manipulation

Two Finger Caging: Squeezing and Stretching

Alberto Rodriguez and Matthew T. Mason

Abstract. This paper studies the problem of placing two point fingers to *cage* a mobile rigid body in a Euclidean space of arbitrary dimension. (To *cage* an object is to arrange obstacles so that all motions of the mobile body are bounded). This paper shows that if a compact connected contractible object is caged by two points, then it is either *stretching caged* or *squeezing caged* or both, where *stretching caged* means the body is trapped even if the point fingers are given the freedom of moving apart, and *squeezing caged* means the the body is trapped even if the fingers are given the freedom of moving closer. This result generalizes a previous result by Vahedi and van der Stappen [18] which applied to two points trapping a polygon in the plane. Our use of a topological approach led to the generalization, and may lead to further generalizations and insights.

1 Introduction

A cage is an arrangement of obstacles that bounds the collision-free paths of some object. Caging is interesting for two reasons. First, caging an object is a way to manipulate it without immobilizing it. Second, even if an immobilizing grasp is needed/preferred over a cage, the cage may provide a useful waypoint to the immobilizing grasp. From some cages a *blind policy* exists that achieves an immobilization while preserving the cage.

Caging is weaker than immobilizing. While the objective of an immobilizing grasp might be to precisely locate an object relative to the hand, the objective of a cage is just to guarantee that the object is within the reach of the manipulator and cannot escape. By weakening the objective, the ultimate task may be easier both in theory and in practice.

Alberto Rodriguez · Matthew T. Mason
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue,
Pittsburgh, 15213 PA
e-mail: albertor@cmu.edu,matt.mason@cs.cmu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 119–133.
springerlink.com

Caging an object with point fingers assumes that the fingers are rigidly fixed relative to the hand and to each other. Vahedi and van der Stappen[18] recently introduced a variation where the fingers are allowed some relative motion while the object remains caged. An object is *squeezing caged* if all its motions are bounded, even if the fingers are allowed to move while not increasing the initial separation between them. Similarly, the object is *stretching caged* if it cannot escape, even if the fingers are allowed to move while not decreasing the initial separation between them. Vahedi and van der Stappen showed that any cage of a planar polygon by two disk fingers is either a squeezing cage, a stretching cage, or both.

This paper extends Vahedi and van der Stappen's result to include arbitrary compact connected contractible objects in Euclidean spaces of arbitrary dimension. Thus the squeezing and stretching caging characterization becomes a fundamental property of the configuration space of a two fingered manipulator.

2 Related Work

The earliest mathematical work on trapping objects was by Besicovitch in 1957 [1] and Shephard in 1965 [13]. Both worked on the problem first posed by Besicovitch as a *contest problem* to undergraduates, of trapping a sphere with a net. However, it was not until 1990 that Kuperberg [3] posed a formal definition of the 2D caging problem:

> "Let P be a polygon in the plane, and let C be a set of k points which lies in the complement of the interior of P. The points capture P if P cannot be moved arbitrarily far from its original position without at least one point of C penetrating the interior of P. Design an algorithm for finding a set of capturing points for P."

Since then, there have been several approaches to the problem, from different perspectives and with different goals. Rimon and Blake [10, 11] introduced the notion of a *caging set*: the maximal connected set of caging configurations that contains a given grasping configuration. They applied Morse Theory to the case of 1-parameter two-fingered grippers to show that the limit configurations where the cage is broken correspond to equilibrium grasps of the object. Later, Davidson and Blake [2] extended the result to 1-parameter 3-fingered planar grippers.

Sudsang and Ponce [15, 16] proposed and studied the application of caging to motion planning for three disc-shaped planar robots manipulating an object. They provided a geometrical method to compute conservative approximations of the so called *Inescapable Configuration Space regions*. They also analyzed in-hand manipulation using caging [17, 14].

Pereira, Campos and Kumar [9] applied caging to decentralized multirobot manipulation. They used the geometrical description of the robots to develop a conservative, on-line and decentralized test to maintain cageness.

Vahedi and van der Stappen [18] formalized squeezing and stretching caging to cage polygonal objects with two disc-shaped fingers, and used that idea to develop the first complete algorithm to compute the entire caging set of two fingers. Their

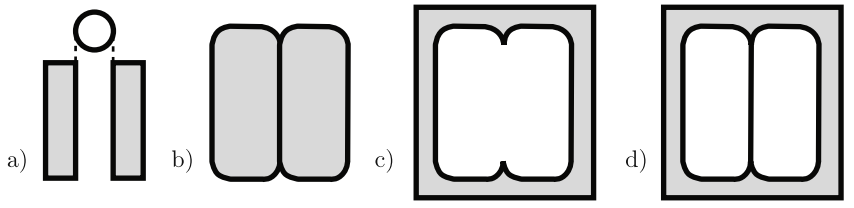


Fig. 1. a) Workspace: two obstacles and a moving object. b) Cspace obstacles. c) Free space. d) A more liberal definition of free space with unwanted thin bits.

algorithm generates a graph structure in the configuration space of the fingers that finds all caging grasps in $O\left(n^2 \log n\right)$ and handles cageness queries in $O\left(\log n\right)$.

3 Preliminary Concepts

3.1 Configuration Space

Assume the *workspace* to be $\mathbb{R}^d$, and let the *manipulator* be a set of position-controlled point *fingers* $p_1 \ldots p_n$ in $\mathbb{R}^d$. Let P_i be the configuration space of finger p_i, and let $M = P_1 \times \ldots \times P_n$ be the configuration space of the manipulator ($\dim M = n \cdot d$).

We assume the object O is a compact region of the workspace. It induces an obstacle for each of the fingers, and therefore for the manipulator. Let O_i be the obstacle induced for finger i in P_i, and let O^M be the obstacle induced for the manipulator in M. We can decompose the manipulator obstacle:

$$O^M = \{\,(p_1 \ldots p_n) \in M \mid \exists p_i \in O\,\} = \bigcup_{i=1}^{n} \{\,(p_1 \ldots p_n) \mid p_i \in O\,\} = \bigcup_{i=1}^{n} O_i^M \qquad (1)$$

where O_i^M is the obstacle induced in M by the interaction of object O with finger i.

Following the convention of [12, 18] we define the *free space of the manipulator* to be $M^{\text{free}} = M \setminus \left(O^M\right)$, where $\left(O^M\right)$ is the interior of O^M as a subset of $\mathbb{R}^{n \cdot d}$. We define the configurations in M^{free} to be the *admissible* configurations of the manipulator. This definition induces a free space in the configuration space of finger i, $P_i^{\text{free}} = P_i \setminus (O)$.

Note that there are alternative, more liberal, definitions of admissible configurations [5, 4]. The advantage of the stricter convention is that the free space is *regular*—it is the closure of its interior. Consequently it has no "thin bits" (Fig. 1). We will skip the proof to conserve space. By [5], regularity of the free space enables the following proposition:

Proposition 3.1 (Connectivity of the Free Space). *For any two configurations in the same connected component of the free space, an admissible connecting path exists, and lies in the interior of the free space except possibly at some isolated points.*

Proposition 3.1 implies that if two configurations of the manipulator are in the same connected component of the free space, they can be joined by a path that avoids contact with the object O except at isolated configurations. This property is essential to prove the main result in Sect. 5.3.

3.2 Paths in the Configuration Space

Given a configuration c of the manipulator in M^{free}, a *closed path based at c* refers to a parameterized curve $\alpha : [0,1] \longrightarrow M^{\text{free}}$ with $\alpha(0) = \alpha(1) = c$. A *contractible path* is a closed path that is path homotopic to a point in M^{free}.

Contractibility of a path, then, implies the existence of a continuous map H, called a *homotopy of paths*:

$$H : [0,1] \times [0,1] \longrightarrow M^{\text{free}} \tag{2}$$

such that $H(t,0) = \alpha(t)$ and $H(t,1) = H(0,s) = H(1,s) = c$, for all $t,s \in [0,1]$.

The next proposition, with proof in appendix 6, relates the contractibility of paths in M^{free} to the contractibility of the individual fingers' paths. First we define Π_i, the *natural projection* from the configuration space of the manipulator to the configuration space of finger i:

$$\begin{array}{rccc} \Pi_i : & M & \longrightarrow & P_i \\ & (p_1 \dots p_n) & \longrightarrow & p_i \end{array} \tag{3}$$

Proposition 3.2 (Characterization of contractible paths). *A closed path α at c is contractible in M^{free} if and only if for each finger i, $\Pi_i(\alpha)$ describes a contractible path in P_i^{free}.*

4 Caging

4.1 Introduction to Caging

To formalize the definition of caging proposed by Kuperberg [3] we could consider an object to be caged if and only if the object configuration lies in a compact connected component of its free space.

However, it is simpler and equivalent to consider the object to be fixed, and instead study the rigid motions of the manipulator, yielding the definition:

Definition 4.1 (Caging Configuration). Let M_c be the set of all configurations with the same pairwise finger distances as c. A caging configuration is a configuration c of the manipulator that lies in a compact connected component of $M^{\text{free}} \cap M_c$.

Hence, an object is caged if and only if the manipulator is unable to escape from the object while preserving its shape.

In the case of a two fingered manipulator, the Euclidean distance between the fingers is the only constraint that defines M_c. Let the map $r : M \longrightarrow \mathbb{R}$ be that distance. The set M_c is then defined as:

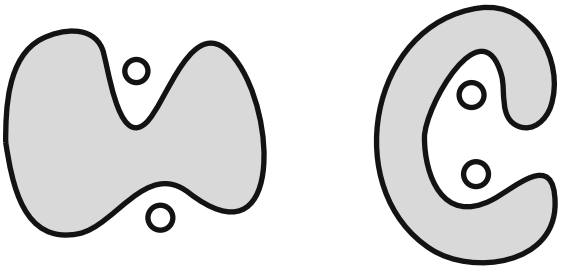


Fig. 2. Examples of squeezing (left) and stretching (right) caging configurations.

$$M_c = \{\, q \in M \mid r(q) = r(c) \,\} \tag{4}$$

4.2 Squeezing and Stretching Caging

Intuitively, the manipulator is in a *squeezing* (*stretching*) caging configuration if the object cannot escape, even by allowing the fingers to move closer (separate), Fig. 2. The definition can be formalized in a similar way as in the case of caging.

Let $\underline{M}_c$ and $\overline{M}_c$ be the sets:

$$\underline{M}_c = \{\, q \in M \mid r(q) \leq r(c) \,\} \tag{5}$$

$$\overline{M}_c = \{\, q \in M \mid r(q) \geq r(c) \,\} \tag{6}$$

Then we define:

Definition 4.2 (Squeezing Caging configuration). Configuration c of the manipulator that lies in a compact connected component of $M^{\text{free}} \cap \underline{M}_c$.

Definition 4.3 (Stretching Caging configuration). Configuration c of the manipulator that lies in a compact connected component of $M^{\text{free}} \cap \overline{M}_c$.

The main objective of this work is to show that all caging configurations are either squeezing caging, stretching caging or both.

5 The Squeezing and Stretching Caging Theorem

5.1 The Result

Theorem 5.1 (Squeezing-Stretching Caging). *Given a two finger caging configuration of a compact connected contractible object in $\mathbb{R}^d$, it is squeezing caging, stretching caging or both.*

We will prove the contrapositive. Suppose that a certain two finger configuration c is neither squeezing nor stretching caging. That means there is an escape path $\underline{\alpha}$ if squeezing is permitted, and there is also an escape path $\overline{\alpha}$ if stretching is permitted. We will use the existence of these two escape paths to construct a third escape path

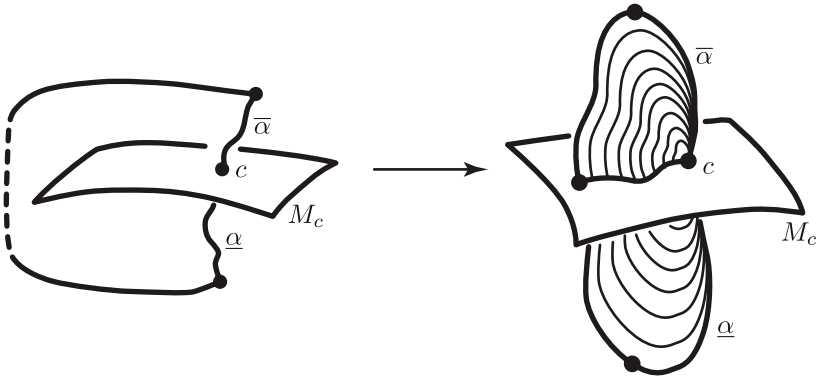


Fig. 3. The proof of theorem 5.1. (left) The contractible curve crossing M_c at infinity. (right) Simplest case intersection between the contraction and M_c.

in M_c, establishing the noncageness of c. From a topological perspective, we can understand the constructed escape path as an *average* of the squeezing and stretching escaping paths.

The proof consists of two steps (Fig. 3):

1. Using the two escape paths we build a closed contractible curve in M^{free}, with the property that every crossing through M_c, except for c, is known to be noncaging.
2. When the contractible curve is actually contracted, the intersection with M_c will give a rigid body path from c to one of the known noncaging configurations.

5.2 Building the Contractible Curve

The contractible curve must satisfy two requirements:

- All crossings of the curve through M_c, except for c, must be known to be noncaging.
- Both the curve and the contraction must live in M^{free}.

Let $B \subset \mathbb{R}^d$ be a ball containing O such that any placement of the fingers outside B is a noncaging configuration of the manipulator. We will say that any configuration with the fingers outside B is *at infinity*. Note that if O is compact, B always exists. For the case of two fingers we can choose B to be any circumscribing ball.

We can assume the escape paths $\underline{\alpha}$ and $\overline{\alpha}$ to terminate respectively at configurations $\underline{c}$ and $\overline{c}$ *at infinity*. Let $\underline{\alpha} \oplus \overline{\alpha}$ be the curve defined by the concatenation of both escaping paths. Closing $\underline{\alpha} \oplus \overline{\alpha}$ with an additional curve lying entirely outside B guarantees that all crossings of the complete curve through M_c will be noncaging. Hence we just need to see that always exists a curve β that closes $\underline{\alpha} \oplus \overline{\alpha}$ outside B in such a way that the complete closed curve is contractible.

By proposition 3.2, we can construct the completion of the curve independently for each finger. As long as each projected path $\Pi_i(\underline{\alpha} \oplus \overline{\alpha})$ is closed in a contractible

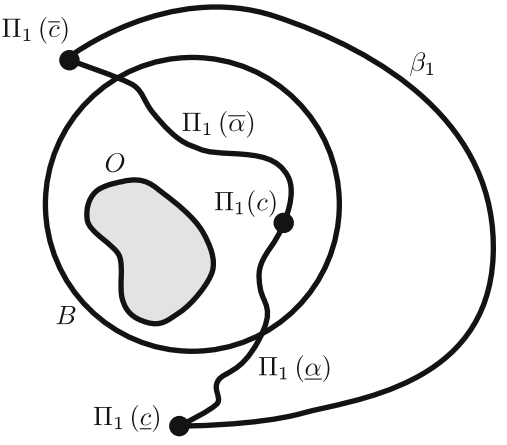


Fig. 4. Completion of the contractible path β_1 in P_1^{free}.

way in P_i^{free}, the curve in M and its contraction will be in M^{free}, satisfying the second requirement. Let β_i be the curve in P_i^{free} that closes $\Pi_i(\underline{\alpha} \oplus \overline{\alpha})$:

- If $d > 2$, as O is contractible (i.e. does not have holes) any path from $\Pi_i(\underline{c})$ to $\Pi_i(\overline{c})$ gives a closed contractible path.
- If $d = 2$, choose β_i to go around B as many times as needed to undo the *winding number* of $\Pi_i(\underline{\alpha} \oplus \overline{\alpha})$ (Fig. 4).

The curve $\beta = (\beta_1 \ldots \beta_n)$ closes $\underline{\alpha} \oplus \overline{\alpha}$ in a contractible way.

5.3 Intersection of the Contraction with M_c

The contractibility of the constructed path implies the existence of an homotopy H, as in equation (2), that contracts the closed path to c in M^{free}.

Let $H(S)$ be the image of the square $S = [0,1] \times [0,1]$ in M by the homotopy map. We are interested in the intersection of the homotopy image with the rigid motions set, $H(S) \cap M_c$. It may seem obvious that an escape path lives within that intersection, as illustrated by the simple case of Fig. 3, but for a more pathological homotopy the path may not exist. We will show that H can always be approximated by a well behaved contraction that yields a nondegenerate intersection.

The construction of the intersection relies on lemma 5.1, borrowed from differential topology [8].

Lemma 5.1. *Let M be an m-dimensional manifold and N an n-dimensional manifold, with $m \geq n$. If $f : M \longrightarrow N$ is smooth, and if $y \in N$ is a regular value, then the set $f^{-1}(y) \subset M$ is a smooth manifold of dimension $m - n$.*

If M is a manifold with boundary and y is also regular for the restriction $f \mid \partial M$, then $f^{-1}(y)$ is a smooth $(m-n)$ manifold with boundary. Furthermore, the boundary $\partial\left(f^{-1}(y)\right)$ is precisely equal to the intersection of $f^{-1}(y)$ with ∂M.

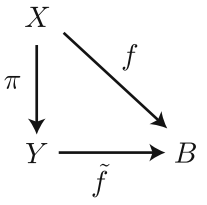


Fig. 5. Quotient map.

The rest of this section uses lemma 5.1 to construct the intersection $H(S) \cap M_c$, requiring some special care to satisfy the smoothness requirements.

To simplify later arguments we will change the domain of the homotopy from a square to a disc. We can view the homotopy H as a parametrization of the set $H(S)$, where ∂S is mapped to the contractible curve $\underline{\alpha} \oplus \beta \oplus \overline{\alpha}$ with the inconvenience that three sides of S are mapped to c. Let π be the quotient map that identifies those three sides of the square into one single point q. The map π transforms S into a disc D, whose boundary is a one to one mapping of the contractible curve.

The characteristic properties of the quotient topology [6], expressed in theorem 5.2, guarantee the existence and uniqueness of a continuous map $\tilde{H} : D \longrightarrow M^{\text{free}}$ that commutes the diagram on Fig. 6. From now on all mentions to the contraction will refer to that quotient induced map $\tilde{H}$.

Theorem 5.2 (Passing to the Quotient). *Suppose $\pi : X \longrightarrow Y$ is a quotient map, B is a topological space, and $f : X \longrightarrow B$ is any continuous map that is constant on the fibers of π (i.e., if $\pi(p) = \pi(q)$, then $f(p) = f(q)$). Then there exists a unique continuous map $\tilde{f} : Y \longrightarrow B$ such that $f = \tilde{f} \circ \pi$, as in Fig. 5.*

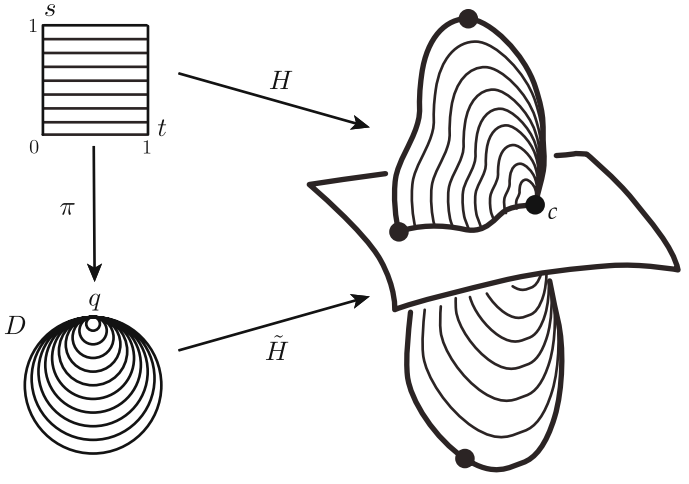


Fig. 6. Construction of the homotopy $\tilde{H}$ induced by the quotient map. See that $\tilde{H}(q) = c$.

Let F be the composition of the contraction $\tilde{H}$ with the distance map r:

$$F : D \xrightarrow{\tilde{H}} M^{\text{free}} \xrightarrow{r} \mathbb{R} \tag{7}$$

If r_c is the distance between the finger points at c, then $F^{-1}(r_c)$ parameterizes the intersection set $\tilde{H}(D) \cap M_c$. $\tilde{H}$ maps the set $F^{-1}(r_c) \cap \partial D$ to the crossings of the contractible curve through M_c. By construction, except for q that maps to c, all points in $F^{-1}(r_c) \cap \partial D$ are mapped by $\tilde{H}$ to noncaging configurations.

Showing that c is also a noncaging configuration – i.e. there is a rigid escape path for c – is equivalent to finding a path from q to any other point in ∂D within $F^{-1}(r_c)$. Lemma 5.1 allow us to construct the set $F^{-1}(r_c)$ and prove the existence of that path.

If F is smooth and r_c is a regular value, lemma 5.1 says that $F^{-1}(r_c)$ is a one-dimensional smooth manifold with the set $F^{-1}(r_c) \cap \partial D$ as boundary – a finite union of copies of $\mathscr{S}^1$ entirely in the interior of D and smooth paths that begin and end in ∂D. Since $q \in F^{-1}(r_c)$ and $q \in \partial D$, it follows that q must be connected to another point in ∂D within $F^{-1}(r_c)$. Thus, the connecting curve in D maps to a rigid escape path from q and the theorem is true. All that remains is to see what happens when F is not smooth or r_c is not a regular value.

5.3.1 Smoothness of $\tilde{H}$

To apply lemma 5.1 to F, $F = r \circ \tilde{H}$ must be smooth, hence the contraction $\tilde{H}$ needs to be smooth. For the contraction to be smooth, the contractible curve $\underline{\alpha} \oplus \beta \oplus \overline{\alpha}$ must also be smooth. We address this issue using theorem 5.3, Whitney's Approximation Theorem [7], to construct smooth ε-approximations of both.

Theorem 5.3 (Whitney Approximation Theorem). *Let M be a smooth manifold and let $F : M \longrightarrow \mathbb{R}^k$ be a continuous function. Given any positive continuous function $\varepsilon : M \longrightarrow \mathbb{R}$, there exists a smooth function $\hat{F} : M \longrightarrow \mathbb{R}^k$ that is ε-close to F ($\|F(x) - \hat{F}(x)\| < \varepsilon(x)\ \forall x \in M$). If F is smooth on a closed subset $A \subset M$, then $\hat{F}$ can be chosen to be equal to F in A.*

The challenge is to preserve two key properties of $\tilde{H}$ when we construct the approximation:

1. All crossings of M_c through the boundary of $\tilde{H}(D)$, other than c, must remain noncaging.
2. The approximation of $\tilde{H}$ must still live in M^{free}.

The first property was obtained by making the contractible curve go *to infinity* before crossing M_c. Spurious crossings of M_c must be prevented. This is especially awkward near the crossing at c. No matter how small an ε we choose, the ε-approximation of $\tilde{H}$ could cross again. Similarly, where $\tilde{H}$ makes contact with the obstacle, the ε-approximation might violate the second property by leaving the free space M^{free}. The following procedure produces a smooth approximation while avoiding the problems:

1. Replace the contractible curve locally at c by a smooth *patch* in M^{free}, as shown in Fig. 7. This is possible because of our "stricter" definition of free space, ensuring the free space has no thin bits. Even if c is in contact with O^M, there is still freedom to smoothly escape the contact through half of the directions on the tangent space.
2. Apply a similar patch wherever the contractible curve contacts the object. Thanks to the regularity of the free space, proposition 3.1 guarantees that these contact points are isolated configurations.
3. Apply theorem 5.3 to approximate the contractible curve by a smooth curve, equal to the original curve on the patches.
4. Define the contraction $\tilde{H}$ as before, but using the smoothed contractible curve.
5. If the contraction makes contact with the object, the approximation could violate the second key property. Repeat the previous strategy of defining smooth patches.
6. Apply theorem 5.3 once more to approximate the contraction by a smooth one $\hat{H}$ that equals the original contraction on the closed set $\partial D \subset D$ and on any patches, and otherwise lives in the free space $M^{\textbf{free}}$.

5.3.2 Regularity of r_c

If r_c is a regular value of the now smooth mapping $\hat{F} = r \circ \hat{H}$, lemma 5.1 says that $F^{-1}(r_c)$ is a one-dimensional smooth manifold. We have seen that consequently there is a smooth escape path within $F^{-1}(r_c)$ that connects q with another point in ∂D, mapped by $\hat{H}$ to a known noncaging configuration. But if r_c is not regular then $F^{-1}(r_c)$ might not be a manifold and the argument fails. This section shows that in such cases the escape path connecting q with a boundary point of $F^{-1}(r_c)$ still exists.

Sard's Theorem characterizes the critical points of a smooth map:

Theorem 5.4 (Sard's Theorem). *Let $f : U \longrightarrow \mathbb{R}^p$ be a smooth map, with U open in R^n and let C be the set of critical points; that is the set of all $x \in U$ with rank $d f_x < p$. Then $f(C) \subset \mathbb{R}^p$ has measure zero.*

The theorem says that for a smooth real valued function, the set of regular values is dense on the image of the function. Consequently, given any critical value z of the

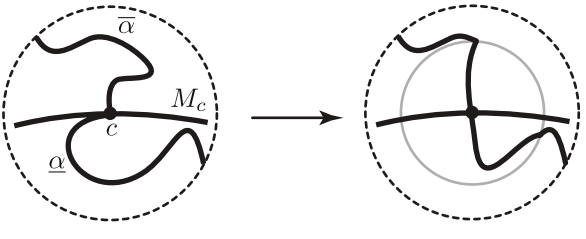


Fig. 7. Smooth patch for replacing the contractible curve in a neighborhood of c and eliminate possible nonsmoothness at c.

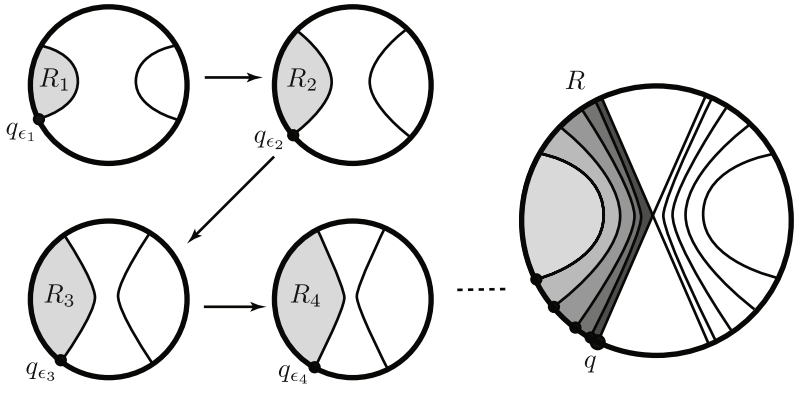


Fig. 8. For every ε_n, there is a smooth path in $F^{-1}(r_c - \varepsilon_n)$ from q_{ε_n}. The sequence of those paths defines in the limit the escape path from q.

smooth function f, there is a monotonic sequence of regular values converging to z, $\{z_n\}_n \longrightarrow z$.

In the specific case of the smooth map $\hat{F} : D \longrightarrow \mathbb{R}$, if $r_c \in \mathbb{R}$ is a critical value, let $\{r_n\}$ be a monotonic sequence of regular values converging to r_c. Let ε_n be the positive difference $r_c - r_n$ so that:

$$\{\varepsilon_n\}_n \longrightarrow 0 \text{ and } \{r_c - \varepsilon_n\}_n \longrightarrow r_c \tag{8}$$

We know that the contractible path goes from $\underline{M}_c$ to $\overline{M}_c$ locally at c. As $\hat{H}$ maps ∂D to the contractible curve, the restriction $\hat{F} \mid_{\partial D}$ is monotonic in a neighborhood

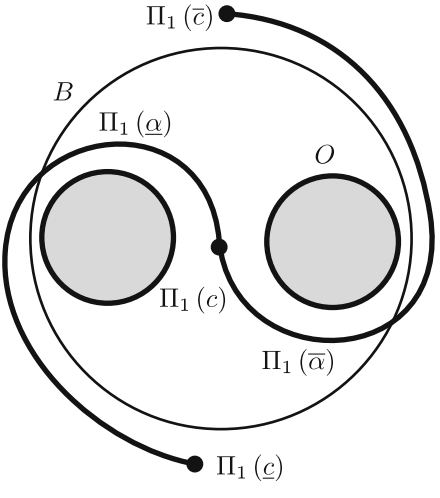


Fig. 9. Impossible to undo the winding number of $\underline{\alpha} \oplus \overline{\alpha}$ outside B, due to the nonconnectedness of O.

of q. Therefore we can expect to find a monotonic sequence of points $\{q_{\varepsilon_n}\}_n$ in a neighborhood of q at ∂D that converges to q, such that $q_{\varepsilon_n} \in F^{-1}(r_c - \varepsilon_n)\ \forall n$.

The sets $F^{-1}(r_c - \varepsilon_n)$ are smooth one-dimensional manifolds, because $r_c - \varepsilon_n = r_n$ are regular values. Consequently there is a smooth path from q_{ε_n} to another point in ∂D within $F^{-1}(r_c - \varepsilon_n)$. The sequence of regular values $\{r_c - \varepsilon_n\}_n$ induces a sequence of smooth paths that gradually approaches the set $F^{-1}(r_c)$ with $\varepsilon_n \longrightarrow 0$, as illustrated in Fig. 8.

Each path of the succession defines a subset $R_n \subset D$ bounded partially by the smooth path and partially by the boundary of D itself. Each region in the sequence is a subset of the next, because two smooth paths of different regular values cannot cross, and the sequence of regular values increases monotonically.

The union of all those regions $R = \bigcup_n R_n$ defines a set in D that, by construction, is partially bounded by ∂D and the set $F^{-1}(r_c)$. The section of that boundary within the set $F^{-1}(r_c)$ provides the desired path. Since $\{q_{\varepsilon_n}\}$ converges on q, the *limit path* connects q with another point in ∂D. Thus in Fig. 8 while $F^{-1}(r_c)$ might not be a manifold, and the path obtained might not be smooth, it still exists.

5.4 Requirements Revisited

Theorem 5.1 imposes three requirements on the object O as a subset of $\mathbb{R}^d$: compactness, connectedness and contractibility. The theorem implies their sufficiency, but not their necessity.

- **Compactness:** One of the requirements when building the contractible curve is that all crossings through M_c should be known noncaging configurations, except for c. We obtain this by constructing a ball around the object, which easily addresses the issue for compact objects. However, noncompact objects do not necessarily falsify the theorem.
- **Connectedness**: We used connectedness to build the contractible curve in a systematic way. For nonconnected objects (Fig. 9) it may be impossible to undo the winding number of $\underline{\alpha} \oplus \overline{\alpha}$ outside the ball B, but there may be some other way to construct a suitable contractible curve.
- **Contractibility**: If the object O has holes and $d > 2$, we may again be unable to close the path in a contractible way outside the ball B.
  Contractibility is not required in the planar case, because:
  - If one or two fingers are inside a hole, the object is clearly caged, squeezing caged and stretching caged.
  - If none of the fingers are inside a hole, the hole is irrelevant and Theorem 5.1 applies directly.

5.5 Implications

The squeezing and stretching theorem (theorem 5.1) gives an interesting characterization of caging configurations: if an object is trapped, it will remain trapped even

if we allow the point fingers a partial motion freedom: squeezing in some cases, stretching in others.

This squeezing and stretching result connects caging to immobilization. Suppose that c is a caging configuration, and for example it is the squeezing caging case. By definition, c lies in a compact connected component of $M^{\text{free}} \cap \underline{M}_c$. If the fingers squeeze, even in a *blind* way, the motion is bounded, and the path must inevitably end in an immobilization, at least for frictionless generic shapes. Caging helps to solve the immobilization problem by thickening the zero measure set of immobilizing configurations to regions of positive measure.

The immobilization problem is the problem of finding a manipulator configuration that eliminates the object's freedom of movement, given a geometric description of its shape and location. The formulation of the problem has two main inconveniences:

- The set of solutions to the immobilization problem is a subset of the contact space. Therefore, the set of solutions to the problem, as a subset of the configuration space of the manipulator, has measure zero.
- The theoretical formulation of the problem relies on unrealistic assumptions such as having a *perfect* geometrical model of the object both in terms of shape and location, and the ability to place the manipulator *perfectly* in a specific configuration.

These two properties would seem to make immobilization impractical, since any error would make it impossible to place the manipulator on a set of measure zero.

Nonetheless, manipulators do achieve immobilizing configurations. Feedback of contact sensor data is part of the answer. More importantly, the inherent compliance of the effector mechanism, the servos, and the object can accommodate errors. Even if the robot does not reach the desired configuration, it may reach a nearby immobilizing configuration.

The squeezing and stretching theorem facilitates the process of achieving an immobilization by providing an initial condition and a blind strategy for achieving an immobilization.

6 Conclusions and Future Work

Our main result is that any caging of a compact connected contractible object in $\mathbb{R}^d$ by two points is either a squeezing caging, a stretching caging, or both. This generalizes the result of Vahedi and van der Stappen [18] which assumes a polygonal object in the plane. Vahedi and van der Stappen developed the squeezing and stretching caging idea to compute the caging configurations for planar polygons. The generalization suggests that the squeezing and stretching caging idea is a fundamental attribute giving structure to the configuration space of two-fingered manipulators.

Sect. 5.5 shows that Theorem 5.1 addresses the immobilization problem. From any caging configuration there is a *blind policy* to immobilize the object. The result says that caging regions are partially bounded by immobilizations, and that there is

always a policy to reach them. Therefore, caging thickens the zero measure set of immobilizing configurations to regions of positive measure.

Does the result generalize to n fingers? One approach is to define an n finger manipulator with a one-dimensional shape space [2], but there are other possible generalizations. One advantage of the topological approach is that the entire proof is independent of dimension, and should yield a natural generalization of the squeezing-stretching concept to the case of n fingers.

Acknowledgements. This work is sponsored by the Defense Advanced Research Projects Agency. This work does not necessarily reflect the position or the policy of the U.S. Government. No official endorsement should be inferred. Alberto Rodriguez has been awarded and sponsored by the graduate fellowships of Caja Madrid and La Caixa.

Appendix - Contractible Paths

Proposition 6.1 (Characterization of contractible paths). *A closed path α at c is contractible in M^{free} if and only if $\Pi_i(\alpha)$ describes a contractible path in P_i^{free} $\forall i$.*

Proof. The result is a consequence of M being the cartesian product of the configuration space of each finger point, $M = \otimes_{i=1}^{n} P_i$. Let's prove both implications:

[$\Rightarrow$] Suppose α is contractible in M^{free}. Let $H(t,s)$ be the corresponding path homotopy. The natural projections of α are the closed curves $\Pi_i(\alpha) \subset P_i^{\text{free}}$. We need to show that α_i is contractible in P_i^{free} $\forall i$.

Consider then the natural projections of the homotopy $H(s,t)$:

$$\begin{array}{ccccc} H_i : [0,1]\times[0,1] & \xrightarrow{H} & M^{\text{free}} & \xrightarrow{\Pi_i} & P_i^{\text{free}} \\ (t,s) & \rightarrow & H(t,s) & \rightarrow & \Pi_i(H(t,s)) \end{array} \tag{9}$$

Each H_i is a continuous map because it is a composition of continuous maps ($H_i = \Pi_i \circ H$). Each H_i is a homotopy of paths because:

$$\begin{array}{ll} H_i(t,0) = \Pi_i(H(t,0)) = \Pi_i(\alpha(t)) = \alpha_i(t) & \forall t \\ H_i(t,1) = \Pi_i(H(t,1)) = \Pi_i(c) & \forall t \\ H_i(0,s) = H_i(1,s) = \Pi_i(H(0,s)) = \Pi_i(c) & \forall s \end{array} \tag{10}$$

We conclude that each α_i is contractible.

[$\Leftarrow$] Suppose that each natural projection $\alpha_i = \Pi_i(\alpha)$ is contractible in its corresponding space P_i^{free}. Let $H_i(t,s) \subset P_i^{\text{free}}$ be the corresponding path homotopy.

Consider the path in M, $\alpha = (\alpha_1 \ldots \alpha_n)$. Note that by construction $\alpha \subset M^{\text{free}}$:

> Suppose $\exists\, t \mid \alpha(t) \notin M^{\text{free}} \overset{eq.(1)}{\iff} \alpha(t) \in O^M \iff \exists\, i \mid \alpha(t) \in O_i^M$. By definition of O_i^M that happens iff $\alpha_i(t) \in O$. However this contradicts $\alpha_i(t)$ being defined on P_i^{free} by hypothesis. Therefore we conclude that $\alpha \subset M^{\text{free}}$ and is well defined.

Consider now the map $H = (H_1(t,s) \ldots H_n(t,s))$. same way we proved it for α we know that $H(t,s) \subset M^{\text{free}}$, and therefore is well defined. It suffices to check that:

$$\begin{aligned} H(t,0) &= (\alpha_1(t)\dots\alpha_n(t)) = \alpha(t) && \forall t \\ H(t,1) &= (\alpha_1(0)\dots\alpha_n(0)) = p && \forall t \\ H(0,s) &= H(1,s) = (\alpha_1(0)\dots\alpha_n(0)) = p && \forall t \end{aligned} \tag{11}$$

to conclude that H is a path homotopy for α and therefore, α is contractible.

References

1. Besicovitch, A.S.: A net to hold a sphere. The Mathematical Gazzete 41, 106–107 (1957)
2. Davidson, C., Blake, A.: Caging planar objects with a three-finger one-parameter gripper. In: Internatinal Conference Robotics and Automation, pp. 2722–2727. IEEE Press, Los Alamitos (1998)
3. Kuperberg, W.: Problems on polytopes and convex sets. In: DIMACS Workshop on polytopes, January 1990, pp. 584–589 (1990)
4. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
5. Laumond, J.P.: Feasible trajectories for mobile robots with kinematic and environment constraints. In: Intelligent Autonomous Systems, pp. 346–354. North-Holland Publishing Co., Amsterdam (1987)
6. Lee, J.M.: Introduction to Topological Manifolds. Graduate Texts in Mathematics. Springer, Heidelberg (2000)
7. Lee, J.M.: Introduction to Smooth Manifolds. Graduate Texts in Mathematics. Springer, Heidelberg (2002)
8. Milnor, J.W.: Topolgy from the Differentiable Viewpoint. Princeton University Press, Princeton (1997)
9. Pereira, G.A.S., Campos, M.F.M., Kumar, V.: Decentralized algorithms for multi-robot manipulation via caging. The International Journal of Robotics Research 23(7), 783–795 (2004)
10. Rimon, E., Blake, A.: Caging 2d bodies by one-parameter, two-fingered gripping systems. In: Internatinal Conference Robotics and Automation, pp. 1458–1464. IEEE Press, Los Alamitos (1996)
11. Rimon, E., Blake, A.: Caging planar bodies by one-parameter two-fingered gripping systems. The International Journal of Robotics Research 18(3), 299–318 (1999)
12. Rimon, E., Burdick, J.: On force and form closure for multiple finger grasps. In: Internatinal Conference Robotics and Automation, pp. 1795–1800. IEEE Press, Los Alamitos (1996)
13. Shephard, G.: A sphere in a crate. Journal of London Mathematical Society 40, 433–434 (1965)
14. Sudsang, A.: Grasping and in-hand manipulation: Geometry and algorithms. Algorithmica 26(3), 466–493 (2000)
15. Sudsang, A., Ponce, J.: On grasping and manipulating polygonal objects with disc-shaped robots in the plane. In: International Conference Robotics and Automation, Leuven, Belgium, pp. 2740–2746. IEEE Press, Los Alamitos (1998)
16. Sudsang, A., Ponce, J.: A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane. In: International Conference Robotics and Automation, San Francisco, USA, pp. 1068–1075. IEEE Press, Los Alamitos (2000)
17. Sudsang, A., Ponce, J., Srinivasa, N.: Grasping and in-hand manipulation: Experiments with a reconfigurable gripper. Advanced Robotics 12(5), 509–533 (1998)
18. Vahedi, M., van der Stappen, F.: Caging polygons with two and three fingers. In: Workshop on the Algorithmic Foundations of Robotics (2006)

A State Transition Diagram for Simultaneous Collisions with Application in Billiard Shooting

Yan-Bin Jia, Matthew Mason, and Michael Erdmann

Abstract. This paper models a multibody collision in the impulse space as a state transition diagram, where each state represents a phase during which impacts are "active" at only a subset of the contact points. A state transition happens whenever an active impact finishes restitution, or an inactive impact gets reactivated, depending on whether the two involved bodies are instantaneously penetrating into each other or not. The elastic energy due to an impact is not only affected by the impulse at the corresponding contact point, but also by other impulses exerted on the two involved bodies during the impact. Consequently, Poisson's impulse-based law of restitution could result in negative energy. A new law governing the loss of elastic energy during restitution is introduced. Convergence of the impulse sequence generated by the state transition diagram is established. The collision outcome depends on the ratios of the contact stiffnesses rather than on their individual values. The collision model is then applied in an analysis of billiard shooting in which the cue stick impacts the cue ball, which in turn impacts the pool table. The system is driven by the normal impulses at the two contacts with the tangential impulses determined via a contact mode analysis.

1 Introduction

Analysis of frictional impact has been a subject of controversy in order to be consistent with Coulomb's law of friction, Poisson's hypothesis of restitution, and the

Yan-Bin Jia
Iowa State University, Ames, IA 50011, USA
e-mail: jia@cs.iastate.edu

Matthew Mason
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: mason+@cs.cmu.edu

Michael Erdmann
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: me@cs.cmu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 135–150.
springerlink.com

law of energy conservation. It requires correct detection of contact modes (sliding, sticking, reverse sliding) and impact phases (compression and restitution). When the sliding direction stays constant (with possible reversals), the tangential impulse can be determined from the normal impulse based on Coulomb's law via case-based reasoning. The total impulse stays in the plane and grows along a polyline. Routh [15] developed a graphical method that constructs the trajectory of impulse accumulation based on Poisson's hypothesis. It was applied in the subsequent studies of two-dimensional rigid-body collisions with friction by Han and Gilmore [6] and by Wang and Mason [18] who classified impact and contact modes and offered a solution for each case. Later, Ahmed *et al.* [1] extended Routh's method to impact analysis for multibody mechanical systems with a similar classification.

To an impact in three dimensions, however, Routh's method hardly applies since the impulse builds along a space curve. The sliding direction generally varies during the impact. A differential equation in the normal impulse can be set up and solved to determine how the sliding direction varies in the course of the impact, as shown by Keller [9]. Closed-form solution does not exist for many three-dimensional impact problems.

This paper deals with simultaneous collisions in three dimensions. No existing impact laws are known to model the physical process well. Previous methods either sequence them into two-body collisions [5] by order of normal approach velocity, or set up linear complementarity conditions at all contacts [16, 2]. High-speed photographs of such collisions nonetheless show that multiple objects are simultaneously in contact rather than two at a time [17].

Stewart [17] pointed out that one difficulty with multiple contacts lies in the lack of a continuous impact law. Observations seem to suggest, during simultaneous collisions, the involved objects may have broken and re-established contacts multiple times. We represent the collision process as a sequence of states based on which impacts are instantaneously "active", or equivalently, which contacts are instantaneously effective. During a state, *multiple impacts may be acting upon one body.* A transition from one state to another happens when either an active impact finishes restitution or an inactive impact gets reactivated. A state transition diagram is introduced in Section 2 via the example of a ball falling onto another resting on the table.

The impulses produced by a pair of active impacts accumulate at a relative rate determined by themselves and by the stiffness ratio. The elastic energy stored at one contact point is no longer affected by just the impulse at this point, but also by those at other contact points involving one or both of these two bodies in the duration of this impact, which could span multiple states. Poisson's hypothesis may lead to overgrowth of an impulse during restitution, driving the contact elastic energy negative sometimes. The solution is to introduce a new law of restitution that *oversees the loss of elastic energy not the growth of impulse.*[1] We will also see that the outcome of impact is affected by the ratios of stiffnesses at the contact points. This

[1] The roles of the coefficients of restitution and friction were discussed in respect of energy loss [4], and solutions were given to two planar single-impact problems.

concept of "relative stiffness" is cited in [17] as missing from the current impact literature.

Section 3 applies the state transition diagram to model billiard shooting with a cue stick. This is a three-dimensional problem with simultaneous impacts between the cue stick and the cue ball, and between the cue ball and the pool table. Analysis of frictional contact modes is required.

In Section 4, we will discuss the extension to simultaneous collisions involving three or more bodies, and introduce an ongoing project to build a robot pool player.

2 System of Two Balls

We start by considering the problem of a rigid ball $\mathscr{B}_1$ with mass m_1 and velocity $v_0 < 0$ striking down onto another rigid ball $\mathscr{B}_2$ with mass m_2 and resting on a table. The centers of the two balls are vertically aligned, as shown on the left in Fig. 1. The lower ball $\mathscr{B}_2$ in turn impacts the table. Let v_1 and v_2 be the respective velocities ($<$ 0 if downward) of the two balls during the collision, where the gravitational forces are negligible compared to the large impulsive forces. Our goal is to determine the ball velocities at the end.

We attach a virtual spring between $\mathscr{B}_1$ and $\mathscr{B}_2$, and another one between $\mathscr{B}_2$ and the table, as shown on the right in Fig. 1. Let x_1 and x_2 be the changes in the lengths of these springs, which have stiffnesses k_1 and k_2, respectively. The kinematic and dynamic equations are given below:

$$\dot{x}_1 = v_1 - v_2, \tag{1}$$

$$\dot{x}_2 = v_2, \tag{2}$$

$$m_1\dot{v}_1 = F_1 = -k_1x_1, \tag{3}$$

$$m_2\dot{v}_2 = F_2 - F_1 = k_1x_1 - k_2x_2, \tag{4}$$

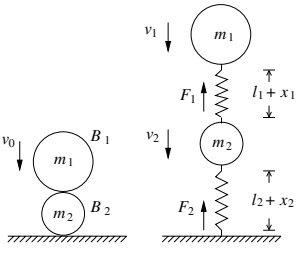


Fig. 1. Two-ball collision.

where F_1 and F_2, both positive, are the contact forces, and the dot '$\cdot$' denotes differentiation with respect to time. The above are a system of four differential equations in four variables v_i and x_i, $i = 1, 2$.

Since an impact happens in infinitesimal time, it is best analyzed in the impulse space. During the two-ball collision, there are two impulses: $I_1 = \int_{t_0}^{t} F_1\,dt$ and $I_2 = \int_{t_0}^{t} F_2\,dt$ with initial values $I_1^{(0)} = I_2^{(0)} = 0$. Integration of (3) and (4) yields the ball velocities in terms of the impulses:

$$v_1 = v_1^{(0)} + \frac{1}{m_1}\Delta I_1, \qquad \text{where } \Delta I_1 = I_1 - I_1^{(0)}, \tag{5}$$

$$v_2 = v_2^{(0)} + \frac{1}{m_2}(\Delta I_2 - \Delta I_1), \qquad \text{where } \Delta I_2 = I_2 - I_2^{(0)}. \tag{6}$$

The two virtual springs store elastic energies $E_1 = \frac{1}{2}k_1x_1^2$ and $E_2 = \frac{1}{2}k_2x_2^2$, respectively. To eliminate x_1, from equations (1), (3), (5), (6) we obtain

$$\frac{dx_1}{dI_1} = \frac{\dot{x}_1}{\dot{I}_1} = \frac{v_1^{(0)} - v_2^{(0)} + (\frac{1}{m_1} + \frac{1}{m_2})\Delta I_1 - \frac{1}{m_2}\Delta I_2}{-k_1x_1}. \tag{7}$$

Multiply both sides of the above equation with $-k_1x_1dI_1$ and then integrate. We obtain the change in the elastic energy E_1 from its initial value $E_1^{(0)}$:

$$\Delta E_1 = \left(v_2^{(0)} - v_1^{(0)}\right)\Delta I_1 - \frac{1}{2}\left(\frac{1}{m_1} + \frac{1}{m_2}\right)\Delta I_1^2 + \frac{1}{m_2}\int_{I_1^{(0)}}^{I_1} \Delta I_2\, dI_1. \tag{8}$$

Similarly, from equations (2) and (6) we derive the change in E_2 from $E_2^{(0)}$:

$$\Delta E_2 = -v_2^{(0)}\Delta I_2 - \frac{1}{2m_2}\Delta I_2^2 + \frac{1}{m_2}\int_{I_2^{(0)}}^{I_2} \Delta I_1\, dI_2. \tag{9}$$

A relationship between the two impulses I_1 and I_2 can be now set up:

$$\frac{dI_2}{dI_1} = \frac{\dot{I}_2}{\dot{I}_1} = \frac{k_2x_2}{k_1x_1} = \frac{\sqrt{k_2E_2}}{\sqrt{k_1E_1}} = \sqrt{\frac{k_2}{k_1}} \cdot \sqrt{\frac{E_2^{(0)} + \Delta E_2}{E_1^{(0)} + \Delta E_1}}. \tag{10}$$

With no closed-form solution to (10) in general, the impact process is simulated via numerical integration with a step size of I_1, say, h. To initialize $\rho = \frac{dI_2}{dI_1}(h)$, we plug into (8)–(10) the values $I_1(h) = h$, $I_2(h) \approx \rho h$, and $\int_0^h \Delta I_2\, dI_1 = \int_0^{\rho h} \Delta I_1\, dI_2 \approx \frac{1}{2}\rho h^2$. Solve the resulting quadratic equation in ρ:

$$\frac{dI_2}{dI_1}(h) = \frac{2k_2}{k_1 \cdot (b + \sqrt{b^2 + 4k_2/k_1})}, \text{ where } b = -2 \cdot \frac{m_2v_0}{h} - 1 - \frac{m_2}{m_1} + \frac{k_2}{k_1}. \tag{11}$$

As h tends to zero, b goes to infinity. Hence $\frac{dI_2}{dI_1}(0) = \lim_{h\to 0}\rho = 0$.

2.1 State Transition Diagram

An impact is divided into two stages [12, p. 212]: compression and restitution. In the classical problem of a particle of mass m with downward velocity v_0 striking a horizontal table, the impact ends compression when the velocity becomes zero, which gives the impulse $I = -mv_0$. Poisson's hypothesis states that I will accumulate $-emv_0$ more during restitution to yield the final velocity $-ev_0$, where e is the coefficient of friction. Setting up a virtual spring at the contact point, we can derive the elastic energy $E = -v_0I - \frac{1}{2m}I^2$. Restitution starts with E assuming the maximum value $\frac{1}{2}mv_0^2$ and ends with loss of energy $\frac{1}{2}mv_0^2 \cdot (1 - e^2)$. Since $0 \le e \le 1$,

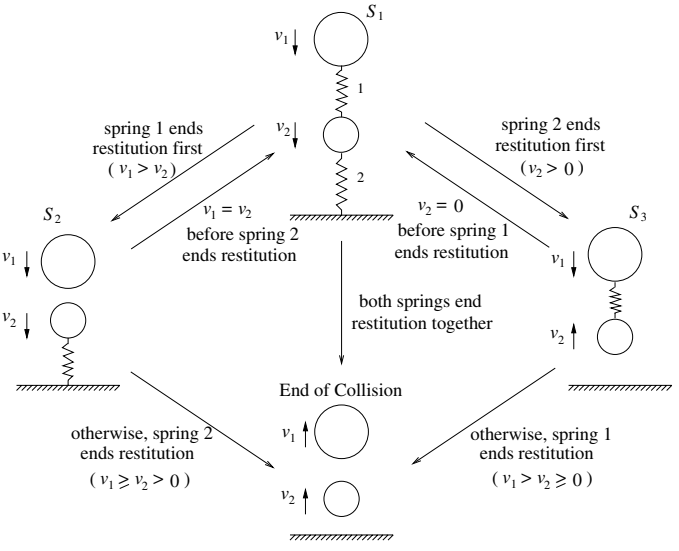


Fig. 2. State transition diagram for the two-ball collision in Fig. 1. During each state, v_1 and v_2 can be either upward or downward, except v_2 is upward in S_3.

there is always enough elastic energy to provide the impulse accumulation $-emv_0$ during restitution.

Coming back to the two-ball collision problem, the ball-ball and ball-table impacts have coefficients of restitution $e_1, e_2 \in [0,1]$, respectively. Compressions end when $\dot{x}_1 = 0$ and $\dot{x}_2 = 0$, respectively; or equivalently, by (1), (2), (5), and (6), when

$$v_0 + \left(\frac{1}{m_1} + \frac{1}{m_2}\right) I_1 - \frac{1}{m_2} I_2 = 0 \quad \text{and} \quad I_1 = I_2, \quad \text{respectively.} \tag{12}$$

The two impacts will hardly start restitution at the same time, neither will they end restitution so. When one of them, say, between the two balls, finishes restitution first, the other one (between the ball and the table) will continue. As a result, the two balls may start moving toward each other at some point later, reactivating the first impact.

The above discussion suggests us to partition the collision process into (repeats of) three states: S_1 when both impacts are active, S_2 when only the ball-table impact is active, and S_3 when only the ball-ball impact is active. Fig. 2 shows a state transition diagram. The collision starts with the state S_1. A transition from S_1 to S_2 happens when the ball-ball impact finishes restitution before the ball-table impact. So the two balls are "breaking" contact momentarily. Since the impulse I_1 was in restitution just before the transition, $\dot{x}_1 > 0$, which by (11) implies $v_1 > v_2$ when S_2 begins. Because gravity is neglected during the collision, v_1 will not vary during S_2. The state will transition back to S_1 when v_2 increases to become equal to v_1 before

restitution ends. If this does not happen, the ball-table impact will finish restitution with $v_1 \geq v_2$, hence the end of collision.

Similarly, a transition from S_1 to S_3 happens when the ball-table impact finishes restitution before the ball-ball impact. The state S_3 will transition to S_1 when $v_2 = 0$, that is, when the lower ball is "re-establishing"contact with the table. Otherwise, the collision will end within the state.

The transition diagram describes the collision as a sequence of states, each being one of S_1, S_2, S_3. Now, I_i, $i = 1, 2$, represent the impulses accumulated since the start of the collision. An impact may start with one state, end compression in another, and finish restitution with a third.

By induction on the number of states, we can generalize (5) and (6):

$$v_1 = v_0 + \frac{1}{m_1} I_1 \quad \text{and} \quad v_2 = \frac{1}{m_2}(I_2 - I_1). \tag{13}$$

It is easy to show that the conditions (12) respectively hold when the two impacts end their compressions in a state.

2.2 An Energy-Based Model

The superscript '(0)' continues to refer to the value of a physical quantity at the start of a state, and the notation 'Δ' its increment so far in the state. The relationship (8) between ΔE_1 and ΔI_1 in the state S_1 depends on the masses and initial velocities of both balls, as well as an integral of ΔI_2 over I_1. If we were to let the impulse I_1 accumulate by a factor of e_1 after restitution under Poisson's hypothesis, there may not be enough elastic energy E_1 left to provide such an increase.[2] To deal with multiple simultaneous impacts, we limit *the amount of energy to be released during restitution relative to the amount accumulated during compression*. Since in the single particle impact case, the loss of energy is $\frac{1}{2}mv_0^2 \cdot (1 - e^2)$, we see that e^2 is the needed ratio.

When compression ends, the elastic energy is at its maximum $E_{\max}$. *Restitution will finish when* $E = (1 - e^2)E_{\max}$. The remaining amount $e^2 E_{\max}$ can be seen as lost at the state transition instead of at the end of compression. In this view, during a state, equations (8) and (9) hold for our convenience, while the total (elastic and dynamic) energy is conserved.

Single-Impact States. The state S_2 starts with $v_2^{(0)} < v_1^{(0)}$ since restitution of the ball-ball impact has just finished. During the state, $v_1 \equiv v_1^{(0)}$ and $\Delta I_1 \equiv 0$. From (6), we conclude that restitution, if not in process, would happen during the state when $\Delta I_2 = -m_2 v_2^{(0)}$ with $E_{2\max} = E_2^{(0)} + \frac{1}{2} m_2 {v_2^{(0)}}^2$ by (9). The next state will be S_1 if S_2 starts during compression and $v_1^{(0)} < 0$. Since v_2 increases toward zero by (6), it will reach $v_1^{(0)}$ before compression ends, hence the transition to S_1. Under the new energy-based model, a transition to S_1 will also happen if S_2 starts during

[2] An example will be given at the end of Section 2.3.

restitution with $E_2^{(0)} + \frac{1}{2}m_2 v_2^{(0)^2} > \frac{1}{2}m_2 v_1^{(0)^2} + (1-e_2^2)E_{2\max}$. If neither case of transition happens, the collision will end. The impulse accumulation during S_2 is

$$\Delta I_2 = \begin{cases} m_2(v_1^{(0)} - v_2^{(0)}), & \text{if } S_1 \text{ next,} \\ m_2\left(\sqrt{v_2^{(0)^2} + 2\frac{E_2^{(0)} - (1-e_2^2)E_{2\max}}{m_2}} - v_2^{(0)}\right), & \text{if impact ends.} \end{cases}$$

A similar analysis based on (8) applies to S_3 in determining whether the collision will end or S_1 will follow, and the amount ΔI_1 during S_3.

Double-Impact State. Evolution in the state S_1 is governed by the differential equation (10) with increasing impulses I_1 and I_2. If S_1 is the start of the collision or follows S_3, I_1 is the *primary* impulse (variable), and I_2 is the *secondary* impulse (function of I_1). If S_1 follows S_2, the roles of the two impulses reverse. Similar to (11), we initialize the impulse derivatives $\frac{dI_1}{dI_2}(h)$ or $\frac{dI_2}{dI_1}(h)$ accordingly for numerical integration. In these two cases, $\frac{dI_1}{dI_2}(0) = 0$ and $\frac{dI_2}{dI_1}(0) = 0$, respectively.

At each step of the numerical integration of (10), we check (12) to see if compression has just ended for either impact, and if so, set the maximum elastic energy $E_{1\max}$ or $E_{2\max}$ accordingly. The state transitions to S_2 when $E_1 = (1-e_1^2)\cdot E_{1\max}$ during restitution of the ball-ball impact, or to S_3 when $E_2 = (1-e_2^2)\cdot E_{2\max}$ during restitution of the ball-table impact, whichever occurs earlier.

2.3 The Impulse Curve

In the state S_1, the differential equation (10), along with (8) and (9), has only one occurrence of the stiffness ratio $\frac{k_2}{k_1}$ but none of k_1 or k_2 separately. Meanwhile, the outcome of the states S_2 and S_3 are independent of k_1 or k_2.

Theorem 2.1. *The outcome of the collision depends on the stiffness ratio k_1/k_2 but not on individual values of k_1 and k_2.*

The next theorem bounds the total elastic energy using the impulses.

Theorem 2.2. *The following is satisfied during the collision:*

$$0 \le E_1 + E_2 \le -v_0 I_1 - \frac{1}{2m_1}I_1^2 - 12m_2(I_1 - I_2)^2. \tag{14}$$

Proof of the theorem is by induction on the number of states while making use of equations (8), (9), and (13). Details are omitted.

In the plane with I_1 and I_2 as the two axes, the *impulse curve* describes the evolution of their values during the collision. Theorem 2.2 states that this curve is bounded by an ellipse:

$$\frac{1}{2m_1}I_1^2 + \frac{1}{2m_2}(I_1 - I_2)^2 + v_0 I_1 = 0. \tag{15}$$

In Fig. 3, the ellipse has its semi-minor axis rotated from the I_1-axis by $\theta = \frac{1}{2}\arctan\left(-\frac{2m_1}{m_2}\right)$ and its center at $\left(\frac{-v_0\cos\theta}{\frac{\cos^2\theta}{m_1}+\frac{1-\sin(2\theta)}{m_2}}, \frac{v_0\sin\theta}{\frac{\sin^2\theta}{m_1}+\frac{1+\sin(2\theta)}{m_2}}\right)$. It is tangent to the I_2-axis at the origin and to the line $I_1 = -2m_1v_0$ at $(-2m_1v_0, -2m_1v_0)$.[3]

The impulse curve is monotone in the sense that I_1 and I_2 never decrease. This is clear if the state is S_2 or S_3 in which one of the impulses increases while the other does not vary. After S_1 starts, the strain energies $E_1, E_2 > 0$, which implies the derivative $dI_2/dI_1 = \sqrt{k_2E_2}/\sqrt{k_1E_1} > 0$, hence the monotonicity.

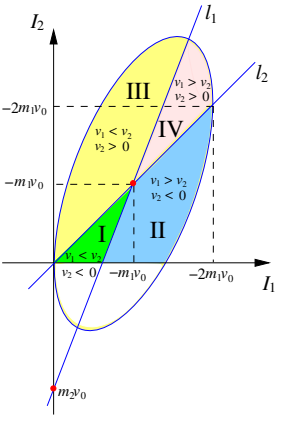


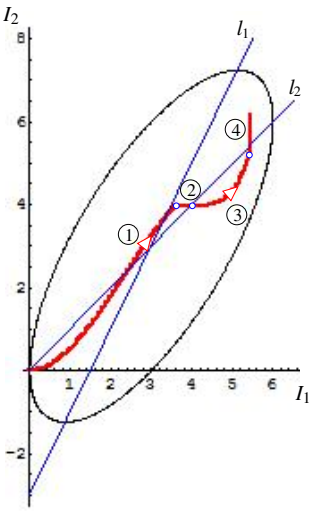
Fig. 3. Impulse plane.

Add two lines ℓ_1 and ℓ_2 defined by equations (12). Referred to as the *compression lines*, they partition the feasible elliptic region ($I_1 \geq 0$) into four smaller regions I–IV. The impulse curve evolves from the origin into region I within the state S_1 as $\frac{dI_2}{dI_1}$ increases from 0. As I_1 increases unconstrained, the curve will cross ℓ_1 (or ℓ_2) when the ball-ball impact (or the ball-table impact) ends compression. During the state S_2, the curve stays to the right of the line ℓ_1, and evolves vertically upward, ending either inside region IV (in which case the collision ends) or on the line ℓ_1 for a transition to S_1. Similarly, during S_3, the curves stays to the left of ℓ_2 and evolves horizontally to the right, ending either inside the region IV or on the line ℓ_2 for a transition to S_1.

Fig. 4 illustrates a collision instance which results in a sequence of four states S_1, S_3, S_1, S_2. In (a), the impulse curve is plotted, along with the bounding ellipse: $-3I_1 + \frac{1}{2}I_1^2 + \frac{1}{2}(I_1 - I_2)^2 = 0$ and the two compression lines ℓ_1: $-3 + 2I_1 - I_2 = 0$ and $\ell_2 : I_1 = I_2$. The impulse segments corresponding to different states are labeled in the order and separated by the dots. The first segment (of a S_1 state) crosses ℓ_2 before ℓ_1, indicating that the ball-table impact goes into restitution before the ball-ball impact. The ball-table impact subsequently finishes restitution. This is marked as C2-C1-R2 in table (b), where each row describes the status at the end of a state. Diagram (c) in the figure plots the evolution of the elastic energies E_1 and E_2 with four segments also labeled in the state order. It shows energy losses at the end of all but the second states. The loss in the total (elastic and dynamic) energy E during the collision is calculated to be 1.2494.

Suppose restitution were decided by impulse accumulation according to Poisson's hypothesis for single impact. The same collision would generate an impulse curve exiting the bounding ellipse (15) in the fourth state — the increase of I_1 to the required value 5.906 would result in a negative elastic energy ($E_1 = -1.01$).

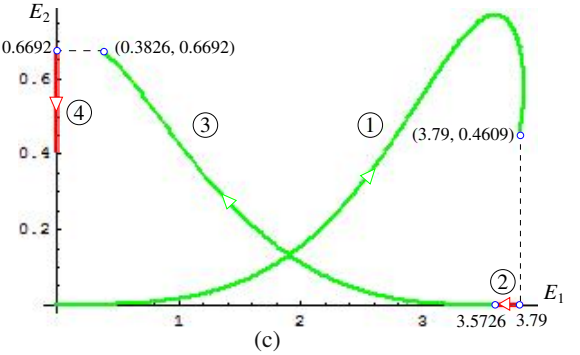
[3] The two horizontal bounding lines are $I_2 = (-m_1 \pm \sqrt{m_1m_2})v_0$ with points of tangency to the ellipse at $I_1 = -m_1v_0$.



(a)

state	v_1	v_2	I_1	I_2	Comp. Res.
S_1	0.6088	0.3573	3.6085	3.9661	C2-C1-R2
S_3	0.9661	0	3.9661	3.9661	
S_1	2.4414	−0.1211	5.4414	5.3202	R1
S_2	2.4414	0.7355	5.4414	6.1769	C2-R2

(b)



(c)

Fig. 4. The impulse and energy curves for a collision of two balls as in Fig. 1 with masses $m_1 = m_2 = 1$kg, the stiffness ratio $\frac{k_2}{k_1} = 10$, the coefficients of restitution $e_1 = \sqrt{0.9}$ and $e_2 = \sqrt{0.4}$, and the upper ball velocity $v_0 = -3$m/s.

Every state terminates. This is trivial for S_2 and S_3 for which ΔI_1 and ΔI_2 are given in Section 2.2. Within the state S_1, the 'primary impulse' must stop increasing because the impulse curve is bounded inside the ellipse (15).

Denote by $I_1^{(i)}$ and $I_2^{(i)}$ the values of the two impulses at the end of the ith state. The sequence $\{(I_1^{(i)}, I_2^{(i)})\}$ is monotone non-decreasing. In case it is finite, the state transitions terminate with $v_1 \geq v_2 \geq 0$ according to the diagram in Fig. 2 as the impulse curve stops inside region IV in Fig. 3 or on it boundary.[4] In case the sequence is infinite, because it is bounded inside the ellipse (15), by a result from calculus it must converge to some point (I_1^*, I_2^*). We can show that this point must lie on the boundary of region IV.

Theorem 2.3. *The state transitions will either terminate with $v_1 \geq v_2 \geq 0$ or the generated impulse sequence will converge with either $v_1 = v_2 \geq 0$ or $v_1 > v_2 = 0$.*

As v_0 scales by a factor of s, we can show that throughout the collision the impulses I_1, I_2 and the velocities v_1, v_2 scale by s while the elastic energies E_1 and E_2 scale by s^2. The differential equation (10) still holds after the scaling, as well as the conditions (12) for ending of compressions and the conditions on state transitions.

Theorem 2.4. *At the end of the collision, the ratios v_1/v_0 and v_2/v_0 are constants depending on m_1, m_2, e_1, e_2 and the stiffness ratio k_1/k_2 only.*

[4] The curve will reach the bounding ellipse at termination only if $e_1 = e_2 = 1$.

2.4 Preliminary Experiment

To validate the collision model, we let a ping pong ball $\mathscr{B}_1$ fall onto another one $\mathscr{B}_2$ resting on a plexiglass block. The ball has mass 0.00023kg and radius 0.019m. The block is placed horizontally on the marker tray of a (vertical) office whiteboard, and against a vertical axis ℓ drawn on the board. The ball $\mathscr{B}_1$ is held in the hand. Both balls are positioned almost in contact with the whiteboard such that ℓ "passes through" their centers in the frontal view.

To measure the coefficient of restitution e_2 between a ball and the plexiglass surface, we drop the ball from certain height h_1 onto the surface and record the rebounding height h_2 (on the axis by human vision). Thus $e_2 \approx \sqrt{h_2/h_1}$. Sixteen measurements from different heights (with four balls) have generated a mean estimate of 0.846529 with a standard deviation of 0.020827. To measure the coefficient of ball-ball restitution e_1, $\mathscr{B}_2$ is held steady on the surface, and $\mathscr{B}_1$ is dropped from the same height onto $\mathscr{B}_2$ multiple times with the highest rebound (from the closest-to-a-perfect hit) recorded. The mean value of e_1 calculated over eight different dropping heights is 0.807755 with standard deviation 0.021231.

A collision trial involves dropping $\mathscr{B}_1$ from a fixed height onto $\mathscr{B}_2$ multiple times, and choosing the one with the highest rebounds of both balls. The input velocity v_0 and the output velocities v_1 and v_2 are calculated. Results from ten trials, each with a different dropping height, are plotted in Fig. 5 as pairs of points $(-v_0, v_1)$ and $(-v_0, v_2)$. Also shown are the two lines v_1 and v_2 as v_0 varies from -2m/s to -4m/s. Despite the rather primitive setup and measurement method, the result suggests a reasonably good match between the model and the physical collision process.

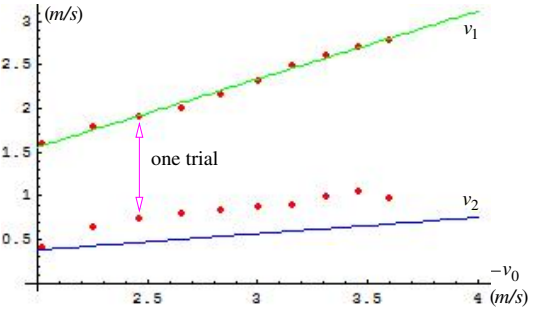


Fig. 5. Collisions between two ping pong balls and a plexiglass surface: experimental results (dots) vs. predictions (lines) by the impact model (with guess $k_2/k_1 = 10$).

3 Shooting a Billiard

We apply the impact model to the problem of a cue stick shooting the cue ball in the game of pool, as illustrated in Fig. 6. The cue stick has initial velocity v_{c0}. Let c be the unit vector $v_{c0}/\|v_{c0}\|$, n the unit normal at the point of impact on the ball, and z the unit normal at the table contact. The condition $n \cdot c < 0$ must hold for the shot to happen. During the shot, we assume that the cue stick is constrained to move along

c or $-c$.[5] The cue stick has velocity v_c, the ball has velocity v and angular velocity ω, all varying during the shot.

Denote by I_1 and I_2 the impulses at the cue-ball and the ball-ball contacts, respectively, as shown in the figure. The impulse I_1 consists of a normal component $I_{1n}n$ and a tangential component $I_{1\perp}$. The impulse I_2 consists of a vertical component $I_{2z}z$ and a horizontal component $I_{2\perp}$. We have $I_{1n} > 0$ or $I_{2z} > 0$ whenever the corresponding impact is active.

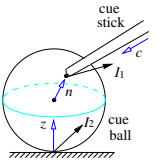


Fig. 6. Pool shot.

Two virtual springs, with stiffnesses k_{cb} and k_{bt}, are attached at the points of impact in alignment with the normals n and z, respectively. Based on the impact model, the shot by the cue stick has three states: S_1 (illustrated in Fig. 7) during which both the ball-ball and the ball-table impacts are active, S_2 during which only the ball-table impact is active, and S_3 during which only the ball-ball impact is active. The shot starts with S_1 and ends in either S_2 or S_3.

Denote by v_{cb} the relative velocity of the cue stick to the ball, and by v_{bt} that of the ball to the pool table. The state transition diagram has the same structure as that in Fig. 2 except in the transition conditions, $v_1 - v_2$ and v_2 are respectively replaced by the normal velocity components $v_{\text{cb}} \cdot n$ and $v_{\text{bt}} \cdot z$. A cue-ball impact is in compression when $v_{\text{cb}} \cdot n < 0$. A ball-table impact is in compression when $v_{\text{bt}} \cdot z < 0$.

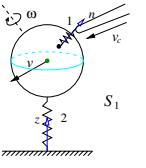


Fig. 7. State S_1.

If the cue stick shoots the ball below its equator (i.e., $n \cdot z < 0$) or at the equator horizontally or upward, only the cue-ball impact exists. In this case, the transition diagram has only one state — S_3.

3.1 Dynamics and Impact State Analysis

The symbols (0) and Δ carry the same meanings as in Section 2. For instance, $\Delta v_{\text{cb}} = v_{\text{cb}} - v_{\text{cb}}^{(0)}$ is the change in the relative velocity of the cue stick to the cue ball during a state from its starting value $v_{\text{cb}}^{(0)}$.

Let M be the mass of the cue stick. Let the ball have radius r and mass m, thus angular inertia $\frac{2}{5}mr^2$. Changes in the velocities during a state can be expressed in terms of the impulse accumulations ΔI_1 and ΔI_2:

$$\Delta v_{\text{c}} = \frac{1}{M}(\Delta I_1 \cdot c)c, \qquad \Delta v = \frac{1}{m}(\Delta I_2 - \Delta I_1), \tag{16}$$

$$\Delta \omega = \frac{5}{2}\frac{1}{mr^2}\Big(rn \times (-\Delta I_1) + (-rz) \times \Delta I_2\Big), \tag{17}$$

$$\Delta v_{\text{cb}} = \Delta v_{\text{c}} - \Delta v - \Delta \omega \times (rn), \qquad \Delta v_{\text{bt}} = \Delta v - \Delta \omega \times (rz). \tag{18}$$

[5] This is the case with our design of a mechanical cue stick to be shown in Fig. 8.

To find out how the normal impulses $I_{1n} = I_1 \cdot n$ and $I_{2z} = I_2 \cdot z$ are related to each other in the state S_1, we notice that the two virtual springs at the cue-ball and the ball-table contacts have their lengths change at the rates $\dot{x}_1 = n \cdot (v_{\text{cb}}^{(0)} + \Delta v_{\text{cb}})$ and $\dot{x}_2 = z \cdot (v_{\text{bt}}^{(0)} + \Delta v_{\text{bt}})$, respectively. From these rates, $\dot{I}_{1n} = -k_{\text{cb}} x_1$, and $\dot{I}_{2z} = -k_{\text{bt}} x_2$, we obtain the derivatives $\frac{dx_1}{dI_{1n}}$ and $\frac{dx_2}{dI_{2z}}$ as linear expressions in ΔI_1 and ΔI_2 over $-k_{\text{cb}} x_1$ and $-k_{\text{bt}} x_2$, respectively. Multiplying away the denominators in the derivative equations and integrating both sides of each equation, we obtain the changes in the elastic energies:

$$\Delta E_1 = E_1 - E_1^{(0)} = -n \cdot \left(\Delta I_{1n} v_{\text{cb}}^{(0)} + \left(\frac{1}{M} c c^T + \frac{1}{m} \right) D_1 - \frac{1}{m} D_2 \right), \tag{19}$$

$$\Delta E_2 = E_2 - E_2^{(0)} = -z \cdot \left(\Delta I_{2z} v_{\text{bt}}^{(0)} - \frac{1}{m} D_3 \right) - \frac{1}{m} D_4, \tag{20}$$

where the four integrals during the state are defined as $D_1 = \int_{I_{1n}^{(0)}}^{I_{1n}} \Delta I_1 \, dI_{1n}$, $D_2 = \int_{I_{1n}^{(0)}}^{I_{1n}} \Delta I_2 \, dI_{1n}$, $D_3 = \int_{I_{2z}^{(0)}}^{I_{2z}} \Delta I_1 \, dI_{2z}$, and $D_4 = \frac{1}{2} \Delta I_{2z}^2$. This sets up a differential relationship between I_{1n} and I_{2z}:

$$\frac{dI_{2z}}{I_{1n}} = \frac{\dot{I}_{2z}}{\dot{I}_{1n}} = \frac{-k_{\text{bt}} x_2}{-k_{\text{cb}} x_1} = \sqrt{\frac{k_{\text{bt}}}{k_{\text{cb}}}} \cdot \sqrt{\frac{E_2^{(0)} + \Delta E_2}{E_1^{(0)} + \Delta E_1}}. \tag{21}$$

The system of equations (19)–(21), along with a contact mode analysis, determines the evolution within the state S_1. A closed-form solution does not exist in general. Numerical integration is performed as follows.

Entering a state, we need to set dI_{1n} to h and compute dI_{2z}/dI_{1n} if at the start of the shot or the previous state is S_3, or set dI_{2z} to h and compute dI_{1n}/dI_{2z} if the previous state is S_2. The tangential impulse increments $dI_{1\perp}$ and $dI_{2\perp}$, as well as D_1, D_2, D_3, D_4, are also initialized. This is similar to that for the two-ball collision as described right before Section 2.1 but is much more involved since we need also determine the contact modes.

After initialization, iterate until one of the impacts ends restitution. At each iteration step, update v_{cb}, v_{bt} according to (18), and E_1, E_2 according to (19)–(20). In case one impact is starting restitution, set maximum elastic energy $E_{1\max}$ or $E_{2\max}$ accordingly. Compute dI_{1n} and dI_{2z} by (21), and $I_{1\perp}$ and $I_{2\perp}$ based on a contact mode analysis to be described below. The iteration step finishes with updating the values of $\Delta I_1, \Delta I_2, D_1, D_2, D_3, D_4$.

Contact Modes. Contact modes depend on the tangential components $v_{\text{cb}\perp}$ and $v_{\text{bt}\perp}$ of the two contact velocities v_{cb} and v_{bt}. Let μ_{cb} and μ_{bt} be the coefficients of friction between the cue tip and the ball and between the ball and the table, respectively. When a tangential velocity, say, $v_{\text{cb}\perp}$, is not zero, the cue tip is sliding on the ball. We have $dI_{1\perp} = -\mu_{\text{cb}} dI_{1n} \cdot \hat{v}_{\text{cb}\perp}$ under Coulomb's law, where $\hat{v}_{\text{cb}\perp} = v_{\text{cb}\perp} / \|v_{\text{cb}\perp}\|$. Similarly, $dI_{2\perp} = -\mu_{\text{bt}} dI_{2z} \cdot \hat{v}_{\text{bt}\perp}$ when the ball is sliding on the table.

When one tangential velocity is zero, there are three cases: (1) $v_{\mathrm{cb}\perp} = 0$ but $v_{\mathrm{bt}\perp} \neq 0$, (2) $v_{\mathrm{bt}\perp} = 0$ but $v_{\mathrm{cb}\perp} \neq 0$, and (3) $v_{\mathrm{cb}\perp} = 0$ and $v_{\mathrm{bt}\perp} = 0$. We here treat the first case only as the other two cases can be handled similarly.

In case (1), $dI_{2\perp} = -\mu_{\mathrm{bt}} dI_{2z} \cdot \hat{v}_{\mathrm{bt}\perp}$. We obtain the derivative of $v_{\mathrm{cb}\perp}$ with respect to I_{1n} in terms of those of the tangential impulses $I_{1\perp}$ and $I_{2\perp}$. To stay in sticking contact, $dv_{\mathrm{cb}\perp}/dI_{1n} = 0$, which determines the value of $dI_{1\perp}/dI_{1n}$. If $\|dI_{1\perp}/dI_{1n}\| \leq \mu_{\mathrm{cb}}$, the contact stays sticking. Otherwise, the contact starts sliding in the direction of $dv_{\mathrm{cb}\perp}/dI_{1n}$, which can be solved.

Ball-Table Impact Only. In the state S_2, $E_1 = 0$ and $I_1 = 0$. In the case that S_2 begins during compression, we set the maximum elastic energy $E_{2\max} = E_2^{(0)} + \frac{1}{2}m(v_{\mathrm{bt}}^{(0)} \cdot z)^2$. From (16)–(18) under $\Delta I_1 = 0$, the change in the tangential velocity is $\Delta v_{\mathrm{bt}\perp} = (1 - zz^T)\Delta v_{\mathrm{bt}} = \frac{7}{2m}\Delta I_{2\perp}$. So $v_{\mathrm{bt}\perp}$ and $I_{2\perp}$ will not change their directions during the state. Once $v_{\mathrm{bt}\perp}$ reduces to zero, it will stay zero so as not to contradict Coulomb's law. To make $v_{\mathrm{bt}}^{(0)}$ zero, $\Delta I_{2\perp} = -mv_{\mathrm{bt}}^{(0)}$, which requires $\Delta I_{2z} \geq m\frac{\|v_{\mathrm{bt}}^{(0)}\|}{\mu_{\mathrm{bt}}}$. Also, S_2 would switch to the state S_1 when $v_{\mathrm{cb}} \cdot n = 0$. We hypothesize the outcome of S_2 (a transition to S_1 or the end of collision), and in the first case, the contact mode (sticking or sliding). Then we test these hypotheses by checking some derived inequalities which depend on v, $v_{\mathrm{c}}^{(0)}$, $v_{\mathrm{bt}}^{(0)}$, $E_2^{(0)}$, ΔE_2, and $E_{2\max}$.

Cue-Ball Impact Only. Entering the state S_3 from S_1, the ball-table impact had just finished restitution, so $v_{\mathrm{bt}}^{(0)} \cdot z > 0$. Substituting $\Delta I_2 = 0$ into (18), we obtain the change in the tangential velocity: $\Delta v_{\mathrm{cb}\perp} = \frac{1}{M}(c \cdot \Delta I_1)c_\perp + \frac{7}{2m}\Delta I_{1\perp}$. Two special cases, $c = n$ and $c \cdot z = n \cdot z = 0$, can be treated with analyses similar to that for the case of ball-table impact only.

Generally, $I_{1\perp}$ varies its direction along a curve in the tangent plane. Numerical integration similar to that described earlier for the two-impact state is employed. The procedure is nevertheless simpler given only one impulse I_1.

3.2 *Billiard Simulation*

Table 1 shows four different shots and the trajectories[6] resulting from three of them. With some simplifications[7], the ball trajectory is completely determined by the x and y components of its velocity v and angular velocity ω. It is known that the ball will first slide along a parabolic arc (unless $\omega \cdot v = 0$) and then roll along a straight line before coming to a stop.

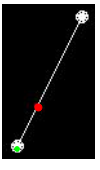
The first shot, vertical but not through the ball center, yields a straight trajectory in figure (a). The second shot (figure (b)), horizontal along the x-axis, hits the point at the polar angle $\frac{3\pi}{4}$ on the ball's equator. Due to friction, the ball trajectory forms a smaller angle with the x-axis than with the y-axis, exhibiting some effect of English.

[6] The trajectory equations are omitted, though some can be found in [11].

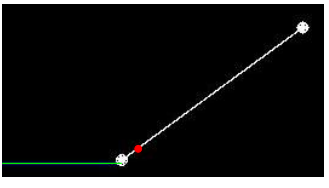
[7] We ignore the effects on the trajectory due to (possibly) multiple collisions between the ball and the table.

Table 1. Four shots at the cue ball (with the x-y plane on the pool table). Trajectories (a), (b), (c) are produced by the 1st, 2nd, 4th shots, respectively. On each trajectory, the red dot marks where sliding switches to pure rolling; and the green line represents the cue stick. We use the following measured physical constants: $m = 0.1673$kg, $M = 0.5018$kg, $r = 0.0286$m, $\mu_{bt} = 0.152479$, $e_{cb} = 0.656532$, and $e_{bt} = 0.51625$. We set $\mu_{cb} = 0.4$ [14] and the stiffness ratio $k_{cb}/k_{bt} = 1.5$.

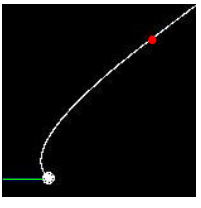
shot	n	$v_c^{(0)}$	v	ω
vertical	$\frac{(-1,-2,1)}{\sqrt{6}}$	$(0,0,-2)$	$(0.3963,0.7926,0.4506)$	$(27.97,-13.98,0)$
horizontal	$(-\frac{\sqrt{2}}{2},-\frac{\sqrt{2}}{2},0)$	$(0.8,0,0)$	$(0.6299,0.4614,0)$	$(0,0,10.42)$
jump	$(-\frac{\sqrt{2}}{2},0,-\frac{\sqrt{2}}{2})$	$(1,0,-1)$	$(1.169,0,0.1883)$	$(0,13.15,0)$
massé	$\frac{(-2,-1,12)}{\sqrt{149}}$	$(4,0,-16)$	$(-0.3253,0.2938,6.232)$	$(-88.48,179.8,-3.178)$



(a)



(b)



(c)

The third is a jump shot in the x-z plane. The last one, shown in figure (c), is a massé shot with the cue erected.

4 Discussion and Future Work

The introduced impact model makes use of the fact that the velocity and angular velocity of a body in simultaneous collisions are linear in the impulses at its contact points (with other bodies) like in (5)–(6) or (16)–(17). The linearity carries over to an object of arbitrary shape with angular inertia matrix Q. Suppose the forces f_i are applied on the object at the locations r_i. Integrating the dynamic equation $\sum_i r_i \times f_i = Q\dot{\omega} + \omega \times Q\omega$ over the duration Δt of an impact, we obtain $\sum_i r_i \times I_i = Q\Delta\omega$ since ω is bounded.

The state transition diagram can deal with three or more impact points via state partitioning based on which impacts are instantaneously "active" and which are not. A transition happens whenever the set of active impacts changes. The evolution within a state is driven by the primary impulse. The elastic energy at a contact can still be expressed in terms of the impulses affecting the two involved bodies as in (8)–(9). A differential relationship like (21) holds between the active normal impulses at two contact points.

Fig. 8. A mechanical cue stick.

We would like to compare the model with other existing models [3, 8, 5] on multiple impacts. Our main effort, though,

will be experimental verification of the impact model for billiard shots. A shooting mechanism has been designed as shown in Fig. 8. It includes a steel cue stick constrained to linear motions by ball bearings inside an aluminum box. The cue stick can be elevated by adjusting the slope of the attached incline. We plan to examine issues like area contact, shearing effect of the cue tip, bending of the cue stick, gravity, etc.

The long term objective is to design a robot able to play billiards with human-level skills based on understanding of the mechanics. To our knowledge, none of the developed systems [13, 10, 7] perform shots based on the mechanics of billiards, or have exhibited real shooting skills.

Acknowledgment. This work began during the first author's 6-month sabbatical visit at Carnegie Mellon University in 2007, and has since continued primarily at Iowa State University and in China. The work was sponsored in part by both universities, and in part by DARPA under contract HR0011-07-1-0002. This work does not necessarily reflect the position or the policy of the U.S. Government. No official endorsement should be inferred. The authors are grateful to Amir Degani and Ben Brown for their generous help in the design of the billiard shooting mechanism, and to the anonymous reviewers for their valuable comments.

References

1. Ahmed, S., Lankarani, H.M., Pereira, M.F.O.S.: Frictional impact analysis in open-loop multibody mechanical systems. J. Applied Mechanics 121, 119–126 (1999)
2. Anitescu, M., Porta, F.A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. ASME J. Nonlinear Dynamics 14, 231–247 (1997)
3. Baraff, D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. Computer Graphics 23, 223–232 (1989)
4. Brach, R.M.: Rigid body collisions. J. Applied Mechanics 56, 133–137 (1989)
5. Chatterjee, A., Ruina, A.: A new algebraic rigid-body collision law based on impulse space considerations. J. Applied Mechanics 65, 939–951 (1998)
6. Han, I., Gilmore, B.J.: Impact analysis for multiple-body systems with friction and sliding contact. In: Sathyadev, D.P. (ed.) Flexible Assembly Systems, pp. 99–108. American Society Mech. Engineers Design Engr. Div. (1989)
7. Ho, K.H.L., Martin, T., Baldwin, J.: Snooker robot player — 20 years on. In: Proc. IEEE Symp. Comp. Intell. Games, pp. 1–8 (2007)
8. Ivanov, A.P.: On multiple impact. J. Applied Math. Mechanics 59, 887–902 (1995)
9. Keller, J.B.: Impact with friction. J. Applied Mechanics 53, 1–4 (1986)
10. Long, F., et al.: Robotic pool: an experiment in automatic potting. In: Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 2520–2525 (2004)
11. Marlow, W.C.: The Physics of Pocket Billiards. Marlow Advanced Systems Technologies (1994)
12. Mason, M.T.: Mechanics of Robotic Manipulation. The MIT Press, Cambridge (2001)
13. Moore, A.W., Hill, D.J., Johnson, M.P.: An empirical investigation of brute force to choose features, smoothers and function approximators. In: Hanson, S.J., et al. (eds.) Computational Learning Theory and Natural Learning, pp. 361–379. The MIT Press, Cambridge (1995)

14. Cross, R.: Billiarads, http://physics.usyd.edu.au/~cross/Billiards.htm
15. Routh, E.J.: Dynamics of a System of Rigid Bodies. MacMillan and Co., Basingstoke (1913)
16. 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 Engr. 39, 2673–2691 (1996)
17. Stewart, D.E.: Rigid-body dynamics with friction and impact. SIAM Review 42, 3–39 (2000)
18. Wang, Y., Mason, M.T.: Two-dimensional rigid-body collisions with friction. J. Applied Mechanics 59, 635–642 (1991)

A Variational Approach to Strand-Based Modeling of the Human Hand

Elliot R. Johnson, Karen Morris, and Todd D. Murphey

Abstract. This paper presents a numerical modeling technique for dynamically modeling a human hand. We use a strand-based method of modeling the muscles. Our technique represents a compromise between capturing the full dynamics of the tissue mechanics and the need for computationally efficient representations for control design and multiple simulations appropriate for statistical planning tools of the hand. We show how to derive a strand-based model in a variational integrator context. Variational integrators are particularly well-suited to resolving closed-kinematic chains, making them appropriate for hand modeling. We demonstrate the technique first with a detailed exposition of modeling an index finger, and then extend the model to a full hand with 19 rigid bodies and 23 muscle strands. We end with a discussion of future work, including the need for impact handling, surface friction representations, and system identification.

1 Introduction

Dynamic models of the hand are difficult to create because of strong coupling between every part of the system. Coupling arises from the geometry of the muscles and tendons (e.g. a single tendon affects multiple joints rather than just one), the physical structure (e.g. coupling between muscles), and neurological factors. Physiologists typically reduce the complexity by focusing on a single finger or joint.

Elliot R. Johnson
Department of Mechanical Engineering, Northwestern University
e-mail: elliot.r.johnson@u.northwestern.edu

Karen Morris
Northwestern University Interdepartmental Neuroscience program, Northwestern University
e-mail: k-morris@u.northwestern.edu

Todd D. Murphey
Department of Mechanical Engineering, Northwestern University
e-mail: t-murphey@northwestern.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 151–166.
springerlink.com

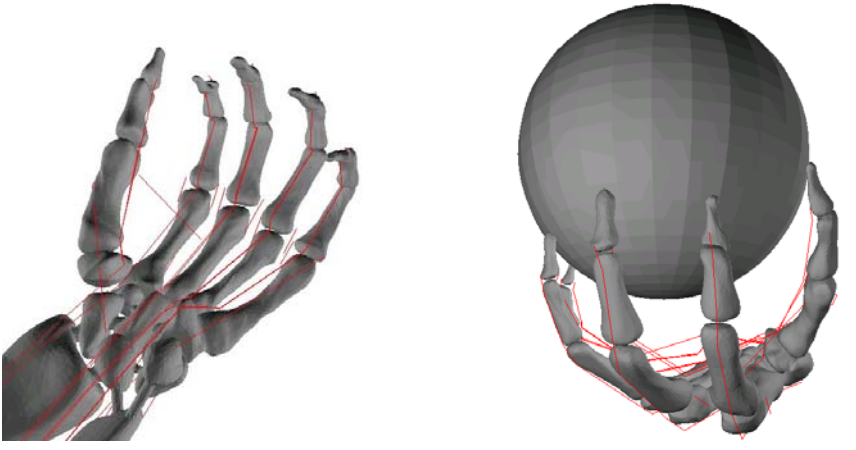


Fig. 1. The dynamic model of a human hand. (Left) The thin lines represent the muscle/tendon strands that actuate the hand. (Right) Contact can be modeled by holonomic constraints between finger tips and an object. (*The STL model was derived from http://www-static.cc.gatech.edu/projects/large_models/hand.html*)

Roboticists often simplify the hand's complicated tendon geometry into models that apply torque directly at each joint. While appropriate for *engineered* systems (e.g. the RIC hand [1]–but not the Shadow Hand [9] which is cable-driven), this approach discards the coupling and geometry that are crucial for studying the mechanical capabilities and control strategies of the hand.

Studies of complete hands in physiology have mostly been restricted to static models because of complexity. Static models are useful for applications such as predicting fatigue and maximum finger-tip forces in equilibrium, but they are fundamentally limited. The hand activates different muscles and activation patterns in dynamic motions compared to static contractions, even when moving through identical postures [22]. Dynamic models are clearly an important yet underdeveloped research area.

This paper presents a dynamic model of a complete hand that is free of numerical dissipation. The muscle/tendon pairs are modeled as one degree of freedom (DOF) elastic elements (e.g. linear or nonlinear springs) called *strands* [2, 20]. The model, including muscle strands, is shown in Fig. 1. Modeling is accomplished with a tree-based [7] variational integrator [21]. The tree approach provides a consistent framework for describing a system's geometry and including elements like forces, constraints, and potential energies. Variational integrators have excellent numeric stability and energy conservation properties, and handle closed kinematic chains extremely well. They also behave well with both elastic and plastic impacts as well as friction. Together, variational integrators and tree-form representations provide stable, physically accurate simulations in generalized coordinates even for mechanically complex systems like the hand.

Variational integrators should be of particular interest to the computer science part of the robotics community. They represent dynamics in a naturally discrete setting–hence the term *discrete mechanics*–rather than a continuous one. This feature, along with the fact that variational integrators are valid over long time horizons, makes probabilistic planning and optimal control more feasible. Moreover, because variational integrators are particularly simple to implement they can be integrated easily into other algorithms. Lastly, algorithms capable of computing variational integrators typically can compute other quantities such as the linearization; this allows one to evaluate the local singular value decomposition of a system at any operating point. The key point is that variational integrators provide a particularly computational view of mechanics, starting from their derivation, and this is useful when working with complex systems in a robotics context.

This work demonstrates that a variational modeling approach leads to numerically stable models that are easy to modify and extend. The model includes dynamics without resorting to PDEs, favoring useful abstractions and computational tractability instead. Variational integrators also deal with constraints (particularly closed kinematic chains) in a natural and robust way. Hence, these techniques can also be used on simpler systems such as the Shadow Hand [9] and other cable-driven robots.

We begin with a background discussion of hand anatomy and discuss previously published models in Sec. 2. We continue with an overview of variational integrators and discrete mechanics in Sec. 3. Section 4 discusses the strand abstraction our model uses for muscles/tendons in the hand. Finally, we present several simulations in Sec. 5 that use our model.

2 Background: Hand Modeling

Hand physiology is a complex subject that involves understanding the mechanical structure (i.e. system identification), characterizing the nervous system's control strategies, and developing mechanical models of the hand. We present a brief overview of hand anatomy and modeling strategies.

2.1 Hand Anatomy

The human hand contains 27 bones [6] with approximately 30 degrees of freedom [12]. It is actuated by 29 muscles, some of which are subdivided into parts that contract independently to provide a total of 38 unique actuators. Modeling the hand involves complex geometry due to the joints of the skeleton, sliding of the muscles, and the routing of the tendons.

We focus on the hand from the wrist to the finger tips. Each of the four digits is associated with four bones: the metacarpal (MCP), proximal phalanx (PP), middle phalanx (MP), and distal phalanx (DP). There are approximately five DOF for each digit: one at the base of the MCP, two at the MCP/PP joint, one at the PP/MP joint, and one at MP/DP joint.

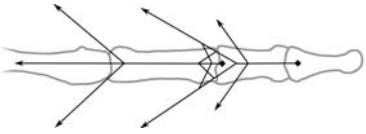


Fig. 2. The dark lines are a schematic representation of an extensor tendon. The single tendon attaches to several bones and muscles. Diamonds indicate fixed insertions to bones. Arrows represent muscle forces. The complex geometry results in strong coupling between muscles and non-trivial control.

The thumb has three bones (MCP, PP, and DP) and approximately four DOF allowing it to oppose, abduct, adduct, and flex [5]. The muscles actuating the thumb have complex geometry compared to those for the digits. As a result, the thumb is often not modeled despite contributing 40% of the function of the hand [8]. The abstractions used in this paper scale well with this type of complexity, and so the thumb *is* included in our model.

Bone geometry plays a significant role in hand dynamics. Tendons slide along bone surfaces, changing where forces are applied as the bones move. Because of the role that bone geometry plays, rapid prototyping is essential for generating meaningful hand models; physiologists often have insight about how a muscle/tendon moves across a bone during hand motion.

One of the hardest aspects of hand models are the tendons. Tendons are made up of dense connective tissue that is elastic, flexible, and strong [18, 5]. We will divide types of tendon into three classes for our purposes.

The first class we will define is made of simple tendons that are short connectors between a muscle and a single bone. The tendons that connect muscles to their origins are typically this simple type.

The second class has slightly more complicated tendons that connect one muscle to two bones, but work over long pathways. These tendons may slide over many bones, joints and fibrous sheaths that act as pulleys. As a result, the tension in the tendon applies forces to multiple points along its path and creates coupling between joints [19]. The flexor tendons of the hand are examples of this class.

Finally, the most complex tendons connect multiple bones and muscles, and also work over long pathways. These tendons can branch off and connect in many places to form complex structures. The extensor tendons, shown in Fig. 2 are this type. Identifying accurate representations of these tendons is still an open research problem in physiology [23].

Dynamic modeling of the hand is complicated by other physiological factors as well. The hand is over-actuated [6] and kinematically redundant [4] so there is no one-to-one mapping between muscles and joint torques. There is also significant coupling between muscles caused by both mechanical coupling and activation of more than one muscle at a time. This is again distinct from traditional robotic hands where all joints are independently controlled. Coupling plays an important role in all aspects of hand motion. Muscles are often co-activated such that some provide the major force while others stabilize the motion.

2.2 Hand Modeling

Partial differential equations have been used for modeling walking [15], facial movements [17], the surface of the heart [10], etc. However, there is no intrinsic reason to believe that a model that simplifies the hand system to a series of one degree-of-freedom-spring kinematic chains would be any less accurate than a highly complex PDE simulation [15]. This is because PDE models necessarily make assumptions about the underlying homogeneity of the muscle and bone tissue, leading to models that cannot be identified well. Moreover, for control purposes we wish to have a "simple" model that incorporates all the hand geometry and coupling in a dynamically correct manner.

Assuming one accepts a finite dimensional modeling setting, a finger can be modeled as a kinematic chain [13]. Lee and Kroemer [11] created a kinematic model of the finger that included flexion and extension. The moment arms of the tendons are constant and external forces were considered. This model was used to measure finger strength. Static equilibrium problems were used to make inferences about the dynamics of the model. Notably, there has been relatively little work on whole hand modeling.

Muscle models in the hand are often modeled as weightless expandable threads [2]. Models in the past that use the weightless expandable threads in hand modeling do so by solving static problems at each step and then animate the steps to create a smooth motions of the hand (see Sueda et al. [20]). The use of static poses and animation is effective at producing human like movement, but may not be natural. The model presented in this paper also uses the weightless expandable thread technique to model the tendons, but has the advantage of having no numerical dissipation (the major drawback described in [20]). Lastly, kinematic redundancy can be dealt with by adding constraints that reflect the physiology of the hand. These constraints can be easily included in our numerical simulation because variational integrators (described in the next section) are particularly well-suited to modeling closed kinematic chains.

3 Background: Variational Integrators

Variational integrators are a result of relatively recent research in discrete mechanics. These integrators are derived in a similar way as the Euler-Lagrange equations. They have been shown to respect important mechanical properties like conservation of energy and momentum (in the absence of nonconservative forcing), and have been observed to have other desirable properties like good dissipation modeling for systems with friction, excellent closed-kinematic chain behavior, and good convergence. Variational integrators also work directly in generalized coordinates which are preferred for describing anatomical aspects of the hand.

We introduce variational integrators with an overview of how the Euler-Lagrange equation is derived from a variational principle and discuss how the derivation is modified to obtain variational integrators.

Lagrangian mechanics provide a coordinate-invariant method for generating a system's dynamic equations. Lagrangian mechanics are derived from a variational principle. We define the Lagrangian of system as its total kinetic energy minus potential energy in terms of generalized coordinates $q = [q_1, q_2, \ldots, q_n] \in \mathbb{R}^n$ and their time derivative $\dot{q} = \left[\frac{\partial q_1}{\partial t}, \frac{\partial q_2}{\partial t}, \ldots, \frac{\partial q_n}{\partial t}\right]$:

$$L(q, \dot{q}) = KE(q, \dot{q}) - PE(q) \tag{1}$$

The integral of the Lagrangian over a trajectory is called the Action (Fig. 3a):

$$S(q([t_0, t_f])) = \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau)) \mathrm{d}\tau \tag{2}$$

Hamilton's Least Action Principle states that a mechanical system will naturally follow the trajectory that extremizes (e.g. minimizes) the action; hence, it is not so much a least action principle as an action stationarity principle. Extremizing (2) with a variational principle shows that such trajectories satisfy the Euler-Lagrange equation:

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}}(q, \dot{q}) - \frac{\partial L}{\partial q}(q, \dot{q}) = 0 \tag{3}$$

which is a second-order ordinary differential equation (ODE) in q. Given a set of initial condition $(q(t_0), \dot{q}(t_0))$, we numerically integrate (3) to simulate the system dynamics. The derivation can be extended to include holonomic/non-holonomic constraints, external forces, and dissipation [14].

In derivation of (3), the system's trajectory is always continuous. The trajectory is not discretized until the last step during numeric integration. A variational integrator, on the other hand, is derived by introducing the time discretization before applying the variational principle.

3.1 Discrete Mechanics

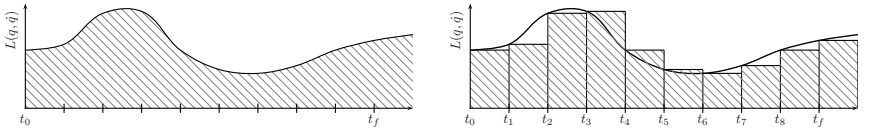
In discrete mechanics, we seek a sequence $\{(t_0, q_0), (t_1, q_1), \ldots, (t_n, q_n)\}$ that approximates the actual trajectory of a mechanical system ($q_k \approx q(t_k)$). In this paper, we assume a constant time-step ($t_{k+1} - t_k = \Delta t \ \forall \ k$) for simplicity, but in general, the time-step can be varied to use adaptive time-stepping algorithms.

A variational integrator is derived by defining a discrete Lagrangian, L_d, that approximates the continuous action integral over a short time interval.

$$L_d(q_k, q_{k+1}) \approx \int_{t_k}^{t_{k+1}} L(q(\tau), \dot{q}(\tau)) \mathrm{d}\tau \tag{4}$$

The discrete Lagrangian allows us to replace the system's action integral with an approximating action sum.

$$S(q([t_0, t_f])) = \int_{t_0}^{t_f} L(q(\tau), \dot{q}(\tau)) \mathrm{d}\tau \approx \sum_{k=0}^{n-1} L_d(q_k, q_{k+1}) \tag{5}$$



(a) Continuous Action Integral (b) Discrete Action Sum

Fig. 3. The discrete action sum approximates the continuous action integral using the discrete Lagrangian (which requires a choice of quadrature rule, in this case the standard Riemann integral).

where $t_f = t_n$. This approximation is illustrated in Fig. 3. The shaded region in Fig. 3a represents the continuous action integral. The shaded boxes in Fig. 3b represent values of the discrete Lagrangian, which are summed to calculate the discrete action.

In continuous mechanics, a variational principle is applied to extremize the action integral and derive the well-known Euler-Lagrange equation. The same approach is used to extremize (5) to get the discrete Euler-Lagrange (DEL) equation[1].

$$D_1 L_d(q_k, q_{k+1}) + D_2 L_d(q_{k-1}, q_k) = 0 \tag{6}$$

$$h(q) = 0 \tag{7}$$

where Eq. (7) represents constraints if there are any. Nonholonomic constraints can be represented as well, but for simplicity we do not discuss them here.

This is an implicit difference equation that depends on the previous, current, and future states. Given q_{k-1} and q_k, (6) is treated as a *root-finding* problem to find q_{k+1}. After advancing k, this process is repeated to simulate the system for as long as desired.

Constraints play a crucial role in simulations with contact and grasping. We can often represent mechanical contact with a holonomic constraint (i.e. a constraint on the system's configuration manifold). In continuous mechanics, holonomic constraints are typically differentiated and included as velocity/acceleration constraints. Over time, numeric integration errors build up and the trajectory violates the original holonomic constraint. In differential-algebraic techniques that enforce the constraint, the error is either accepted or corrected with heuristic methods that (as a side effect) artificially inject or dissipate energy. This is acceptable for computer entertainment applications, but leads to unrealistic system identification and unstable controllers for physical applications.

Variational integrators avoid this problem and implement holonomic constraints well as a result. At each time step, the constraints $h(q)$ are satisfied (to within the tolerance of the numeric root solver) by appending them to (6) and including a constraint forcing term in the discrete Euler-Lagrange equation. The constraint resolution is also directly coupled to the dynamics via the discrete Lagrange D'Alembert principle rather than being a heuristic fix as in the continuous case.

[1] $D_n f(\dots)$ is the derivative of $f(\dots)$ with respect to its n-th argument. This is sometimes called the *slot derivative.*

3.2 Complexity

A common problem with Euler-Lagrange simulations is that the equations grow too quickly as the system becomes larger (or more complex). Most simulation methods are therefore based on force balance methods and avoid generalized coordinates. Our recent work [7] discusses a new approach to Lagrangian simulations that significantly reduces the complexity growth and keeps generalized coordinates feasible for much larger systems. This approach ensures through the use of caching in a tree-structure that every transformation is only computed once, trivially leading to $O(n)$ computation of Eq. (6) for unconstrained systems (for constrained system one gets $O(n+m)$ to compute Eqs. (6) and (7) with m constraints). However, most root-solving techniques require differentiating Eqs. (6) and (7) so that Newton's Method can be applied. This entails inverting an $n \times n$ matrix, which in many cases is also an $O(n)$ calculation so long as one is careful to take advantage of the group structure.

It is worth noting here that the computational complexity of computing $f(\cdot)$ in $\dot{x} = f(x,u)$ is not always the relevant notion of complexity; it assumes that we are only interested in the complexity of evaluating a single step of an integrator. Rather, we are typically interested in knowing the complexity of obtaining a solution that is within some error of the "true" solution to the equations of motion. We have an example in [7] of a system–the scissor lift–that scales linearly in terms of computing $f(\cdot)$ but scales *exponentially* in terms of computing the correct solution. This point should not be taken lightly–it implies that one of the main metrics we use for evaluating simulation techniques is often times off-point. For that example, variational integrators are substantially more efficient at computing the correct solution even though they are less efficient at computing Eqs. (6) and (7).

We have implemented these ideas in a freely available, open-source package called `trep`[2]. `trep` is designed as an easy to use tool for rapid, incremental development of simulations without sacrificing performance. It provides useful facilities like compact representations of systems and automatic visualization. Most importantly, `trep` is easy to extend with new types of potentials, forces, and constraints. This allows us, for example, to quickly explore different muscle/tendon representations in a common, structured environment. The ability to quickly adapt a simulation is critical to a low-dimensional model's success.

4 Strand Models of the Hand

We now move on to the use of strand models in a variational context. As previously mentioned, a simplified but useful model in physiology is to consider muscle/tendon groups as linear springs. A spring constant, k, is chosen to reflect the bulk elasticity of the the tendon and muscle tissue. The muscle is contracted and relaxed by controlling the natural length, x_0, of the spring.

For the spring model to be useful for hand models, we must extend it to handle the routing and sliding needed for extrinsic muscle tendons. We can think of a

[2] http://trep.sourceforge.net

muscle/tendon as a *strand* that connects two points and slides through intermediate points.

Formally, a strand is defined by a spring constant, $k \in \mathbb{R}$, a (possibly time-varying) natural length $x_0 \in \mathbb{R}$, and a set of points $p_1, p_2 \ldots p_N \in \mathbb{R}^3$ where $N \geq 2$. The current length of the strand is found by accumulating the linear distance between adjacent points:

$$\begin{aligned} x(q) &= \sum_{i=1}^{N-1} ||p_i(q) - p_{i+1}(q)|| \\ &= \sum_{i=1}^{N-1} \left[(p_i(q) - p_{i+1}(q))^T (p_i(q) - p_{i+1}(q)) \right]^{\frac{1}{2}} \end{aligned} \tag{8}$$

The potential energy of the strand is then:

$$V(q,t) = \tfrac{1}{2} k (x(q) - x_0(t))^2 . \tag{9}$$

As one would expect, one can replace a nonlinear potential with the quadratic one. `trep` also requires the derivative of (9) to implement the strand potential. These are straightforward to calculate by applying chain rule.

$$\frac{\partial V}{\partial q}(q,t) = k (x(q) - x_0(t)) \cdot \frac{\partial x}{\partial q}(q)$$

The derivative of $x(q)$ is found similarly.

$$\begin{aligned} \frac{\partial x}{\partial q_j}(q) = \sum_{i=1}^{N-1} \Bigg(& \tfrac{1}{2} \left[(p_i(q) - p_{i+1}(q))^T (p_i(q) - p_{i+1}(q)) \right]^{-\frac{1}{2}} \cdot \\ & \left[\left(\frac{\partial p_i}{\partial q_j}(q) - \frac{\partial p_{i+1}}{\partial q_j}(q) \right)^T (p_i(q) - p_{i+1}(q)) + \right. \\ & \left. (p_i(q) - p_{i+1}(q))^T \left(\frac{\partial p_i}{\partial q_j}(q) - \frac{\partial p_{i+1}}{\partial q_j}(q) \right) \right] \Bigg) \\ = \sum_{i=1}^{N-1} & \frac{(p_i(q) - p_{i+1}(q))^T \left(\frac{\partial p_i}{\partial q_j}(q) - \frac{\partial p_{i+1}}{\partial q_j}(q) \right)}{||p_i(q) - p_{i+1}(q)||} \end{aligned}$$

There are a number of ways this model can be improved. The most simple improvement is to use non-linear potentials instead of a linear model. Once a better potential shape is identified experimentally, it can be used by updating (9) and (4). The strands should also be extended to include branching and sliding so that complex tendons like the digit extensors can be modeled correctly. However, this will require system identification to determine how the topology should be defined; real tendons do not join at unique locations and are instead defined by large, somewhat amorphous regions of connection.

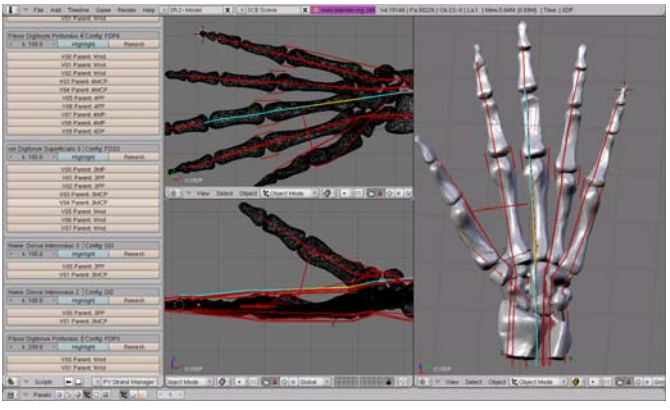


Fig. 4. A graphical user interface (GUI) makes it easy to integrate new muscle strands in the model.

5 Model Implementation

The strand model was implemented as a new potential type in `trep`. We created a 2D finger model (Sec. 5.1) and a full 3D hand model (Sec. 5.2). A third model is presented by adding a sphere and holonomic constraints to simulate grasping contact. Hand dimensions were based on the STL model in Fig. 1 and the spring constants were chosen to be rather stiff (100-300 N/m) to represent the stiffness of the tendons. However, careful system identification using the whole model should be done in the future (along the lines of [16]) to properly calculate the model parameters.

Simulations were developed using Blender[3] as a graphical user interface (GUI). Figure 4 shows a custom plug-in provides convenient ways to define and modify new strands. Blender was also used to define the desired trajectories and poses. The combination of `trep` and Blender makes extending and improving the model easy.

The sources for the hand model can be downloaded from `trep`'s website at http://trep.sourceforge.net/examples/hand.html.

5.1 A Finger

A two-dimensional finger is simple compared to a full hand, but also easier to understand. The model is shown in Fig. 5. We have simplified the extensor tendon into a simple tendon with one muscle. The model has three degrees of freedom and four muscle strands.

The simulation was created by defining a desired trajectory and computing the corresponding natural lengths for each strand. The dynamic system was then simulated using these lengths as inputs. The result is a rudimentary control scheme that does not incorporate feedback. It is intended to demonstrate the dynamic model rather than accurately simulate hand control.

[3] http://www.blender.org

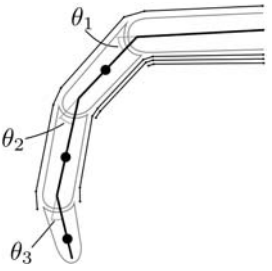


Fig. 5. The 2D finger model has three degrees of freedom and four muscles/tendon strands.

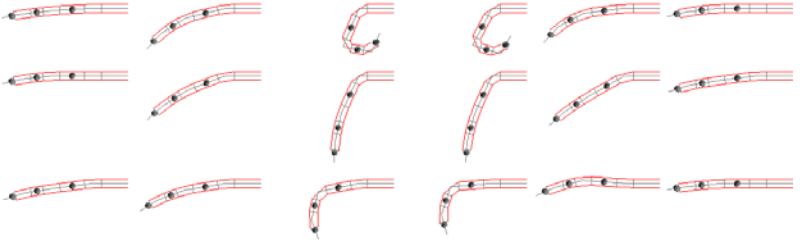


Fig. 6. The finger was moved through several motions. The strand model was capable of actuating the finger to follow the trajectory.

The model was simulated in `trep` for 30.0s with a time step of 0.01s. The simulation took approximately 34s to compute on a 2.2GHz Intel Core2 Duo processor. The results are shown in Fig. 6.

The trajectories for the three joint angles are plotted in Fig. 7. For each of the three motions, the joint angles tend to move in the same direction as a result of the *coupling* introduced by the tendons.

5.2 Full Hand

We extended the finger to a full hand model. The goal for this model is to demonstrate that the fingers can be individually actuated when there is no coupling between muscle groups; coupling and co-activation can always be added to this model without difficulty because once the tendon topology is known these simply add stress/strain relationships. Our model has 20 degrees of freedom along with 23 independently actuated strands. Fig. 8 illustrates the model.

The most significant simplification is that the extensor tendons have been divided into several different muscle/tendons.

The model was tested using a hand closure trajectory. The hand begins with all digits extended. The digits are flexed inwards toward the palm and then return to their original extended configurations. The simulation is defined and run in the same manner as the above finger simulation.

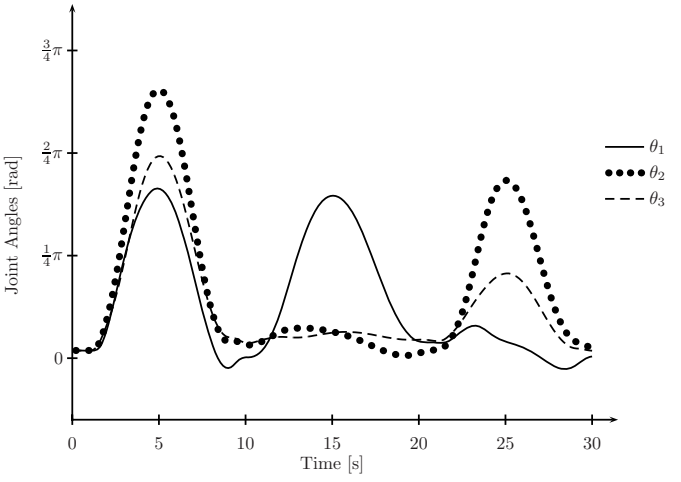
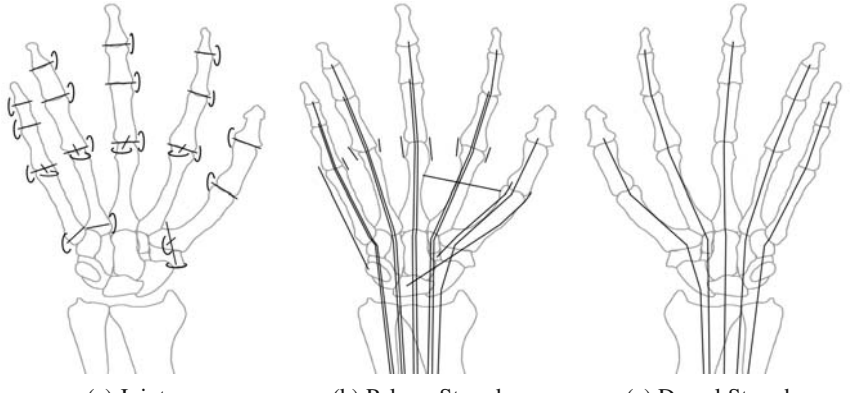


Fig. 7. Joint angles vs. time for a finger simulation. Note how all three angles tend to move in the same direction whenever there is movement because of coupling between joint angles.



(a) Joints (b) Palmar Strands (c) Dorsal Strands

Fig. 8. The complete hand model including coupling between fingers.

The simulation lasts for 10 seconds and uses a time step of 0.001s. A 2.2GHz Intel Core2 Duo Processor completed the simulation in approximately one hour. Several stages of the simulation are shown in Fig. 9.

The simulation is slow (compared to real-time) because of the small time steps that are required by large spring constants. However, the corresponding high-frequency oscillations are almost completely absent in the trajectory. This suggests that a stiff elastic tendon model may be inappropriate for grasping/large-movement simulations. A better model might take muscle tension as input and use the strand geometry to determine the resulting forces on the hand. While this has been deferred for future work, the change is straightforward in `trep`.

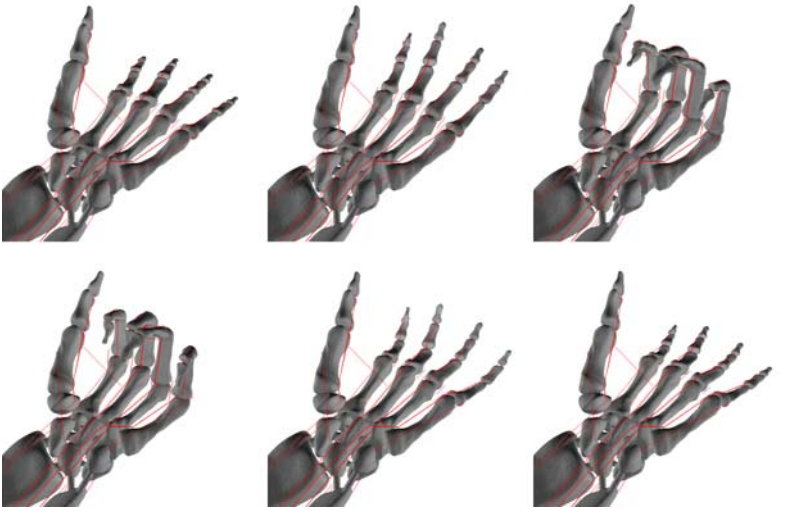


Fig. 9. This figure shows several stages of the simulation results. (Time advances left to right, top to bottom)

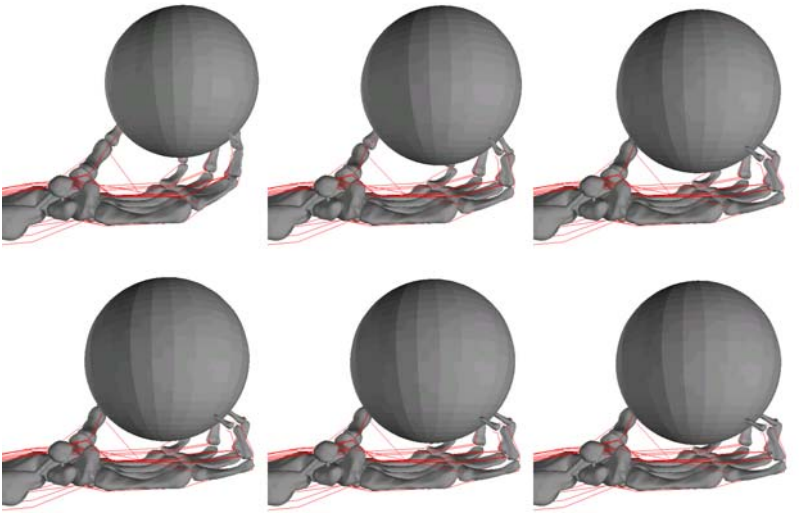


Fig. 10. This figure shows several stages of the simulation results. The simulation exhibits the expected behavior of the ball settling down as the tendons are stretched. (Time advances left to right, top to bottom)

5.3 Grasping Simulation

Finally, we demonstrate the extendibility of the model with a grasping simulation. A sphere was added and brought into contact with the five finger tips. Holonomic

constraints attach the finger tips to the sphere's surface, so we are modeling assuming "infinite" friction between the finger tip surface and the sphere surface. For this simulation, the natural length of each strand was fixed and gravity was added. The resulting trajectory is due entirely to the dynamic interaction between the sphere and hand.

The simulation ran for 10 seconds with a time step of 0.001s. The simulation was completed in approximately one and half hours on a 2.2GHz Intel Core2 Duo processor. Several stages of the simulation are shown in Fig. 10.

Again, the length of time for the simulation could be dramatically reduced by changing the dynamic model of the strands to avoid the high spring constants and the associated high frequency vibrations that occur in the strands. This is a focus of future work, but our preliminary work in this area suggests that in addition to using strand tension as an input, treating the strands as kinematic variables (while leaving the rest of the hand dynamic) results in between one and two orders of magnitude faster calculation.

6 Conclusions and Future Work

We have developed a strand-based model of the hand that simulates the complete hand. The model is based on variational integrators which provide excellent behavior with constraints, coupling, and closed kinematic chains and fixes the numerical dissipation issues that the hand model in [20] exhibits. The variational integrators also simulate the system directly in generalized coordinates.

This work represents only the first steps towards accurate hand models for manipulation tasks. The next step in modeling is to design more advanced tendon models that are capable of branching, sliding, and becoming slack.

The simulation environment also needs to be extended to handle elastic impacts, plastic impacts, and nonholonomic constraints. Relevant theoretical work has already been done [3] and variational integrators are known to handle these well. `trep` is currently being improved in this direction.

Better models of friction, including stick/slip phenomenon are also needed. This is an active research area in the dynamics community. The progress there is expected to work well in the variational integrator setting.

A hand model can only be as accurate as the parameters used to design it. System identification experiments are needed to collect empirical data and characterize the hand [16]. This includes measuring properties like elasticity or damping, and improving abstract representations like spring networks for the extensor tendons.

Lastly, this paper does not address control for manipulation. Future work in this direction includes automating the calculation of Jacobians for torque/force calculation and linearization of the dynamics for feedback control design (including optimal control).

References

1. Baker, B.: The magic touch: Scientist invent the cyberhand, a brain controlled robotic hand with fingers that can actually feel (2006), http://www.popsci.com/article/2006-03/magic-touch
2. Biryukova, E., Yourovskaya, V.: A model of hand dynamics. In: Schuind, F., An, K., Cooney, W., Elias, M.G. (eds.) Advances in the Biomechanics of Hand and Wrist, pp. 107–122. Plenum Press (1994)
3. Fetecau, R.C., Marsden, J.E., Ortiz, M., West, M.: Nonsmooth Lagrangian mechanics and variational collision integrators. SIAM Journal on Applied Dynamical Systems (2003)
4. Flanagan, J.R., Haggard, P., Wing, A.M.: The task at hand. In: Wing, A.M., Haggard, P., Flanagan, J.R. (eds.) Hand and Brain, pp. 5–13. Academic Press, Inc., London (1996)
5. Henry Gray, F.R.S.: The Complete Gray's Anatomy. Merchant Book Company, Finland (2003)
6. Hepp-Reymond, M.-C., Huesler, J., Maier, M.A.: Precision grip in humans: temporal and spatial synergies. In: Wing, A.M., Haggard, P., Flanagan, J.R. (eds.) Hand and Brain, pp. 33–68. Academic Press, Inc., London (1996)
7. Johnson, E.R., Murphey, T.D.: Scalable variational integrators for constrained mechanical systems in generalized coordinates. IEEE Transactions on Robotics (2008) (submitted)
8. Jones, L., Lederman, S.: Human Hand Function. Oxford University Press, New York (2006)
9. Kochan, A.: Shadow delivers first hand. Industrial Robot: An International Journal 32(1), 15–16 (2005)
10. Kounchev, O., Wilson, M.J.: Mathematics of Surfaces. Springer, Heidelberg (2003)
11. Lee, K.-H., Kroemer, K.H.E.: A finger model with constant tendon moment arms. In: Proceedings of the Human Factors and Ergonomics Society, pp. 710–714 (2007)
12. Lin, J., Wu, Y., Huang, T.S.: Modeling the constraints of human hand motions. In: Human Motion, 2000. Proceedings, pp. 121–126. IEEE, Los Alamitos (2000)
13. Linscheid, R.L.: Historical perspective of finger joint motion: the hand-me-downs of our predecessors. Journal of Hand Surgery 27, A:1–A:25 (2002)
14. Marsen, J.E., West, M.: Discrete mechanics and variational integrators. Acta Numerica, 357–514 (2001)
15. Pandy, M.G.: Computer modeling and simulation of human movement. Annual Review of Biomedical Engineering 3, 245–273 (2001)
16. Pearlman, J., Roach, S., Valero-Cuevas, F.: The fundamental thumb-tip force vectors produced by the muscles of the thumb. Journal of Orthopaedic Research 22(2), 306–312 (2006)
17. Sancho-Bru, J.L., Perèz-Gonzalez, A., Vergara-Monedero, M., Giurintano, D.: A 3-D dynamic model of human finger for studying free movements. Journal of Biomechanics 34, 1491–1500 (2001)
18. Scheepers, F., Parent, R.E., Carlson, W.E., May, S.F.: Anatomy-based model of the human musculature. In: International Conference on Computer Graphics and Interactive Techniques, Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, pp. 163–172 (1997)
19. Schieber, M.H.: Individuated finger movements: rejecting the Labeled-Line Hypothesis. In: Wing, A.M., Haggard, P., Flanagan, J.R. (eds.) Hand and Brain, pp. 81–98. Academic Press, Inc., London (1996)

20. Sueda, S., Kaufman, A., Pai, D.K.: Musculotendon simulation for hand animation. In: ACM SIGGRAPH conference proceedings, vol. 27 (2008)
21. West, M.: Variational integrators. California Institute of Technology Thesis (2004)
22. Wing, A.M.: Anticipatory control of grip force in rapid arm movement. In: Wing, A.M., Haggard, P., Flanagan, J.R. (eds.) Hand and Brain, pp. 33–68. Academic Press, Inc., London (1996)
23. Zilber, S., Oberlin, C.: Anatomical variations of the extensor tendons to the fingers over the dorsum of the hand: a study of 50 hands and a review of the literature. Plastic Reconstructive Surgery 113, 214–221 (2004)

A Stopping Algorithm for Mechanical Systems

Jason Nightingale, Richard Hind, and Bill Goodwine

Abstract. Analysis and control of underactuated mechanical systems in the nonzero velocity setting remains a challenging problem. In this paper, we demonstrate the utility of a recently developed alternative representation of the equations of motion for this large class of nonlinear control systems. The alternative representation gives rise to an intrinisic symmetric form. The generalized eigenvalues and eigenvectors associated with the symmetric form are used to determine control inputs that will drive a class of mechanical systems underactuated by one control to rest from an arbitrary initial configuration and velocity. Finally, we illustrate the stopping algorithm by presenting numerical simulation results for the planar rigid body, snakeboard and planar rollerblader.

1 Introduction

Mechanical control systems form a large and important subclass of nonlinear control systems. The areas of application of control theory for mechanical systems are diverse and challenging. Such areas include autonomous aerospace and marine vehicles, robotics and automation, mobile robots, and constrained systems. The formalism of affine connections and distributions on a Riemannian manifold provides an elegant framework for modeling, analysis and control. This framework has given rise to new insights into nonlinear controllability in the *zero velocity setting* motivating stabilization, tracking and motion planning algorithms [1]. For fully actuated mechanical systems, it is possible to provide a comprehensive solution to the problem of trajectory tracking [10]. In contrast, motion planning algorithm for underactuated mechanical systems is still not well understood. Due to the challenging nature

Jason Nightingale and Bill Goodwine
Department of Aerospace and Mechanical Engineering, University of Notre Dame
e-mail: jnightin@nd.edu, jgoodwin@nd.edu

Richard Hind
Department of Mathematics, University of Notre Dame
e-mail: rhind@nd.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 167–180.
springerlink.com

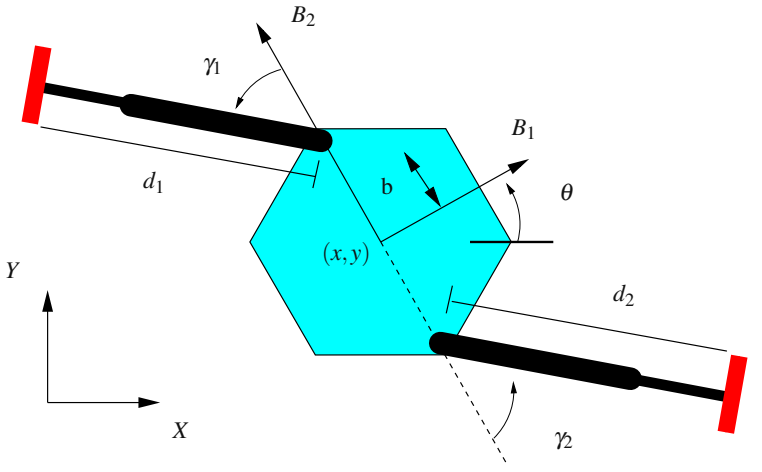


Fig. 1. A schematic of the planar rollerblader.

of these problems, many of the existing results have been limited for example to gait generation algorithms applicable only to the specific systems [9] [8], configuration to configuration algorithms with zero-velocity transitions between feasible motions for specific systems [3] and numerically generated optimal trajectories [4].

This paper is also closely related to several efforts that have been made to obtain conditions in the zero velocity setting from properties of a certain intrinsic vector-valued quadratic form which does not depend upon the choice of basis for the input distribution [2], [6]. It has been observed that vector-valued quadratic forms come up in a variety of areas in control theory which has motivated a new initiative to understand the geometry of these forms [7].

1.1 Motivating Example

As a concrete example, take the planar rollerblader illustrated in Figure 1. The schematic drawing illustrates the kinematics and actuator locations of the model. Note that each leg is composed of two links which are connected by a translation joint at the knee and a pin joint at the hip. The foot is a roller blade which is constrained to the plane in such a way that prohibits motion of the foot perpendicular to the blade. A single actuator capable of generating torque in both the clockwise and counterclockwise directions is placed at each pin joint. Another set of linear actuators are placed at each translation joint. The planar rollerblader has five degrees of freedom and only four actuators. This is an example of an **underactuated** control system. Whenever fewer actuators are available than degrees of freedom, various control questions arise. For instances, it is not immediately clear whether the moving rollerblader can be "stopped" using the limited control authority. If it cannot be stopped, then the set of reachable velocities does not include zero velocity. *In this, and other underactuated mechanical systems, existing geometric control*

theory does not provide a general test for stopping and more generally speaking, the set of reachable velocities from a nonzero velocity is not well understood. The modern development of geometric control of mechanical systems has been limited, for the most part, to the zero velocity setting. Yet the underlying mathematical structure is that of second-order dynamics where the state of the system is defined by a configuration and velocity. Theoretical results that are limited to zero velocity states do not provide an adequate characterization of the behavior of mechanical systems and limits the development of motion planning algorithms.

1.2 Statement of Contribution

The fundamental approach of this paper is to exploit the inherent geometric structure for the purpose of stopping an underactuated mechanical system. We use the governing equations of motion of the mechanical control system to partition or foliate the velocity-phase manifold (*e.g.* set of all configurations and velocities). This partitioning has given rise to two key theoretical results which have been the main topics of recent works [13], [14]. First, we have identified an intrinsic vector-valued symmetric bilinear form that can be associated with an underactuated mechanical control system. Second, we have provided computable tests dependent upon the definiteness of the symmetric form to determine if the system can or cannot be driven to rest.

Our theoretical results are useful for two reasons. First, such results are necessary conditions for a stopping algorithm. If zero velocity is not contained in the set of reachable states then it is impossible to specify a control law that will drive the system to rest. Second, these results are useful design tools which provide constructive strategies for actuator assignment and help to make the control scheme robust to actuator failure. The task of actuator assignment is always a balance between the sophistication of the system design and the associated complexity of the controller. For example, a system which is fully actuated requires a simple control scheme to drive it to rest. In contrast, if the system is underactuated even by just one control, a control scheme must take into account the underlying geometry or nonlinearities of the geometric model. Such a control scheme is theoretically challenging due to nonzero drift which indicates a component of the dynamics that is not directly controlled. These systems are not amendable to standard techniques in control theory. However, underactuated systems do appear in many practical applications resulting from design choices motivated by cost reductions or in some cases they are the result of a failure in fully actuated mechanical systems.

The main contribution of this paper is a stopping algorithm for mechanical systems underactuated by one control. We focus our analysis on such systems whose symmetric form is indefinite for almost all configurations. The choice of control inputs are dependent upon the generalized eigenvalues and eigenvectors associated with the symmetric form. The stopping algorithm is applied to the planar rigid body, snakeboard [9] and rollerblader [8]. For each system, we provide the geometric model, our alternative representation and simulation results.

2 Geometric Model

2.1 Mechanical Control System

We consider a simple mechanical control system with no potential to be comprised of an n-dimensional configuration manifold M; a Riemannian metric $\mathbb{G}$ which represents the kinetic energy; m linearly independent one forms $F^1,\ldots,F^m$ on M which represents the input forces; a distribution H on M which represents the constraint; and $U=\mathbb{R}^m$ which represents the set of inputs. We do not require the set of inputs to be a subset of $\mathbb{R}^m$. This allows use to focus on the geometric properties of our system that inhibit or allow motion in the foliation as opposed to a limitation on the set of inputs. We represent the input forces as one forms and us the associated dual vector fields $Y_a=\mathbb{G}^\sharp(F^a)$, $a=1,\ldots,m$ in our computations. Formally, we denote the control system by the tuple $\Sigma=\{M,\mathbb{G},\mathscr{Y},V,U\}$ where $\mathscr{Y}=\{Y_a \mid Y_a=\mathbb{G}^\sharp(F^a)\ \forall\ a\}$ is the input distribution. Note we restrict our attention to control systems where the input forces are dependent upon configuration and independent of velocity and time. DoCarmo [12] provides an excellent introduction to Riemannian geometry. A thorough description of simple mechanical control systems is provided by Bullo and Lewis [1].

It is well known that the Lagrange-d'Alembert principle can be used to generate the equations of motion for a forced simple mechanical system in coordinate invariant form. If we set the Lagrangian equal to the kinetic energy, then the equations are given by

$$\nabla_{\dot\gamma(t)}\dot\gamma(t) = u^a(t)Y_a(\gamma(t)) \tag{1}$$

where ∇ is the Levi-Civita connection corresponding to $\mathbb{G}$, u is a map from $I\subset\mathbb{R}\mapsto\mathbb{R}^M$, $\gamma: I\to M$ is a curve on M and $t\in I$. Therefore, a *controlled trajectory* for Σ is taken to be the pair (γ,u) where γ and u are defined on the same interval $I\subset\mathbb{R}$. Note the usual summation notation will be assumed over repeated indices throughout this paper.

Given a constraint distribution H of rank k, we may restrict the Levi-Civita connection ∇ to H [11]. Bullo and Zefran [5] showed that given two vector fields X and Y on M then the so-called *constrained affine connection* $\tilde\nabla$ is given by

$$\tilde\nabla_X Y = P(\nabla_X Y)$$

where P is the orthogonal projection $TM\mapsto H$. The latter approach provides a computationally efficient method and is used to generate the coordinate expression for the constrained affine connection for the roller racer and the snakeboard.

The natural coordinates on TM are denoted by $((q^1,\ldots,q^n),\ (v^1,\ldots,v^n))$ where $(v^1,\ldots,v^n)$ are the coefficients of a tangent vector given the usual basis $\{\frac{\partial}{\partial q^1},\ldots,\frac{\partial}{\partial q^n}\}$. We will denote a point in TM by v_q. We may lift the second-order differential equation defined by (1) to TM. This gives rise to the following system of first-order differential equations on TM

$$\begin{aligned} \frac{dq^k}{dt} &= v^k, \\ \frac{dv^k}{dt} &= -\Gamma_{ij}^k v^i v^j + u^a Y_a^k, \end{aligned} \tag{2}$$

where Γ_{ij}^k is the usual Christoffel symbol and $i, j, k = 1, \ldots, n$.

A critical tool used to analyze distributions and mechanical control systems is the symmetric product. Given a pair of vector fields X, Y, their symmetric product is the vector field defined by

$$\langle X : Y \rangle = \nabla_X Y + \nabla_Y X.$$

2.2 Symmetric Bilinear Form

In this section we expand upon and adapt the definition of an affine subbundle found in Hirschorn and Lewis [6]. We restrict our attention to configuration manifolds that admit a well defined global set of basis vector fields however our results generalize under appropriate conditions. *The basic geometry of our construction can be captured by assuming $H = TM$ however we can always relax this assumption by properly accounting for the orthogonal projection P.*

Recall that an input distribution $\mathscr{Y}$ on M is a subset $\mathscr{Y} \subset TM$ having the property that for each $q \in M$ there exists a family of vector fields $\{Y_1, \ldots, Y_m\}$ on M so that for each $q \in M$ we have

$$\mathscr{Y}_q \equiv \mathscr{Y} \cap T_q M = \operatorname{span}_{\mathbb{R}}\{Y_1(q), \ldots, Y_m(q)\}.$$

We refer to the vector fields $\{Y_1, \ldots, Y_m\}$ as generators for $\mathscr{Y}$. Let $\mathscr{Y}^\perp$ denote an orthonormal frame $\{Y_1^\perp, \ldots, Y_{n-m}^\perp\}$ that generates the $\mathbb{G}$-orthogonal complement of the input distribution $\mathscr{Y}$. Note that even though $\mathscr{Y}^\perp$ is canonically defined, we must choose an orthonormal basis. It is clear that $\{\mathscr{Y}_q, \mathscr{Y}_q^\perp\}$ forms a basis for $T_q M$ at each $q \in M$. Note that $\mathscr{Y} = \{Y_1, \ldots, Y_m\}$ is a set of *m linearly independent* vector fields while $\mathscr{Y}^\perp = \{Y_1^\perp, \ldots, Y_{n-m}^\perp\}$ is a set of $n-m$ *orthonormal* vector fields. This basis will be used to define an affine subbundle and construct an affine foliation of the tangent bundle.

An *affine subbundle* on M is a subset $A \subset TM$ having the property that for each $q \in M$ there exists a family of vector fields $\{Y_0, \ldots, Y_m\}$ so that for each $q \in U$ we have

$$\begin{aligned} A_q &\equiv A \cap T_q M \\ &= \{Y_0(q) = Y_1^\perp(q) + \cdots + Y_{n-m}^\perp(q)\} \\ &\quad + \operatorname{span}_{\mathbb{R}}\{Y_1(q), \ldots, Y_m(q)\}. \end{aligned}$$

An *affine foliation*, $\mathscr{A}$, on TM is a collection of disjoint immersed affine subbundles of TM whose disjoint union equals TM. Each connected affine subbundle A is called

an *affine leaf* of the affine foliation. Now let us apply this framework to a simple mechanical control system.

Definition 2.1. Let $(M, \mathbb{G}, V, \mathscr{Y}, U)$ be a simple mechanical control system with the input distribution $\mathscr{Y}$ generated by $\{Y_1, \ldots, Y_m\}$ and the corresponding $\mathbb{G}$-orthogonal distribution $\mathscr{Y}^\perp$ generated by $\{Y_1^\perp, \ldots, Y_{n-m}^\perp\}$. An **input foliation** $\mathscr{A}_\mathscr{Y}$ is an affine foliation whose affine leaves are affine subbundles given by

$$A_s(q) = \{v_q \in TM \mid \langle\langle Y^\perp, v_q \rangle\rangle = s, s \in \mathbb{R}^{n-m}\}.$$

Remark 2.1. The input foliation is parametrized by $s \in \mathbb{R}^{n-m}$. Note that when $s = 0$, $A_0 = \mathscr{Y}$ and $A_0(q) = \mathscr{Y}_q$ where $\mathscr{Y}$ is an immersed submanifold of TM and $\mathscr{Y}_q$ is a linear subspace of T_qM. Thus, the input distribution $\mathscr{Y}$ is a single leaf of the affine foliation.

Given a basis of vector fields $\{X_1, \ldots, X_n\}$ on M, we define the *generalized Christoffel symbols* of ∇ to be

$$\nabla_{X_i} X_j = \hat{\Gamma}_{ij}^k X_k.$$

Note that when $X_i = \frac{\partial}{\partial q^i}$ we recover the usual Christoffel symbols of ∇. We introduce the *symmetrization* of the generalized Christoffel symbols.

Definition 2.2. We define the **generalized symmetric Christoffel symbols** for ∇ with respect to the basis of vector fields $\{X_1, \ldots, X_n\}$ on M as the n^3 functions $\tilde{\Gamma}_{ij}^k : M \to \mathbb{R}$ defined by

$$\begin{aligned} \tilde{\Gamma}_{ij}^k X_k &= \frac{1}{2}\left(\hat{\Gamma}_{ij}^k + \hat{\Gamma}_{ji}^k\right) X_k \\ &= \frac{1}{2}\langle X_i : X_j \rangle. \end{aligned}$$

We may define the velocity vector $\dot{\gamma}(t) = \dot{\gamma}^i(t)\frac{\partial}{\partial q^i}$ of the curve $\gamma(t)$ in terms of the family of vector fields $\{\mathscr{Y}, \mathscr{Y}^\perp\}$. The new expression for $\dot{\gamma}(t)$ is in the form

$$\dot{\gamma}(t) = w^a(t) Y_a(\gamma(t)) + s^b(t) Y_b^\perp(\gamma(t)) \tag{3}$$

where $s^b(t) = \langle\langle \dot{\gamma}(t), Y_b^\perp \rangle\rangle_{\gamma(t)}$. We now provide a local expression for a measure of a simple mechanical control system's ability to move among the leaves of the input foliation $\mathscr{A}_\mathscr{Y}$.

Lemma 2.1. *Let $(M, \mathbb{G}, V, \mathscr{Y}, U)$ be a simple mechanical control system with an input foliation $\mathscr{A}_\mathscr{Y}$ defined above. The following holds along the curve $\gamma(t)$ satisfying (7):*

$$\begin{aligned}\frac{d}{dt}s^b(t) = &-\frac{1}{2}w^a(t)w^p(t)\langle\langle\langle Y_a : Y_p\rangle, Y_b^\perp\rangle\rangle \\ &-\frac{1}{2}w^a(t)s^r(t)\langle\langle\langle Y_a : Y_r^\perp\rangle, Y_b^\perp\rangle\rangle \\ &-\frac{1}{2}s^r(t)w^p(t)\langle\langle\langle Y_r^\perp : Y_p\rangle, Y_b^\perp\rangle\rangle \\ &-\frac{1}{2}s^r(t)s^k(t)\langle\langle\langle Y_r^\perp : Y_k^\perp\rangle, Y_b^\perp\rangle\rangle\end{aligned} \tag{4}$$

where $a, p \in \{1,\ldots,m\}$, $b,k,r \in \{1,\ldots,n-m\}$.

Proof. Recall from the definition of an input foliation that

$$s^b(t) = \langle\langle Y_b^\perp, \dot{\gamma}(t)\rangle\rangle. \tag{5}$$

We could proceed by substituting (3) into (5) and differentiating taking advantage of the compatibility associated with the Levi-Civita connection. Alternatively, we use the notion of a generalized symmetric Christoffel symbol. It follows from the construction of $\mathscr{Y}^\perp$ that the bth component of $\tilde{\Gamma}_{ij}^b$ along the the orthonormal vector field $Y_b^\perp$ can be expressed as a projection using $\mathbb{G}$.

We observe that (5) is quadratic in the parameter $w(t)$. Now we relate an intrinsic vector-valued symmetric bilinear form to the measure derived in Lemma 2.1.

Definition 2.3. Let $(M,\mathbb{G},V,\mathscr{Y},U)$ be a simple mechanical control system with the input distribution $\mathscr{Y}$ generated by $\{Y_1,\ldots,Y_m\}$ and the corresponding $\mathbb{G}$-orthogonal distribution $\mathscr{Y}^\perp$ generated by $\{Y_{m+1}^\perp,\ldots,Y_n^\perp\}$. We define the **intrinsic vector-valued symmetric bilinear form** to be $B : \mathscr{Y}_q \times \mathscr{Y}_q \to \mathrm{d}_q^\perp$ given in coordinates by

$$B_{ap}^b w^a w^p = \frac{1}{2}\langle\langle\langle Y_a : Y_p\rangle, Y_b^\perp\rangle\rangle w^a w^p,$$

where $a, p \in \{1,\ldots,m\}, b \in \{1,\ldots,n-m\}$.

Remark 2.2. If Σ is underactuated by one control then $b = 1$ and B is a real-valued symmetric bilinear form.

The intrinsic vector-valued symmetric bilinear form defined above is an important measure of how the velocity components w parallel to the input forces influence the velocity components s orthogonal to the input forces. The remainder of the paper will detail the properties of B that form the foundation for the stopping algorithm.

3 Stopping Algorithm

The stopping algorithm consists of three simple stages. The first stage of the algorithm is driving the w-velocities towards the appropriate eigenvector. Recall that the

symmetric form measures the influence the w-velocities have on the s-velocity. If we wish to decrease the s-velocity then we drive the w-velocities toward the eigenvector associated with the most negative eigenvalue of the symmetric form. In contrast, if we desire to increase the s-velocity then we drive the w-velocities toward the eigenvector associated with the most positive eigenvalue of the symmetric form. For this paper, we assume that the symmetric form is indefinite for almost all configurations. This guarantees the existence of both positive and negative eigenvalues.

The second stage of the algorithm consists of driving w-velocities along the appropriate eigenvector. The third stage of the algorithm consists of driving the w-velocities to zero. This is achieved by choosing a control input directly opposing the current w-velocities. The stopping algorithm cycles through each stage until the magnitude of each velocity component drops below a specified bound. The cycling can be observed in the simulation results for three different mechanical systems underactuated by one control found in Section 4.

4 Examples

4.1 Planar Rigid Body

In this section we review the geometric model of the planar rigid body (Fig. 2).

The configuration manifold for the system is the Lie group $SE(2)$ and the potential function is assumed to be identically zero. Let us use coordinates (x, y, θ) for the planar robot where (x, y) describes the position of the center of mass and θ describes the orientation of the body frame $\{b_1, b_2\}$ with respect to the inertial frame $\{e_1, e_2\}$. In these coordinates, the Riemannian metric is given by

$$\mathbb{G} = mdx \otimes dx + mdy \otimes dy + Jd\theta \otimes d\theta,$$

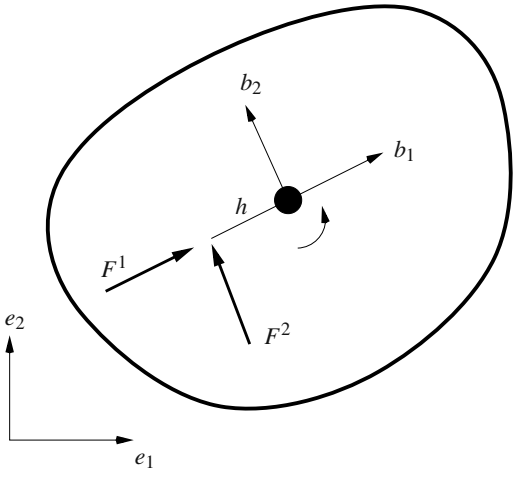


Fig. 2. A schematic of the planar rigid body.

where m is the mass of the body and J is the moment of inertia about the center of mass. The inputs for this system consist of two independent forces applied to an arbitrary point. We assume that the point of application of the force is a distance $h > 0$ from the center of mass along the b_1 body-axis. Physically, the input force can be thought of as a variable-direction thruster on the body which can be resolve into components along the b_1 and b_2 directions. The control inputs are given by

$$\begin{aligned} F^1 &= \cos\theta dx + \sin\theta dy, \\ F^2 &= -\sin\theta dx + \cos\theta dy - hd\theta. \end{aligned}$$

Using Lemma 2.1, we determine that

$$\frac{ds}{dt} = \frac{\sqrt{2}}{2} w^1(t) w^2(t) - \frac{1}{2} s(t) w^1(t).$$

It also follows from Definition 2.3 that the symmetric form is given by

$$B_{12} = B_{21} = -\frac{1}{2}\langle\langle Y_1 : Y_2\rangle, Y^{\perp}\rangle\rangle = \frac{\sqrt{2}}{4}$$

$$B_{11} = B_{22} = 0.$$

Figure 3 is a simulation of the stopping algorithm driving the planar rigid body to rest given the initial velocities $w^1(0) = -10$, $w^2(0) = 5$ and $s(0) = -60$.

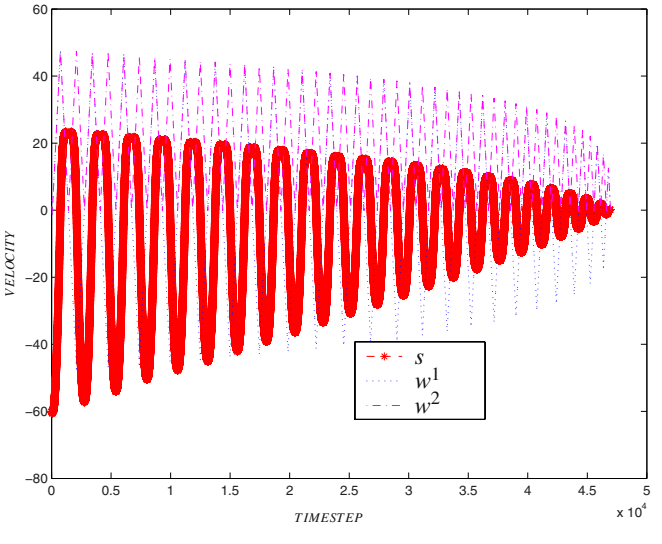


Fig. 3. A simulation of the stopping algorithm applied to the planar rigid body.

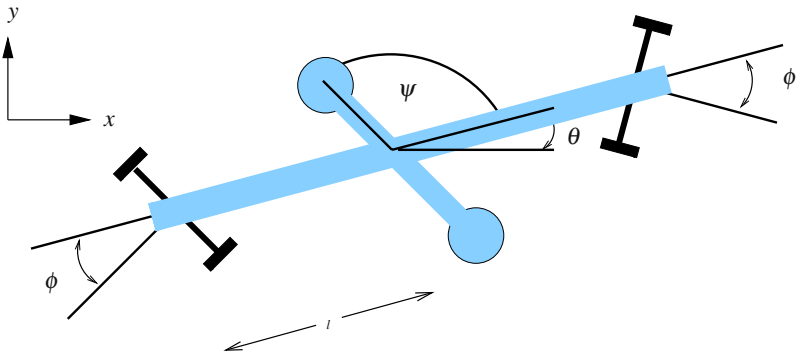


Fig. 4. A schematic of the snakeboard.

4.2 Snakeboard

In this section we review the geometric model of the snakeboard (SB) (Fig. 4).

The configuration manifold for SB is $SE(2) \times \mathbb{S} \times \mathbb{S}$ with local coordinates $(x, y, \theta, \psi, \phi)$. The Riemannian metric is given by

$$\begin{aligned}\mathbb{G} = &\, m dx \otimes dx + m dy \otimes dy + l^2 m d\theta \otimes d\theta \\ &+ J_r d\psi \otimes d\theta + J_r d\theta \otimes d\psi + J_r d\psi \otimes d\psi + J_w d\phi \otimes d\phi,\end{aligned}$$

where $m > 0$ is the total mass of SB, $J_r > 0$ is the moment of inertia of the rotor mounted on top of the body's center of mass, and $J_w > 0$ is the moment of inertia of the wheel axles. The constraint one-forms are given by

$$\begin{aligned}0 &= \sin(\phi - \theta)\, dx + \cos(\phi - \theta)\, dy + l \cos(\phi)\, d\theta, \\ 0 &= -\sin(\phi + \theta)\, dx + \cos(\phi + \theta)\, dy - l \cos(\phi)\, d\theta.\end{aligned}$$

The two control forces are pure torques $F^1 = d\psi$ and $F^2 = d\phi$.

Using Lemma 2.1, we determine that

$$\frac{ds}{dt} = 2\left\{-\cos(\phi)\sqrt{\frac{1}{10\cos(2\phi) + 30}}\right\} w^1(t) w^2(t)$$

It also follows from Definition 2.3 that the symmetric form is given by

$$B_{12} = B_{21} = -\frac{1}{2}\langle\langle\langle Y_1 : Y_2\rangle, Y^{\perp}\rangle\rangle = -\cos(\phi)\sqrt{\frac{1}{10\cos(2\phi) + 30}}$$

$$B_{11} = B_{22} = 0$$

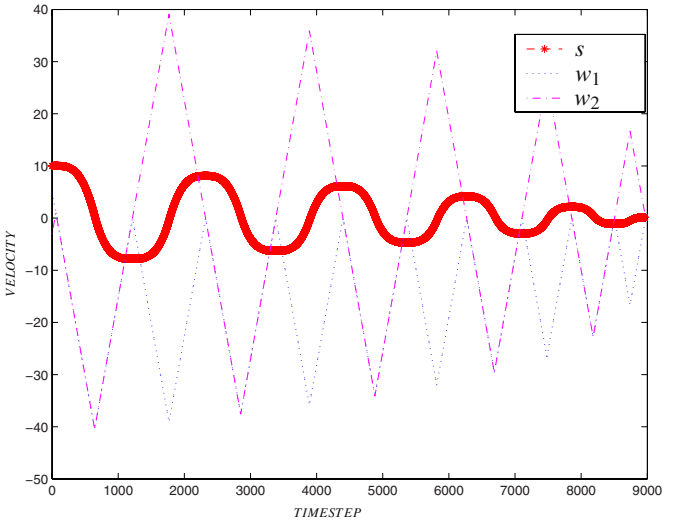


Fig. 5. A simulation of the stopping algorithm applied to the snakeboard.

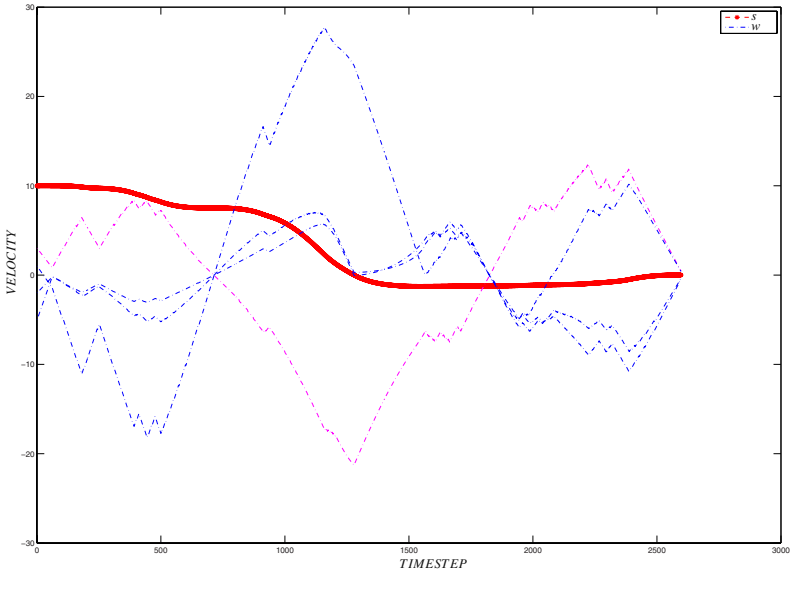


Fig. 6. A simulation of the stopping algorithm applied to the planar rollerblader.

Figure 5 is a simulation of the stopping algorithm driving the snakeboard to rest given the initial velocities $w^1(0) = 5$, $w^2(0) = -3$ and $s(0) = 10$.

4.3 Planar Rollerblader

The schematic model and description of the planar rollerblader (RB) can be found in the introduction of this paper. The configuration manifold for RB is $SE(2) \times \mathbb{S} \times \mathbb{R} \times \mathbb{S} \times \mathbb{R}$ with local coordinates $(x, y, \theta, \gamma_1, d_1, \gamma_2, d_2)$. The Riemannian metric is given by

$$
\begin{aligned}
\mathbb{G} = & \left(m + \frac{M}{2}\right) dx \otimes dx + \frac{1}{2}(d_1 m \cos(\gamma_1) - d_2 m \cos(\gamma_2)) dx \otimes d\theta \\
& + \frac{1}{2} d_1 m \cos(\gamma_1) dx \otimes d\gamma_1 + \frac{1}{2} m \sin(\gamma_1) dx \otimes dd_1 \\
& - \frac{1}{2} d_2 m \cos(\gamma_2) dx \otimes d\gamma_2 + \frac{1}{2} m \sin(\gamma_2) dx \otimes dd_2 \\
& + \left(m + \frac{M}{2}\right) dy \otimes dy + \frac{1}{2}(d_1 m \sin(\gamma_1) - d_2 m \sin(\gamma_2)) dy \otimes d\theta \\
& + \frac{1}{2} d_1 m \sin(\gamma_1) dy \otimes d\gamma_1 - \frac{1}{2} m \cos(\gamma_1) dy \otimes dd_1 \\
& - \frac{1}{2} d_2 m \sin(\gamma_2) dy \otimes d\gamma_2 + \frac{1}{2} m \cos(\gamma_2) dy \otimes dd_2 \\
& + \frac{1}{2}(d_1 m \cos(\gamma_1) - d_2 m \cos(\gamma_2)) d\theta \otimes dx + \frac{1}{2}(d_1 m \sin(\gamma_1) - d_2 m \sin(\gamma_2)) d\theta \otimes dy \\
& + \left(m b^2 + d_1 m \cos(\gamma_1) b + d_2 m \cos(\gamma_2) b + \frac{I_c}{2} + I_p + \frac{d_1^2 m}{2} + \frac{d_2^2 m}{2}\right) d\theta \otimes d\theta \\
& + \frac{1}{2}\left(m d_1^2 + b m \cos(\gamma_1) d_1 + I_p\right) d\theta \otimes d\gamma_1 + \frac{1}{2} b m \sin(\gamma_1) d\theta \otimes dd_1 \\
& + \frac{1}{2}\left(m d_2^2 + b m \cos(\gamma_2) d_2 + I_p\right) d\theta \otimes d\gamma_2 + \frac{1}{2} b m \sin(\gamma_2) d\theta \otimes dd_2 \\
& + \frac{1}{2} d_1 m \cos(\gamma_1) d\gamma_1 \otimes dx + \frac{1}{2} d_1 m \sin(\gamma_1) d\gamma_1 \otimes dy \\
& + \frac{1}{2}\left(m d_1^2 + b m \cos(\gamma_1) d_1 + I_p\right) d\gamma_1 \otimes d\theta + \left(\frac{m d_1^2}{2} + \frac{I_p}{2}\right) d\gamma_1 \otimes d\gamma_1 \\
& + \frac{1}{2} m \sin(\gamma_1) dd_1 \otimes dx - \frac{1}{2} m \cos(\gamma_1) dd_1 \otimes dy \\
& + \frac{1}{2} b m \sin(\gamma_1) dd_1 \otimes d\theta + \frac{m}{2} dd_1 \otimes dd_1 \\
& - \frac{1}{2} d_2 m \cos(\gamma_2) d\gamma_2 \otimes dx - \frac{1}{2} d_2 m \sin(\gamma_2) d\gamma_2 \otimes dy \\
& + \frac{1}{2}\left(m d_2^2 + b m \cos(\gamma_2) d_2 + I_p\right) d\gamma_2 \otimes d\theta + \left(\frac{m d_2^2}{2} + \frac{I_p}{2}\right) d\gamma_2 \otimes d\gamma_2 \\
& + \frac{1}{2} m \sin(\gamma_2) dd_2 \otimes dx + \frac{1}{2} m \cos(\gamma_2) dd_2 \otimes dy \\
& + \frac{1}{2} b m \sin(\gamma_2) dd_2 \otimes d\theta + \frac{m}{2} dd_2 \otimes dd_2.
\end{aligned}
$$

Let the mass and rotational inertia of the central platform of the robot be M and I_c respectively. Let each link have a rotational inertia I_p. The mass of the link is assumed to be negligible. Each roller blade has mass m, but is assumed to have no rotational inertia. The constraint one-forms are given by

$$\begin{aligned} 0 &= -\sin(\theta+\gamma_1)dx+\cos(\theta+\gamma_1)dy-b\sin(\gamma_1)d\theta-dd_1 \\ 0 &= -\sin(\theta+\gamma_2)dx+\cos(\theta+\gamma_2)dy-b\sin(\gamma_2)d\theta+dd_2. \end{aligned}$$

The four control forces consist of two torques $F^1 = d\gamma_1$ and $F^2 = d\gamma_2$, as well as, two linear actuators $F^3 = dd_1$ and $F^4 = dd^2$.

Here we only include the simulation results due to the complexity associated with the symbolic representation of the symmetric form. Figure 6 is a simulation of the stopping algorithm driving the planar rollerblader to rest given the initial velocities $w^1(0) = 1$, $w^2(0) = -1$, $w^3(0) = 2$, $w^4(0) = -5$ and $s(0) = 10$.

5 Conclusions

We seek to extend our results to mechanical systems underactuated by an arbitrary number of controls. This will involve characterizing coordinate invariant properties of the intrinsic *vector-valued* symmetric bilinear form that allow motion in the input foliation.

References

1. Bullo, F., Lewis, A.: Geometric Control of Mechanical Systems. Springer Science+Business Media, New York (2005)
2. Bullo, F., Lewis, A.: Low-order controllablility and kinematic reductions for affine connection control systems. SIAM Journal on Control and Optimization 44(3), 885–908 (2005)
3. Bullo, F., Lewis, A.: Kinematic controllability and motion planning for the snakeboard. IEEE Transactions on Robotics and Automation 19(3), 494–498 (2003)
4. Ostrowski, J.P., Desai, J.P., Kumar, V.: Optimal gait selection for nonholonomic locomotion systems. The International Journal of Robotics Research 19(5), 225–237 (2000)
5. Bullo, F., Zefran, M.: On mechanical control systems with nonholonomic constraints and symmetries. Systems and Control Letters 45(2), 133–143 (2002)
6. Hirschorn, R., Lewis, A.: Geometric local controllability: second-order conditions. In: Proceedings of the 41st IEEE CDC, vol. 1, December 2002, pp. 368–369 (2002)
7. Bullo, F., Cortes, J., Lewis, A., Martinez, S.: Vector-valued quadratic forms in control theory. In: Blondel, V., Megretski, A. (eds.) Sixty Open Problems in Mathematical Systems and Control Theory, pp. 315–320. Princeton University Press, Princeton (2004)
8. Chitta, S., Kumar, V.J.: Dynamics and generation of gaits for a planar rollerblader. In: Proceedings of the 2003 IEEE IROS, vol. 1, pp. 860–865 (2003)
9. Lewis, A., Ostrowski, J.P., Murray, R., Burdick, J.: Nonholonomic mechanics and locomotion: the snakeboard example. In: Proceedings of the 1994 IEEE ICRA, pp. 2391–2400 (1994)
10. Bullo, F., Murray, R.: Tracking for fully actuated mechanical systems: A geometric framework. Automatica. The Journal of IFAC 35(1), 17–34 (1999)

11. Lewis, A.: Simple mechanical control systems with constraints. IEEE Transactions on Automatic Control 45(8), 1420–1436 (2000)
12. do Carmo, M.P.: Riemannian Geometry. Birkhauser, Boston (1992); English edn.
13. Nightingale, J., Hind, R., Goodwine, B.: Geometric analysis of a class of constrained mechanical control systems in the nonzero velocity setting. In: Proceedings of the 2008 IFAC (2008)
14. Nightingale, J., Hind, R., Goodwine, B.: Intrinsic Vector-Valued Symmetric Form for Simple Mechanical Control Systems in the nonzero velocity setting. In: Proceedings of the 2008 IEEE ICRA (2008)

Part IV
Robust Planning

Perceived CT-Space for Motion Planning in Unknown and Unpredictable Environments*

Rayomand Vatcha and Jing Xiao

Abstract. One of the ultimate goals in robotics is to make high-DOF robots work autonomously in unknown changing environments. However, motion planning in completely unknown environments is largely an open problem and poses many challenges. One challenge is that in such an environment, the configuration-time space (CT-space) of a robot is not known beforehand. This paper describes how guaranteed collision-free regions in the unknown CT-space can be discovered progressively via sensing in real time based on the concept *dynamic envelope*, which is not conservative, i.e., does not assume worst-case scenarios, and is robust to uncertainties in obstacle behaviors. The introduced method can be used in general by real-time motion planners for high-DOF robots to discover the existence of guaranteed collision-free future motions efficiently. The utility is further confirmed both in simulation and in real-world testing involving a 5-DOF robot manipulator.

1 Introduction

Most of the existing work addresses robot motion planning in *known* environments, which can be catogorized into the following:

1. Path planning for a robot in a static and known environment to search a collision-free path in the (static) configuration space (C-space) [6] of the robot. Approaches include finding collision-free regions or free space in the C-space [3][10] and sampling-based planners to deal with high-dimensional C-space for robots of high degrees of freedom (DOF)[2][4].

Rayomand Vatcha
Department of Computer Science, University of North Carolina - Charlotte
e-mail: rvatcha@uncc.edu

Jing Xiao
Department of Computer Science, University of North Carolina - Charlotte
e-mail: xiao@uncc.edu

* This work is supported by the National Science Foundation Grant IIS-0742610.

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 183–198.
springerlink.com

2. Motion planning for a robot in a known dynamic environment to search a trajectory in the CT-space [3] of the robot. Again, sampling-based planners were used here [4] to avoid constructing high-dimensional CT-obstacles. For a mobile robot, the notion of "Inevitable Collision Regions" (ICS) in the CT-space was introduced [1] essentially to characterize the CT-obstacles.

In both cases planning can be done off-line without the need of sensing (sensing is only used to deal with uncertainities during actual execution of motion). Collision detection is usually the most time consuming component of any sampling-based planner. Its complexity increases as the complexity of the robot and environment increases.

An extension to the above basic problems is through adding some obstacles of unknown motion into the largely known environment. This is mostly addressed by on-line revising pre-planned paths with reactive schemes to avoid collisions (e.g., [11] [16] [5]). These schemes usually assume partially unchanging/known C-space or CT-space to limit the scale of revising/replanning in order to facilitate real-time computation.

A further extension is motion planning in drastically changing environments with unknown obstacle motions. A real-time adaptive planning approach [9] [8] for high-DOF robots is very effective, characterized by simultaneous planning and execution based on sensing. However, the approach assumes known obstacles with unknown motions. Thus, planning future motion is based on predicting obstacle motion through tracking and frequently updating the predictions.

Note that in those approaches, unknown changes in an environment are dealt with by repeated computation or recomputation of (some parts of) paths/trajectories, which involve repeated collision checking.

Another extension is motion planning in unknown but static environments. No information about the obstacles is known. Such a problem needs active sensing of the environment. One approach represents an environment in terms of voxels so that obstacle geometry need not be known [13]. Sensing is used to discover which voxels are occupied by obstacles. Such sensor based motion planners [14] [12] are often adapted from model based planners to plan paths incrementally, as unknown C-Space becomes known gradually by sensing.

A largely open problem is motion planning in completely unknown environments, where obstacle geometries and states are unknown, i.e., if and when they move or not is not known. In other words, the CT-space of a robot is completely unknown. The existing approaches to deal with known obstacles of unknown motions are not suitable here as obstacles cannot be distinguished, and thus their motions cannot be tracked and predicted.

This paper addresses this open problem. We present a novel approach to discover guaranteed collision-free regions in the unknown and unpredictable CT-space in real-time via sensing. We show how the approach can be used for real-time motion planning in such an environment and test it through simulation and experiments.

2 Perceived CT-Space vs. Predicted CT-Space

We are interested in discovering true collision-free regions, or free space, in the CT-space of a robot in an unknown and unpredictable environment (i.e., obstacles are not known and whether and how they move are not known).

This is different from the existing approaches that predict the CT-space to deal with obstacles of unknown motion. Such approaches predict an obstacle's motion mostly by tracking its location or assuming constant velocity within a planning period. Motion planning is then done in the *predicted CT-space* of a robot, involving collision checking of the robot's configuration at any future time against the predicted obstacle configurations at the same time instant. However, if obstacles themselves are unknown, they cannot be detected or tracked so that prediction-based approaches are not suitable.

Moreover, predicted CT-space is often not the true CT-space and only matches closely to it within a short period immediately after the time when the prediction is made. It requires repeated modification as new sensing information becomes available. Hence, motion planning based on prediction will lead to re-computation of motions, and the planned motions may fail to be collision-free due to inaccurately predicted CT-space. It is too conservative to assume worst-case obstacle motion in order to have guaranteed collision-free motions of the robot with the CT-obstacle space exaggerated.

We use the term *perceived CT-space* to call the CT-space discovered by our approach via sensing, which includes actual (i.e., guaranteed) collision-free regions discovered that will not turn false later as sensing continues. Therefore, it is not the same as a predicted CT-space. Figure 1 illustrates the difference between the two. It compares predicted vs. perceived vs. actual CT-space in a 2-D example. Both

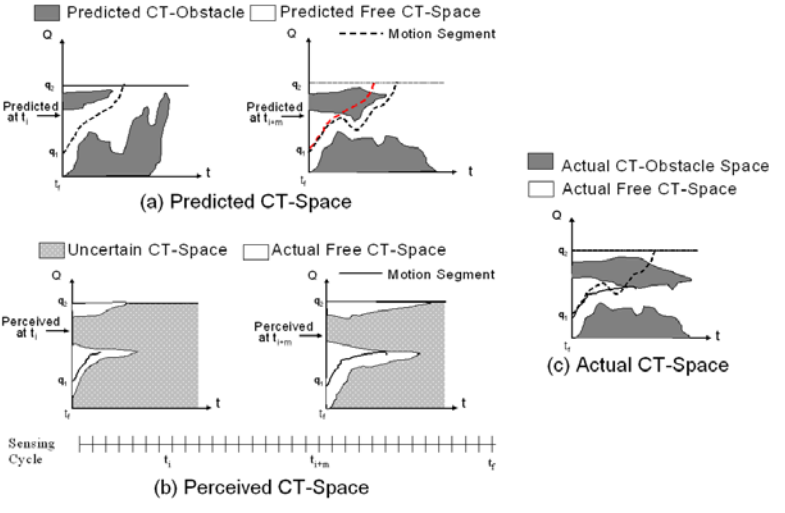


Fig. 1. Predicted CT-space vs. Perceived CT-Space

predicted CT-space and perceived CT-space will change as sensing/time progresses. However, unlike predicted CT-space, where a point predicted collision-free may not be actually collision-free, the perceived CT-space consists of *actual* collision-free regions that can only grow over time and *uncertain* regions, which can be either free or CT-obstacle regions.

3 Perceived CT-Space

We now show how the actual CT-free space can be perceived over time.

3.1 Atomic Obstacles

First of all, with obstacles completely unknown in an environment, it will be difficult to try to distinguish different geometric obstacles from sensing. For static environments, this problem can be addressed by, for example, representing an environment in terms of voxels without knowing obstacle geometry [13]. However, if the environment is changing *drastically*, this is computationally costly as the entire set of voxels are only valid for the current sensing interval and have to be re-computed at each subsequent sensing interval. Hence, it makes sense to use the lower-level data from sensors *directly* to represent obstacles without ever performing elaborate sensor information processing. Without the loss of generality, the lower-level sensory data for obstacles can be treated as *atomic obstacles* of similar and simple geometry at different locations. Collectively the atomic obstacles represent actual obstacles in an environment without distinguishing them (see Section 4). At each sensing instant, only the locations of atomic obstacles (with default geometry) can be sensed. However, we can put an upper bound on the changing rate of the environment in terms of a maximum possible speed v_{max} of each atomic obstacle. Of course, an atomic obstacle may have varied actual speeds in $[0, v_{max}]$.

3.2 Dynamic Envelope

We are interested in discovering collision-free regions via sensing in real-time in the otherwise unknown CT-space for a general (high-DOF) robot. Assume sensing data are obtained/updated at discrete times starting at $t = 0$. For any CT-point $(\mathbf{q},t)$, we ask the **question**: *will the robot be guaranteed collision-free at* $(\mathbf{q},t)$ *based on the sensed information at the current time* t_i?

We can answer this question using the concept of *dynamic envelope*.

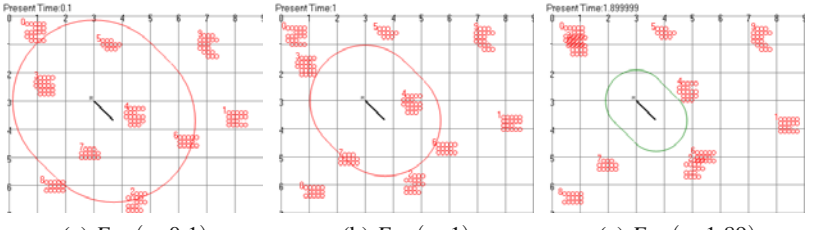
Definition 3.1. For a CT-point $\chi = (\mathbf{q},t)$, a *dynamic envelope* $E_{t_o}(\chi,t_i)$, as a function of current time $t_i \leq t$, is a closed surface enclosing the region occupied by the robot at configuration $\mathbf{q}$ in the physical space so that the minimum distance between any point on $E_{t_o}(\chi,t_i)$ and the region is $d_i = v_{max}(t - t_i)$, where $t_o \leq t_i$ is the time when the envelope was created. $t - t_o$ is the maximum *lifespan* of $E_{t_o}(\chi,t_i)$.

The following are general properties of a dynamic envelope $E_{t_o}(\chi, t_i)$. They capture non-worst case scenarios regarding atomic obstacle motions, *without* assuming any particular kinds of obstacle motion.

1. A dynamic envelope shrinks monotonically over sensing time with speed v_{max}, i.e., $E_{t_o}(\chi, t_{i+m}) \subset E_{t_o}(\chi, t_i)$, where $m > 0$, $t_o \leq t_i < t_{i+m} \leq t$.
2. An atomic obstacle not on or inside $E_{t_o}(\chi, t_o)$ will never be on or inside $E_{t_o}(\chi, t_i)$.
3. An atomic obstacle on $E_{t_o}(\chi, t_o)$ will never be inside $E_{t_o}(\chi, t_i)$.
4. An atomic obstacle either on or inside $E_{t_o}(\chi, t_o)$ can be outside $E_{t_o}(\chi, t_i)$, for certain t_i, if not moving *towards* the robot in maximum speed v_{max} *all the time*, i.e., if not moving in the worst case.

From these properties, one can envision the following: suppose some atomic obstacles are on or inside a dynamic envelope $E_{t_o}(\chi, t_i)$ initially; as the dynamic envelope shrinks during its maximum lifespan, no new atomic obstacles will ever enter the dynamic envelope, and the atomic obstacles initially on or inside it will be "squeezed" out of the envelope at some later time during the lifespan *if these atomic obstacles do not always move towards the robot configuration* $\mathbf{q}$ *in* v_{max}*, i.e., under non-worst case scenarios*. Hence, at time t_i, if no atomic obstacle is on or inside the dynamic envelope $E_{t_o}(\chi, t_i)$, $\chi = (\mathbf{q}, t)$ is guaranteed collision-free, i.e., the above **question** is answered. Moreover, all the continuous configuration-time points in the interval $[(\mathbf{q}, t_i), (\mathbf{q}, t)]$ are guaranteed collision-free.

Figure 2 shows an example, where $\chi = ((3,3),3)$, $v_{max} = 1$ unit/s.



(a) $E_{0.1}(\chi, 0.1)$ (b) $E_{0.1}(\chi, 1)$ (c) $E_{0.1}(\chi, 1.89)$

Fig. 2. Dynamic envelope of a planar rod robot. In (c), χ is percieved collision free at $t_i = 1.89$s.

In general, as soon as at some t_l, no atomic obstacle sensed is on or inside the dynamic envelope $E_{t_o}(\chi, t_l)$, the envelope is no longer needed, and it can *expire* at t_l, i.e., before its maximum lifespan is reached.

3.3 Collision-Free Region vs. Uncertain Region

We have answered in the above that a CT-point $(\mathbf{q}, t)$ can be perceived at $t_i (\leq t)$ as guaranteed collision-free and also explained that if $(\mathbf{q}, t)$ is perceived at t_i as

guaranteed collision-free, the hyperline segment $[(\mathbf{q},t_i),(\mathbf{q},t)]$ in the CT-space is also guaranteed collision-free.

Now a natural next question is: *given a configuration* $\mathbf{q}$*, what is the longest hyperline segment* $[(\mathbf{q},t_i),(\mathbf{q},t)]$*, or the furthest time* t*, that can be perceived at* t_i *as guaranteed collision-free?* The answer to that question depends on the minimum distance $d_{min}(\mathbf{q},t_i)$ between the robot (if it were) at configuration $\mathbf{q}$ and the closest atomic obstacle sensed at t_i. Let

$$\Delta t(\mathbf{q},t_i) = \frac{d_{min}(\mathbf{q},t_i)}{v_{max}}, \tag{1}$$

which is the minimum period before a collision can *possibly* occur at $\mathbf{q}$. Let

$$t_f(\mathbf{q},t_i) = t_i + \Delta t(\mathbf{q},t_i) \tag{2}$$

Clearly, as long as t is within the time interval $[t_i,t_f(\mathbf{q},t_i))$, the hyperline segment $[(\mathbf{q},t_i),(\mathbf{q},t)]$ can be perceived at t_i as guaranteed collision-free. Thus, the longest hyperline segment that can be perceived at t_i as guaranteed collision-free is $[(\mathbf{q},t_i),(\mathbf{q},t_f(\mathbf{q},t_i))$.

The union of all the guaranteed collision-free hyperline segments of the CT-space perceived at t_i is the maximum collision-free region (that may include multiple connected continuous regions) perceived at t_i, denoted as $F(t_i)$. $F(t_i)$ consists of only CT-points for $t \geq t_i$. The union of the rest of the regions in the CT-space for time $t \geq t_i$ forms the uncertain region $U(t_i)$.

Theorem 3.1. *For any* t_i *and* t_j*, such that* $t_i \leq t_j$*, if a CT-point* $(\mathbf{q},t)$*, where* $t \geq t_j$*, belongs to* $F(t_i)$*, then it also belongs to* $F(t_j)$*. On the other hand, if the point* $(\mathbf{q},t)$ *belongs to* $U(t_i)$*, it may still belong to* $F(t_j)$*.*

Proof. From t_i to t_j, the change in minimum distance at configuration $\mathbf{q}$ can be expressed as:

$$d_{min}(\mathbf{q},t_j) - d_{min}(\mathbf{q},t_i) = p v_{max}(t_j - t_i),\ -1 \leq p \leq 1. \tag{3}$$

From equations (1) and (2), and using equation (3), we get

$$t_f(\mathbf{q},t_j) - t_f(\mathbf{q},t_i) = (t_j - t_i) + \frac{d_{min}(\mathbf{q},t_j) - d_{min}(\mathbf{q},t_i)}{v_{max}} = (1+p)(t_j - t_i)$$
$$\Rightarrow t_f(\mathbf{q},t_j) - t_f(\mathbf{q},t_i) \geq 0$$

That is, if $(\mathbf{q},t)$ is on the hyperline $[t_i,t_f(\mathbf{q},t_i))$, then, since $t \geq t_j$, it is also on the hyperline $[t_j,t_f(\mathbf{q},t_j))$. On the other hand, if $(\mathbf{q},t)$ belongs to $U(t_i)$, then $t \geq t_f(\mathbf{q},t_i)$, but as long as $t < t_f(\mathbf{q},t_j)$, $(\mathbf{q},t)$ belongs to $F(t_j)$. □

The significance of the above theorem is that more collision-free CT-space points can be discovered as sensing time progresses, i.e., the collision-free regions can only grow, while uncertain regions can only shrink.

4 Representing Unknown Obstacles of Unknown Geometry: An Example

There is much research on how to explore an unknown (mostly static) environment using robots with sensors mounted, and issues studied include how to move a robot to maximize sensing views (i.e., minimize occlusions) [15] and how to map an environment accurately through repeated exploration (e.g., SLAM [7]). For sensor-based robot navigation, different kinds of sensors are used either mounted in the environment to provide a world view or mounted on a robot to provide a robot-centric, local view. However, these issues regarding the arrangement and style of sensing is out of the scope of this paper. It is important to note that the concepts introduced in Section 3 are independent of any kind of sensing style.

In general, the lowest level data points of whatever sensor (e.g., laser range finders, sonar, etc.) constitute atomic obstacles (as mentioned in Section 3). As a concrete example, consider that an overhead stereovision sensor is used to provide a view of an unknown environment. The stereovison sensor provides an image of the environment. Every pixel (i, j) of that image maps to a surface region R_{ij} of 3-D points in the physical world. We can further obtain the 3-D point (x, y, z) in R_{ij} that is closest to the image plane. This mapping between 3-D point (x, y, z) and pixel (i, j) is a one-to-one mapping.

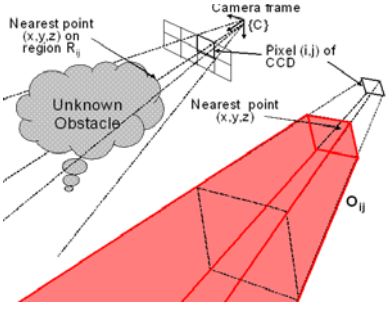


Fig. 3. An atomic obstacle O_{ij} from stereo vision

Since R_{ij} occludes the space behind it, from it one cannot tell if and how there are objects in that occluded space. Therefore, to be safe, R_{ij} and the infinite volume of points it occludes can be viewed as an atomic obstacle O_{ij} that a robot cannot collide with. O_{ij} is associated with a pixel (i, j) of the image, which starts from the point (x, y, z) extending towards infinity. It can be viewed as a trapezoidal ray originated from (x, y, z) as shown in Figure 3. The 3-D environment can now be viewed as consisting of *only* these atomic obstacles O_{ij} for all (i, j)'s in the image. Let $M \times N$ be the size of the image.

Note that since actual obstacles in the environment are entirely *unknown* and can move/change unpredictably, we cannot relate the atomic obstacles from an image taken at time t_k to those from a image taken at t_{k+1}. Thus, the low-level sensory data

from t_k is only useful within that sensing interval and should be replaced entirely by the sensory data obtained from t_{k+1}. In other words, there is no need for accumulating sensory data, and the space complexity for storing sensory data is simply a constant $C = M \times N$.

Now consider the time complexity of using the stored data. Since the sensed atomic obstacles are used for perceiving collision-free or uncertain CT-points, only those atomic obstacles enclosed in a dynamic envelope (of the robot at the considered CT-point) need to be considered. For this example, atomic obstacles are *directly* indexed by (i, j) in a one-to-one mapping between a physical 3-D point (x, y, z) to a pixel (i, j). Projecting a dynamic envelope onto the image plane, we can obtain the indices (i, j) of the atomic obstacles enclosed and consider only them for collision test. Let n indicate the number of such indices, then for any CT-point $(\mathbf{q}, t)$, the time complexity of collision test is a function of n, which is usually much smaller than C. As the dynamic envelope shrinks overtime, so is the time for collision test.

Of course, as viewing directions change, the atomic obstacles as defined above change too, but that does not matter because we are not concerned here with what the actual obstacles look like. Also note that the atomic obstacles do not have to have the same size. It is important that the atomic obstacles come directly from sensory data and are of simple shapes.

5 Computing Motions in the Perceived CT-Space

The concept of dynamic envelope introduced in section 3 can be used by motion planners to discover collision-free regions in the CT-space for future motions efficiently, which do *not* require re-computation or revision.

Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be two configurations of a robot. Let $(\mathbf{q}_1, t_s)$ and $(\mathbf{q}_2, t_e)$ be two points in the CT-space, and let the current time be $t_o < t_s$. We are interested in finding whether a trajectory segment connecting $(\mathbf{q}_1, t_s)$ and $(\mathbf{q}_2, t_e)$, where $t_s < t_e$, is collision-free or not, based on sensing at each $t_i \in [t_o, t_s)$, $i = 1, 2, \ldots$. The trajectory segment can be represented by a sequence Γ of CT-points between $(\mathbf{q}_1, t_s)$ and $(\mathbf{q}_2, t_e)$ through interpolation. If the resolution for interpolation is chosen such that the maximum gap between the robot put at two consecutive CT-space points (after interpolation) is smaller than the known size of an atomic obstacle, then if the two CT-space points are guaranteed collision-free, the CT-points in between are also guaranteed collision-free. In that sense, the sequence Γ truly represents a continuous motion segment in the perceived CT-Space.

For every point $\chi_k = (\mathbf{q}_k, t^k)$ in the sequence Γ, where $t^1 = t_s$, $t^m = t_e$, and $1 \leq k \leq m$, we need to check if it is collision-free or not. This can be done by creating the dynamic envelope $E_{t_o}(\chi_k, t_o)$ at t_o for each k. Note that for a robot consisting of many links, a dynamic envelope can be built for each link with a simple shape. We next observe all $E_{t_o}(\chi_k, t_i)$'s along Γ shrink as the sensing time t_i progresses from t_o. If for every dynamic envelope $E_{t_o}(\chi_k, t_i)$, there exists a time $t_i^k \in [t_o, t_s)$ when $E_{t_o}(\chi_k, t_i^k)$ is free of atomic obstacles, then it means Γ is a guaranteed collision-free motion segment, discovered before its starting time t_s. Moreover, if the starting time

Algorithm 1 Collision-Free Perceiver (CFP)

Input configuration-time point $\chi = (\mathbf{q}, t)$, current time t_o
$i = 1$, $t_i = t_o$
Create dynamic envelope $E_{t_o}(\chi, t_i)$
while $t_i < t$ and (not time-limit) **do**
  if no atomic obstacle is on or inside $E_{t_o}(\chi, t_i)$ **then**
    $E_{t_o}(\chi, t_i)$ expires
    return $\chi \in F(t_j), \forall t_i \leq t_j \leq t$, ($\chi$ is guaranteed collision-free)
  else
    $i = i + 1$ (Next sensing moment)
  end if
end while
return χ may not be collision-free

of Γ is moved to t'_s, such that $\max(t_i^k) \leq t'_s < t_s$ (i.e., earlier than t_s), the shifted Γ (along the time axis) is also guaranteed collision-free. This means that the whole sweped region of Γ along the time axis from $\max(t_i^k)$ to t_s in the CT-space is now discovered to be collision-free.

Our general algorithm to check whether a CT-point $\chi = (\mathbf{q}, t)$ is collision-free or not based on the (shrinking) dynamic envelope $E_{t_o}(\chi, t_i)$, for $t_i \in [t_o, t)$, is called the *collision-free perceiver* (CFP) as shown in Algorithm 1. CFP is quite efficient for real-time operation because of the following: (1) CFP returns a boolean value and does not require (more expensive) minimum distance computation. (2) As shown in Section 4, CFP only needs to consider a subset of the atomic obstacles sensed. The number n of such atomic obstacles is related to the size of the dynamic envelope, which shrinks over time. (3) Both the atomic obstacles and the dynamic envelope are of simple shapes. A time-limit for CFP can be further imposed. The above method of computing guaranteed collision-free motion segments in CT-space using CFP can be employed by any real-time motion planner seeking collision-free motion in an unknown and unpredictable environment. As the robot moves along a perceived collision-free trajectory segment, the planner can continue finding subsequent collision-free motion segments. Since the segment currently followed by the robot is truly collision-free, the robot can safely stay on it until it needs to execute a subsequent collision-free motion segment provided by the planner. As the planner does not need to worry if the current segment being executed by the robot will become infeasible, it can solely focus on planning the next motion segment. In section 7, we will show concrete simulation and real-world examples of real-time planning using CFP.

6 Robustness of Approach over Exaggerated v_{max}

As v_{max}, the maximum speed of an atomic obstacle, is the only known or estimated parameter we assume in our approach dealing with completely unknown environment, it is necessary to investigate how robust our approach of perceiving

collision-free CT-space points is with respect to very inaccurate v_{max}. Specifically, it is natural to over-estimate v_{max} to be safe, i.e., the estimated v'_{max} satisfies: $v'_{max} > v_{max}$. The effect of such over-estimation can be stated in the following theorem.

Theorem 6.1. *Let $v'_{max} = cv_{max}, c > 1$, and let $(\mathbf{q},t)$ be a collision-free CT-point. If t'_k and t_k are the respective time instants when $(\mathbf{q},t)$ is perceived to be collision-free, then, they satisfy: $t_k < t'_k \leq t$.*

Proof. Suppose at time t_o, the dynamic envelopes $E_{t_o}((\mathbf{q},t),t_o)$ and $E'_{t_o}((\mathbf{q},t),t_o)$ for ($\mathbf{q}$, t) were created with respect to v_{max} and v'_{max} respectively, where

$$d_o = v_{max}(t - t_o) \text{ , and}$$
$$d'_o = v'_{max}(t - t_o) = cv_{max}(t - t_o)$$

Clearly for any time $t_o \leq t_i < t$, $E'_{t_o}((\mathbf{q},t),t_i)$ is greater than $E_{t_o}((\mathbf{q},t),t_i)$. Suppose further that at least one atomic obstacle was on or inside $E_{t_o}((\mathbf{q},t),t_o)$. Thus, it was also on or inside $E'_{t_o}((\mathbf{q},t),t_o)$.

Suppose at time t_k, where $t_o \leq t_k \leq t$, the dynamic envelope $E_{t_o}((\mathbf{q},t),t_k)$ has shrunk enough to just "squeeze out" atomic obstacles, i.e., the CT-point $(\mathbf{q},t)$ is perceived collision-free. Recall that $d_{min}(\mathbf{q},t_k)$ is the minimum distance between the robot if put at $\mathbf{q}$ and the atomic obstacles. Thus,

$$d_k = v_{max}(t - t_k) = d_{min}(\mathbf{q},t_k) - \varepsilon \tag{4}$$

where $\varepsilon > 0$ is very small. Clearly at t_k, $E'_{t_o}((\mathbf{q},t),t_k)$ still has atomic obstacles because it is larger than $E_{t_o}((\mathbf{q},t),t_k)$. Later, suppose at time instant t'_k, where $t_k < t'_k \leq t$, $E'_{t_o}((\mathbf{q},t),t'_k)$ has shrunk enough to "squeeze out" atomic obstacles in it, perceiving the CT-point $(\mathbf{q},t)$ as collision-free. Thus,

$$d'_k = cv_{max}(t - t'_k) = d_{min}(\mathbf{q},t'_k) - \varepsilon. \tag{5}$$

From (4) and (5), we have $d'_k - d_k = d_{min}(\mathbf{q},t'_k) - d_{min}(\mathbf{q},t_k)$. From the above equation and equation (3), we have

$$d'_k - d_k = pv_{max}(t'_k - t_k),\ -1 \leq p \leq 1 \tag{6}$$

From the equations (4), (5), and (6), we can further obtain

$$t'_k - t_k = (\frac{c-1}{c+p})(t - t_k) \leq t - t_k \tag{7}$$

Hence, $t_k < t'_k \leq t$. □

The significance of the theorem is that, if a CT-point $(\mathbf{q},t)$ is collision-free, then it will be perceived as collision-free *no later than* time t no matter how badly the actual v_{max} is overestimated as v'_{max}. Moreover, $t'_k = t$ only in the very scenario when, at any time $t_i \in [t_k,t]$, the nearest atomic obstacle at t_k originally inside $E_{t_o}((\mathbf{q},t),t_o)$ moves towards the robot's configuration $\mathbf{q}$ with v_{max}, and in all other cases, $t'_k < t$. This shows the robustness of the CFP.

7 Implementation and Experimental Results

We have tested our approach in both simulation and real-world experiments.

7.1 Test in Simulation

A planar rod robot was considered in the test. It can only translate on a plane with a fixed orientation $\theta = -45°$. Thus the robot has two translational degrees of freedom, with reference position set at the top point of the rod. As shown in Figure 4, the robot is initially at a collision-free CT-point $(\mathbf{S}, t_o)$ and needs to reach a goal configuration $\mathbf{G}$ in this completely unknown environment, where there are unknown obstacles of arbitrary shapes formed by what the robot can only sense as identical red circles (called the atomic obstacles). The sensing frequency is 20 Hz. The obstacles can either be static or move randomly with changing speeds no greater than v_{max} units/s, which can be overestimated by the robot as $v'_{max} > v_{max}$.

We want to check how effective the collision-free perceiver (CFP) of Algorithm 1 can be used by a real-time motion planner to guide the rod robot to its goal while avoiding obstacles. While the robot waits at the starting configuration $\mathbf{S}$, the planner can explore the perceived CT-space to find a motion segment for the robot to move. Different planners can be used here, and the difference is only that they will provide different candidate motion segments for CFP to check for feasibility (i.e., if

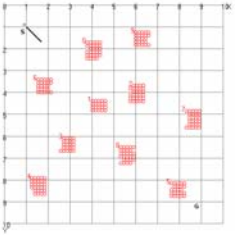


Fig. 4. Simulation environment

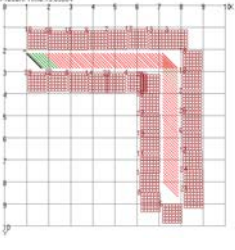
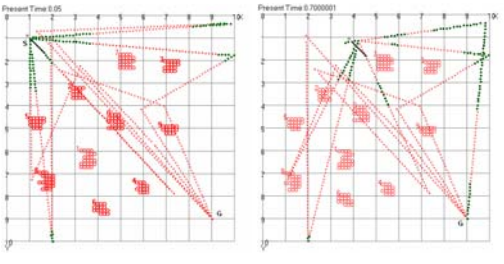
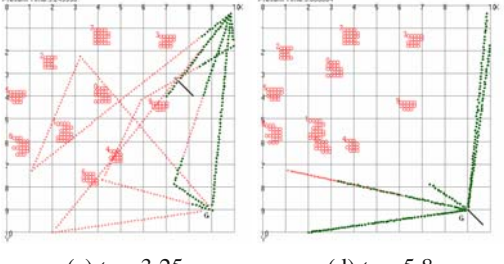


Fig. 5. Static narrow passage of width approximately 1 unit



(a) $t_i = 0.05$s (b) $t_i = 0.7$s



(c) $t_i = 3.25$s (d) $t_i = 5.8$s

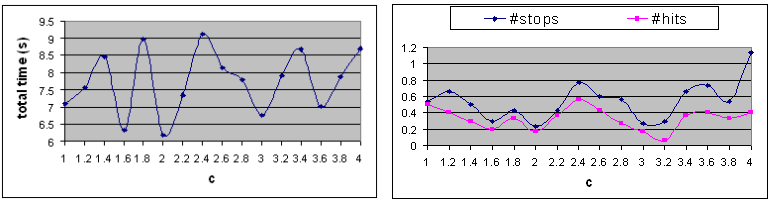
Fig. 6. Snapshots of an example run in simulation

guaranteed collision-free or not). A motion segment can consist of multiple straight-line segments.

We use the real-time adaptive motion planner (RAMP) [9] to provide CFP candidate motion segments. The planner is implemented in C#, on Dell Optiplex GX620. RAMP can simultaneously establish a diverse set of trajectories starting from the robot's current location for the CFP to check and let the robot to execute the best feasible trajectory segment. While the robot executes the guaranteed collision-free trajectory segment, RAMP continues planning subsequent feasible trajectory segments, using CFP. Thus, once the robot finishes executing the current segment, it can hopefully move to the next guaranteed collision-free trajectory segment seamlessly without stop.

Figure 6 shows the snapshots of an example run, when the rod robot of unit length moves from position $\mathbf{S}=(1,1)$ to the goal $\mathbf{G}=(9,9)$ with speed 5 units/s. $v_{max}=1$ unit/s, and the obstacles can change velocities instantly. The sequences of green (or dark in B/W) reference positions show the perceived collision-free trajectory segments (without showing the time instants). The sequences of red reference positions indicate uncertain trajectory segments at each moment of perception, which may or may not be collision-free. The robot executes the best green option found. Note that the robot never hits an obstacle *while moving along a green trajectory* because it is guaranteed collision-free. The attached movie shows the process in four examples with increasing number of obstacles and v_{max}. The robot may only get hit by an obstacle (and momentarily change color to blue) when it cannot find a green trajectory so fast and has to stop its motion.

We tested the effects of overestimating the speed bound v_{max} of obstacles in the same environment of the example run. Figure 7 shows the average results over 30 runs for each c. The *total time* is the average total time for the robot to plan and move simultaneously from the start position to the goal position. If the robot cannot find a collision-free trajectory segment to move to when it reaches the end of the current collision-free trajectory segment, it has to stop its motion until a new collision-free segment is found. Thus, the *# stops* means the average number of stops the robot has to make during its journey from the start position to the goal position. The *# hits* is the average number of times when the robot got hit by obstacles during its stops (i.e., before the next collision-free trajectory segment is found). The results show that increasing c, or the level of over-estimation of v_{max}, has little effect on those



(a) c vs. total time (sec) (b) c vs. # hits and # stops

Fig. 7. Effects of Over-estimating v_{max} as $v'_{max}=cv_{max}$, $c\geq 1$

performance parameters. The ups and downs in the curves reflect the randomness in the environment.

We also tested how our approach works in a static environment ($v_{max} = 0$) with a narrow passage shown in figure 5, where the robot has to move through the narrow passage to a goal, shown in every position of the path. We performed experiments for varied over-estimation $1 \leq v'_{max} \leq 4$. In all cases, the travel time for the robot from the start to the goal position was constant: 2.49s. This, in fact, experimentally verified the second theorem.

7.2 Real-World Experiments with a High DOF Robot

We have also tested the CFP (Algorithm 1) by embedding it in a simple real-time motion planner for a real desktop 5-DOF robot manipulator with revolute joints in an unknown and unpredictable environment, sensed via an overhead stereovision sensor (Figure 8). Our real-time motion planner finds a collision-free straight-line segment in the CT-space as the next-step motion for the robot to execute, with a search method compromising randomized and greedy search[1] and using CFP for discovering collision-free motion. As the robot moves, the planner simultaneously finds again the subsequent next step until the goal is reached.

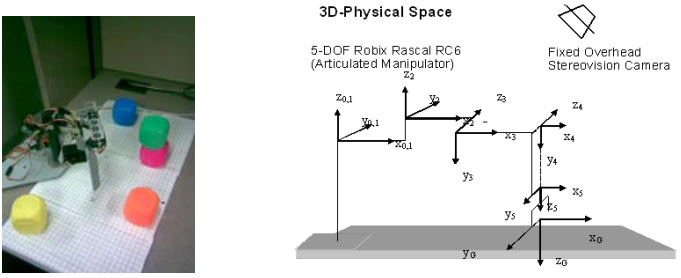
The planner was implemented in C++ on a low-end PC (Dell Optiplex GX260). The 5-DOF manipulator is made from the Robix Rascal RC6 kit. The stereo vision camera is PGR's Digiclops. The obstacles are blocks unknown to the robot, which can be moved in ways also unknown to the robot. Table 1 shows the input parameter values to the planner, where **S** and **G** are the starting and goal configurations respectively. $\dot{q}_{-ve}$ and $\dot{q}_{+ve}$ are the negative and positive bounds on the joint speeds of the robot. Note that the number of atomic obstacles of an actual obstacle increase if the obstacle is close to the origin of the camera frame.

The atomic obstacles generated from stereo vision are as described in section 4. The atomic obstacles representing the robot itself and the known desk surface (as "floor") were filtered out. The shape of an atomic obstacle was approximated as a straight-line ray. The shape of a link of the robot was simplified by a cylindrical bounding volume. The number of atomic obstacles in the test environment were in the range of 345–800. The average rate of collision checking in the CFP computation was 1430.64 CT-points/second.

Figure 9 shows a test environment and two different resulting paths that the robot traveled. The environment had 4 blocks as obstacles, where two were placed at the corners and two were stacked together to form a taller obstacle in between. The taller obstacle created a local optima for the given robot structure with limited dexterity, which our planner was able to overcome.

Figure 10 shows a sequence of selected snapshots of the robot motion in another test environment, where there are four obstacles, and two of them are dynamic,

[1] In this way the planner is able to overcome local minima, but the details of search and handling local minima is not the focus of this paper since a number of different strategies can be used.



(a) (b)

Fig. 8. Experimental setup. Fig(a) shows Robix Rascal RC 6 and obstacles of unknown geometry in its workspace and (b) shows kinematics of the manipulator

Table 1. Input Parameters and Values

S, **G** (degrees)	v_{max} cm/sec	$(\dot{q}_{-ve}, \dot{q}_{+ve})$ (degrees/sec)	*Sensor Image resolution*	*min#O(i, j) per obstacle*
$[-70,45,0,0,0]^T$ $[70,-45,0,0,0]^T$	1	$([-6,-6,-5,-6,-7]^T,$ $[6,6,5,6,7]^T)$	160×120	115

moved by the two hands of a human operator. (Note that a grid of 1×1 cm^2 squares on the desk was used as a guidance to move obstacles close to $v_{max} = 1$cm/s.) As shown, the operator first moved one block towards the robot. Between step 9 to step 18, the planner tried to get most of links closer to their goal positions while avoiding the moving block and the block at the bottom left corner. After step 18, the moving block decreased its speed, and a new block was moved into the visible robot workspace. The planner noticed the reduced speed of the first moving block in time due to the non-conservative nature of the dynamic envelopes and simply guided the robot to pass by the moving block and the static block, while moving away from

(a) Path 1

(b) Path 2

Fig. 9. An environment (Env1) and two traveled paths by the robot

(a) Step #4

(b) Step #9

(c) Step #18

(d) Step #19

(e) Step #25

(f) Step #27

Fig. 10. Selected steps taken by the robot in an unknown dynamic environment (Env2). In (a), robot is near the configuration **S** and in (f), robot is at configuration **G**

Table 2. Average results from two environments (for the same start and goal configurations of the robot)

Env	**Path length** (deg)	**#Steps**	**Total time** (*sec*)	**#Sensing cycle** (*Hz*)
Env1	514.924	62.2	82.8	3.5
Env2	235.98	27	49	4.58

the newly entered block to reach the goal in step 27. Table 2 shows the resulting statistics characterizing the planner performance in the two task environments.

8 Conclusions and Future Work

The paper introduces the notion of perceived CT-Space for a robot, which characterizes what truely collision-free regions can be perceived from sensing in an otherwise completely unknown and unpredictable environment. Through the novel concept of dynamic envelopes complemented by low-level atomic obstacles directly from sensing, the paper presents an approach to discover guaranteed collision-free motion segments to facilitate real-time robot motion planning in completely unknown and unpredictable environments. The approach is in essence efficient because it does not assume worst-case behaviors but rather operates based on perceiving the actual obstacle behaviors, and no re-computation is needed for the already found collision-free motion segments. The approach is also proven robust with respect to unknown maximum velocities of obstacles. It can be used by different motion planners regardless of specific planning strategies. It is tested in both simulation and real experiments with a real 5-DOF robot manipulator.

We will further take into account the robot's position and control uncertainty in producing guaranteed collision-free motions and further test the approach in experiments to see how fast the obstacles have to move relative to the robot for the approach to be infeasible.

References

1. Fraichard, T., Asama, H.: Inevitable collision states. a step towards safer robots? In: Advanced Robotics, pp. 1001–1024 (2004)
2. Kavraki, L., Svestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 566–580 (1996)
3. Latombe, J.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
4. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
5. Leven, P., Hutchinson, S.: A framework for real-time path planning in changing environments. Int. J. of Robotics Research (IJRR), 999–1030 (2002)
6. Lozano-Perez, T.: Spatial planning: A configuration space approach. IEEE Transaction on Computers, 108–120 (1983)
7. Thrun, S.: Robotic mapping: A survey. In: Exploring artificial intelligence in the new millenium. Morgan Kaufmann, San Francisco (2002)

8. Vannoy, J., Xiao, J.: Real-time motion planning of multiple mobile manipulators with a common task objective in shared work environments. In: IEEE ICRA, April 2007, pp. 20–26 (2007)
9. Vannoy, J., Xiao, J.: Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes. IEEE Transactions on Robotics (October 2008)
10. Ward, J., Katupitiya, J.: Free space mapping and motion planning in configuration space for mobile manipulators. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 4981–4986 (2007)
11. Yang, Y., Brock, O.: Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In: Proceedings of Robotics: Science and Systems II (2006)
12. Yili, F., Bao, J., Shuguo, W., Zhengcai, C.: Real-time sensor-based motion planning for robot manipulators. In: IEEE ICRA, pp. 3108–3113 (2005)
13. Yu, Y., Gupta, K.: An efficient on-line algorithm for direct octree construction from range images. In: ICRA, pp. 3079–3084 (1998)
14. Yu, Y., Gupta, K.: Sensor-based probabilistic roadmaps: experiments with an eye-in-hand system. In: Advanced Robotics, pp. 515–536 (2000)
15. Yu, Y., Gupta, K.: C-space entropy: A measure for view planning and exploration for general robot-sensor systems in unknown environments. In: IJRR, pp. 1197–1223 (2004)
16. Zucker, M., Kuffner, J., Branicky, M.: Multipartite rrts for rapid replanning in dynamic environments. In: IEEE ICRA, pp. 1603–1609 (2007)

Bounded Uncertainty Roadmaps for Path Planning

Leonidas J. Guibas, David Hsu, Hanna Kurniawati, and Ehsan Rehman

Abstract. Motion planning under uncertainty is an important problem in robotics. Although probabilistic sampling is highly successful for motion planning of robots with many degrees of freedom, sampling-based algorithms typically ignore uncertainty during planning. We introduce the notion of a *bounded uncertainty roadmap* (BURM) and use it to extend sampling-based algorithms for planning under uncertainty in environment maps. The key idea of our approach is to evaluate uncertainty, represented by collision probability bounds, at multiple resolutions in different regions of the configuration space, depending on their relevance for finding a best path. Preliminary experimental results show that our approach is highly effective: our BURM algorithm is at least 40 times faster than an algorithm that tries to evaluate collision probabilities exactly, and it is not much slower than classic probabilistic roadmap planning algorithms, which ignore uncertainty in environment maps.

1 Introduction

Probabilistic sampling is a highly successful approach for motion planning of robots with many degrees of freedom. Sampling-based motion planning algorithms typically assume that input environments are perfectly known in advance. This assumption is reasonable in carefully engineered settings, such as robot manipulators on manufacturing assembly lines. As robots venture into new application domains at homes or in offices, environment maps are often acquired through sensors subject to substantial noise. It is essential for sampling-based motion planning algorithms to take into account uncertainty in environment maps during planning so that the resulting motion plans are relevant and reliable.

Leonidas J. Guibas
Stanford University, Stanford, CA 94305, USA
e-mail: guibas@cs.stanford.edu

David Hsu, Hanna Kurniawati, and Ehsan Rehman
National University of Singapore, Singapore 117417, Singapore
e-mail: {dyhsu,hannakur,ehsanreh}@comp.nus.edu.sg

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 199–215.
springerlink.com

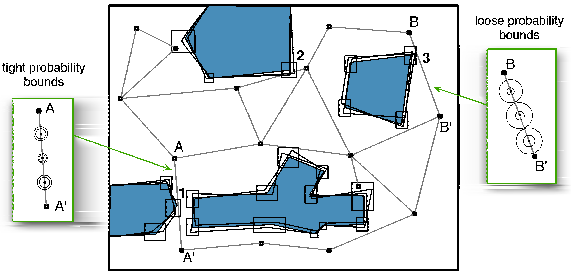


Fig. 1. An uncertainty roadmap. The positions of polygon vertices are uncertain and modeled as probability distributions. The rectangular boxes around the vertices indicate the regions of uncertainty. In the insets, for each configuration marked along a roadmap edge, there are two circles indicating the upper and lower bounds on the probability that the configuration is collision-free. The size of the circles indicates the probability value. In region 1, the two circles have similar size, indicating that the probability bounds are tight.

In sampling-based motion planning, a robot's configuration space $\mathscr{C}$ is assumed to be known and represented implicitly by a geometric primitive $\texttt{Free}(q)$, which returns true if and only if the robot placed at q does not collide with obstacles in the environment. The main idea is to capture the connectivity of $\mathscr{C}$ in a graph, usually called a *roadmap*. The nodes of the roadmap correspond to collision-free configurations sampled randomly from $\mathscr{C}$ according to a suitable probability distribution. There is an edge between two nodes if the straight-line path between them is collision-free. Now suppose that the shapes or poses of obstacles in the environment are not known exactly, but are modeled as a probability distribution $\pi_{\mathscr{C}}$ of possible shapes and poses. Then $\texttt{Free}(q)$ cannot always determine whether q is collision-free or not: it depends on the distribution of obstacle poses and shapes. Instead of relying on $\texttt{Free}(q)$, we need to compute the probability that q is collision-free with respect to $\pi_{\mathscr{C}}$, and instead of a usual roadmap, we construct an *uncertainty roadmap* U by annotating roadmap edges with probabilities that they are collision-free (Fig. 1). We then process path planning queries by finding a path in U that is best in the sense of low collision probability or other suitable criteria that take into account, *e.g.*, path length as well.

Unfortunately, constructing a complete uncertainty roadmap U incurs high computational cost, as computing collision probabilities exactly is very expensive computationally. It effectively requires integrating over a high-dimensional distribution $\pi_{\mathscr{C}}$ of obstacle shapes and poses. The *dimensionality* of $\pi_{\mathscr{C}}$ depends on the geometric complexity of the environment obstacles. Consider, for example, a simple two-dimensional environment consisting of 10 line segments. Each line segment is specified by its endpoints, whose positions are uncertain. We then need to integrate over a distribution of $10 \times 2 \times 2 = 40$ dimensions!

To overcome this difficulty, we turn the high-dimensional integral into a series of lower-dimensional ones. More interestingly, observe that a path planning query may be answered without knowing the complete uncertainty roadmap with exact

collision probabilities. We maintain upper and lower bounds on the probabilities and refine the bounds incrementally as needed. We call such a roadmap a *bounded uncertainty roadmap* (BURM). See the insets in Fig. 1 for an illustration. The key idea of our approach is to evaluate uncertainty, represented by the collision probability bounds, at multiple resolutions in different regions of the configuration space, depending on their relevance for finding a best path in U. Often, the critical decision of favoring one path over another depends on the uncertainty in localized regions only, for example, in narrows passages where the robot must operate in close proximity of the obstacles. It is thus sufficient to evaluate uncertainty accurately only in those regions and only to the extent necessary to choose a best path. Consider the example in Fig. 1. If we want to go form A to A', it is important to evaluate the precise collision probabilities in region 1 in order to decide whether to take the risk of going through the narrow passage or to make a detour. Knowing the precise collision probabilities in regions 2 and 3 is much less relevant for this decision. Thus, evaluating collision probabilities at different resolutions hierarchically leads to drastic reduction in computation time by avoiding unnecessarily computing the exact collision probabilities.

In the following, we start by briefly reviewing related work (Section 2). We then introduce the notion of a BURM (Section 3) and describe our approach for path planning with BURMs (Section 4). Preliminary experimental results show that our approach for path planing under uncertainty is highly effective: in our tests, it is at least 40 times faster than an algorithm that tries to evaluate collision probabilities exactly, and it is not much slower than classic probabilistic roadmap planning algorithms, which ignore environment uncertainty (Section 5). Finally, we conclude with directions for future work (Section 6).

2 Related Work

Motion planning under uncertainty is an important problem in robotics and has been studied widely [13, 14, 22]. In robot motion planning, uncertainty arises from two main sources: (i) noise in robot control and sensing and (ii) imperfect knowledge of the environment. In this work, we address the second only. Partially observable Markov decision processes (POMDPs) is a general and principled approach for planning under uncertainty [20, 10]. Unfortunately, under standard assumptions, the computational cost of solving a POMDP exactly is exponential the number of states of the POMDP [18]. An uncertain environment usually generates a large number of states. Despite the recent advances in approximate POMDP solvers [19, 21, 12], uncertain environments still pose a significant challenge for POMDP planning. In mobile robot motion planning, a common representation of an uncertain environment is an occupancy grid. Each cell of an occupancy grid contains the probability that the cell is occupied by obstacles. Assuming that uncertainty in robot control and sensing is negligible, one can find a path with minimum expected collision cost by graph search algorithms, such as Dijkstra's algorithm or the A* algorithm. In the motion planning literature, occupancy-grid planning belongs to the class of

approximate cell decomposition algorithms [13], whose main disadvantage is that they do not scale up well as a robot's number of degrees of freedom increases.

Probabilistic sampling of the robot's configuration space is the most successful approach for overcoming this scalability issue [5, 8, 14]. Our work is based on this approach, but extends it to deal with uncertainty in environment maps. Usually, sampling-based motion planning algorithm first build a roadmap that approximates the connectivity of the robot's configuration space and then search for a collision-free path in the roadmap. The idea that a path planning query can be answered without constructing the complete roadmap in advance is a form of lazy evaluation and has appeared before in sampling-based algorithms for single-query path planning, *e.g.*, Lazy-PRM [2], EST [9], and RRT [15]. However, since classic motion planning assumes perfect knowledge of the geometry of the robot and the obstacles, lazy construction of the roadmap is simpler.

Sampling-based motion planning has been extended to deal various types of uncertainty, including robot control errors [1], sensing errors [4, 23], and imperfect environment maps [17]. Our problem is related to that in [17]. To overcome the difficulty of computing collision probability, the earlier work proposes a nearest-point approximation technique. Although the approximation is supported by experimental evidence, its error is difficult to quantify. Also, the technique is restricted to two-dimensional environments only [17].

3 Bounded Uncertainty Roadmaps

Let us start with two-dimensional environments. The obstacles are modeled as polygonal objects, each consisting of a set of primitive geometric features—line segments for two-dimensional environments. The endpoints of the line segments are not known precisely and modeled as probability distributions with finite support, such as truncated Gaussians. See Fig. 1 for an illustration. Environment maps of this kind can be obtained by, for example, feature-based extended Kalman filtering (EKF) mapping algorithms [22]. For generality, we model the robot in exactly the same way. In three-dimensional environments, the representation is similar, but the primitive geometric feature are triangles rather than line segments.

Given such a representation of the obstacles and the robot, we can construct an uncertainty roadmap U in the robot's configuration space $\mathscr{C}$. The nodes of U are configurations sampled at random from $\mathscr{C}$. For every pair of nodes u and u' that are close enough according to some metric, there is an edge in U, representing the straight-line path between u and u'. Recall that in classic motion planning, sampling-based algorithms construct a roadmap whose nodes and edges are guaranteed to be collision-free, and the goal is to find a collision-free path in the roadmap. In our setting, due to the uncertainty, we cannot guarantee that the nodes and edges of U are collision-free, and there may exist no path that is collision-free with probability 1. So instead, we want to find a path with minimum cost according to a suitable cost function. A cost function may incorporate various properties of the desired path. To be specific, our cost function incorporates two considerations: the collision

probability and the path length. This allows us to trade off the distance that the robot must travel against the risk of collision.

To define such a path cost function, we assign a weight to each edge e of U:

$$W(e) = \ell(e) + \mathbf{E}[C(e)], \tag{1}$$

where $\ell(e)$ is the length of e and $C(e)$ is the cost of collision for e. $\mathbf{E}[C(e)]$ denotes the expected collision cost, and the expectation is taken over $\pi_{\mathscr{C}}$, the probability distribution of the obstacle and robot geometry. This cost function assumes that collision is tolerable and we want to trade off the risk of collision against the robot's travel distance. Paths that do not conform to this assumption, *e.g.*, those that penetrate through the interior of obstacles, must be excluded. To obtain $W(e)$, we need to calculate $\mathbf{E}[C(e)]$. Doing so directly is extremely difficult, because of the need to integrate over $\pi_{\mathscr{C}}$, a high-dimensional distribution whose dimensionality is proportional to the number of geometric features describing the obstacles and the robot, as illustrated by the example in Section 1. Furthermore, we must perform this integration for every edge of U. Instead of this, we break down the integration process into several steps. First, we integrate over the configurations contained in e:

$$C(e) = \int_{q \in e} C(q)\, dq,$$

where we slightly abuse the notation and use $C(q)$ to denote the collision cost at q. Following the usual practice in sampling-based motion planning, we discretize the edge e into a sequence of of configurations $(q_1, q_2, \ldots, q_n)$ at a fixed resolution and approximate the integral by

$$C(e) = \sum_{i=1}^{n} C(q_i). \tag{2}$$

Next, recall that the obstacles and the robot are each represented as a polygonal object consisting of a set of primitive geometric features. Denote the two feature sets by S for the obstacles and S' for the robot. We define the collision cost of the robot with the obstacles as the sum of collision costs of all pairs of geometric features $s \in S$ and $s' \in S'$. This can model, for example, the preference that configurations with fewer feature pairs in collision are more desirable. In formula, we have

$$C(q) = \sum_{s \in S, s' \in S'} C_{s,s'}(q), \tag{3}$$

where $C_{s,s'}(q)$ is the collision cost for the feature pair s and s' when the robot is placed at configuration q. Combining Eqs. (1–3) and using the linearity of expectation, we get $W(e) = \ell(e) + \sum_{i=1}^{n} \sum_{s \in S, s' \in S'} \mathbf{E}[C_{s,s'}(q_i)]$. Let $I_{s,s'(q)}$ denote the event that s and s' intersect when the robot is placed at q. Then $\mathbf{E}[C_{s,s'}(q_i)] = \alpha \mathrm{P}(I_{s,s'(q)})$, where α is the cost of collision when a pair of features intersect. The value of α is usually constant for two-dimensional environments, but may vary according to s and s' for three-dimensional environments. In practice, α is adjusted to reflect our willingness to take the risk of collision in order to shorten the robot's travel distance. To summarize, the weight of an edge e is given by

$$W(e) = \ell(e) + \sum_{i=1}^{n} \sum_{s \in S, s' \in S'} \alpha \, \mathrm{P}(I_{s,s'(q_i)}), \tag{4}$$

and the cost of a path γ in U is the sum of the weights of all edges contained in γ.

To compute the cost of a path, each edge of an uncertainty roadmap U must carry a set of probabilities $\mathrm{P}(I_{s,s'(q_i)})$. Although s and s' are primitive geometric features of constant size, computing $\mathrm{P}(I_{s,s'(q_i)})$ exactly is still expensive. If s and s' are both uncertain, then the computation requires integration over a distribution of 8 dimensions for two-dimensional features and 18 dimensions for three-dimensional features. To reduce the computational cost, we maintain upper and lower bounds on $\mathrm{P}(I_{s,s'(q_i)})$ rather than calculate the exact probability. Thus each edge e of U carries a set of probability bounds on $\mathrm{P}(I_{s,s'(q_i)})$ for each configuration $q_i \in e$ resulting from the discretization of e and for each pair of features $s \in S$ and $s' \in S'$. We call such a roadmap a *bounded uncertainty roadmap* or BURM for short. The probability bounds are refined incrementally by subdividing the integration domain hierarchically during the path finding.

4 Path Planning with BURMs

4.1 Overview

Suppose that we are given the (uncertain) geometry of the obstacles and the robot in the representation described in the previous section. Our goal is to find a minimum-cost path between a start configuration q_s and a goal configuration q_g. Conceptually, there are two steps, First, we construct a BURM U with trivial probability bounds by sampling the robot's configuration space $\mathscr{C}$. Next, we tighten up the probability bounds incrementally while searching for a minimum-cost path in U.

The first step is similar to that in the usual sampling-based motion planning algorithms. We sample a set of configurations from $\mathscr{C}$ according to a suitable probability distribution and insert the sampled configurations along with q_s and q_g as nodes of U. We then create an edge for every pair of nodes that are sufficiently close according to some metric. We filter out those nodes and edges that are in collision. Here collision is defined with respect to the mean geometry of the obstacles and the robot, which means that the primitive geometric features representing the obstacles and the robot are all at their mean positions. The purpose of filtering is to exclude those paths that cause the robot to pass through the interior of the obstacles. It is well known that the probability distribution for sampling $\mathscr{C}$ is crucial, and there is a lot of work on effective sampling strategies for motion planning. See [5, 8, 14] for comprehensive surveys. There is also recent work on how to adapt the sampling distribution when the environment map is uncertain. In this paper, we do not address the issue of sampling strategies. BURMs can be used in combination with any of the existing sampling strategies.

In the second step, we search for a minimum-cost path in U using a variant of Dijkstra's algorithm. While Dijkstra's algorithm deals with path cost, a BURM contains only bounds on path cost. When there are two alternative paths, we may not be

Algorithm 1. Searching for a minimum-cost path in a BURM.

1: For every node u of a BURM U, initialize the lower and upper bounds on the cost of the minimum-cost path from q_s to u: $\underline{K}(u) = 0, \overline{K}(u) = 0$ if $u = q_s$, and $\underline{K}(u) = +\infty, \overline{K}(u) = +\infty$ otherwise.
2: Insert all nodes of U into a priority queue Q_u.
3: **while** Q_u is not empty **do**
4: Find in Q_u a node u such that $\overline{K}(u) \leq \underline{K}(v)$ for all $v \in Q_u$. Remove u from Q_u.
5: **if** u = q_g **then return**.
6: **for** every node v incident to u **do**
7: Discretize the edge between u and v at a given resolution into a sequence of configurations $q_i, i = 1, 2, \ldots$.
8: For every q_i, invoke `FindColEvents`(q_i, S, S') to find feature pairs $s \in S$ and $s' \in S'$ that are likely to have $\mathrm{P}(I_{s,s'(q_i)}) > 0$. For each such feature pair, set $\underline{\mathrm{P}}(I_{s,s'(q_i)}) = 0$ and $\overline{\mathrm{P}}(I_{s,s'(q_i)}) = 1$, and insert $I_{s,s'(q_i)}$ into Q_e.
9: Set $\underline{K}_u(v) = \underline{K}(u) + \underline{W}(u,v)$ and $\overline{K}_u(v) = \overline{K}(u) + \overline{W}(u,v)$.
10: **while** the two intervals $(\underline{K}_u(v), \overline{K}_u(v))$ and $(\underline{K}(v), \overline{K}(v))$ overlap **do**
11: `RefineProbBounds`(Q_e, U, q_s, u, v).
12: **end while**
13: **if** $\overline{K}_u(v) < \underline{K}(v)$ **then**
14: Set $\underline{K}(v) = \underline{K}_u(v)$ and $\overline{K}(v) = \overline{K}_u(v)$.
15: Update Q_u, using the new bounds on the cost of the minimum-cost path to v. Call `RefineProbBounds` if needed.
16: **end if**
17: **end for**
18: **end while**

able to decide which one is better, as their bounds may "overlap". To resolve this, we need to refine the probability bounds in a suitable way. The details are described in the next three subsections.

4.2 Searching for a Minimum-Cost Path

Given a BURM U, we search for a minimum-cost path in U using a variant of Dijkstra's algorithm. A sketch of the algorithm is shown in Algorithm 1. For each node u in U, we maintain the lower bound $\underline{K}(u)$ and upper bound $\overline{K}(u)$ on the minimum-cost path from q_s to u. Recall that every edge e of U carries a set of probability bounds on $I_{s,s'(q_i)}$, for every $q_i \in e$ resulting from the discretization of e and every feature pair $s \in S$ and $s' \in S'$. Let $\underline{\mathrm{P}}(I_{s,s'(q_i)})$ and $\overline{\mathrm{P}}(I_{s,s'(q_i)})$ denote the lower and upper bounds on the probability of $I_{s,s'(q_i)}$, respectively. Using these bounds, we can calculate the lower bound $\underline{W}(e)$ and upper bound $\overline{W}(e)$ on the edge weight for each edge $e \in U$. By the definition, $\underline{K}(u)$ and $\overline{K}(u)$ can then be obtained by summing up the bounds on the edge weights. Like Dijkstra's algorithm, we insert each node u of U into a priority queue Q_u, which is implemented as a heap, and then

dequeue them one by one until a minimum-cost path to q_g is found. However, instead of using the path cost from q_s to u as the priority value, we use the path cost bounds $(\underline{K}(u), \overline{K}(u))$. For two nodes u and v in U, We say that u has higher priority than v, if $\overline{K}(u) \leq \underline{K}(v)$. In our implementation of Q_u, if the bound intervals $(\underline{K}(u), \overline{K}(u))$ and $(\underline{K}(v), \overline{K}(v))$ overlap, we refine the probability bounds until we can establish which node has higher priority in Q_u.

As we have mentioned in Section 3, computing collision probabilities requires integration over a high-dimensional distribution and is very expensive computationally. We use two techniques for efficient computation of the probability bounds. In line 8 of Algorithm 1, `FindColEvents` find feature pairs $s \in S$ and $s' \in S'$ such that s and s' are likely to intersect with non-zero probability, when the robot is placed at q_i. For each such feature pair, we attach an initial probability bound of $[0,1]$ to the event $I_{s,s'(q_i)}$ and insert $I_{s,s'(q_i)}$ into a set Q_e as a candidate for probability bound refinement in the future. Observe that usually, at each configuration, only a small number of feature pairs are in close proximity and likely to intersect with non-zero probability. Therefore, this step drastically reduce the number of collision probability bounds that need to be calculated. To search for the intersecting feature pairs efficiently, we exploit a hierarchical representation of the geometry of the obstacles and the robot (see Section 4.3) and quickly eliminate most of the feature pairs that are guaranteed to have zero collision probability.

In lines 11 and 15 of Algorithm 1, `RefineProbBounds` refines probability bounds. While searching for a minimum-cost path in U, we may encounter two paths γ and γ' and must decide which one has lower cost. If the bound intervals on the cost of γ and γ' overlap, refinement of the probability bounds becomes necessary. To do so, we find all events $I_{s,s'(q)}$ in Q_e such that q lies in an edge along γ or γ'. We then refine the probability bounds on these events (see Section 4.4), until we can determine the path with lower cost. For efficiency, we order the events found with a heuristic and process those more likely to separate the bound intervals first.

To focus on the main issue and keep the presentation simple, Algorithm 1 uses Dijkstra's algorithm for graph search. Informed search, such as the A* algorithm with an admissible heuristic function, is likely to give better results. In our case, one possible heuristic function is the Euclidean distance between two configurations.

4.3 Bounding Volume Hierarchies

In this and next subsections, we describe our computation of probability bounds. We restrict ourselves to the two-dimensional case. The basic idea generalizes to the three-dimensional case in a straightforward way, but the details are more involved.

For the two-dimensional case, `FindColEvents` (Algorithm 1, line 8) finds line segment pairs $s \in S$ and $s' \in S'$ that are likely to have intersection probability $\mathrm{P}(I_{s,s'(q)}) > 0$, when s' is placed at configuration q. It does so by quickly eliminating most of the line segment pairs with $\mathrm{P}(I_{s,s'(q)}) = 0$. Recall that we model the endpoints of these line segments as probability distributions with finite support.

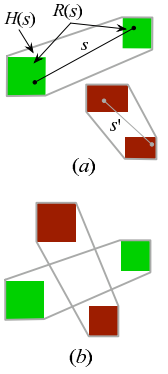


Fig. 2. The line segment pair s and s' intersect with (*a*) probability 0 and (*b*) probability 1.

Without loss of generality, assume that the support regions are rectangular. Let $R(s)$ denote the endpoint regions for a line segment s and $H(s)$ denote the convex hull of $R(s)$. Using the result below, we can check whether a line segment pair s and s' has $\mathrm{P}(I_{s,s'(q)}) = 0$ in constant time, as the convex hulls and the endpoint regions of s and s' are all polygons with a constant number of edges. See Fig. 2 for an illustration.

Theorem 4.1. *Let s and s' denote two line segments with uncertain endpoint positions in two dimensions.*

(1)*If $H(s) \cap H(s') = \emptyset$, then s and s' intersect with probability* 0.
(2)*If $H(s) \cap H(s') \neq \emptyset$, $R(s) \cap H(s') = \emptyset$, and $R(s') \cap H(s) = \emptyset$, then s and s' intersect with probability* 1.

If S and S' contain m and n line segments, respectively, it takes $O(mn)$ time to check all line segment pairs $s \in S$ and $s' \in S'$. This is very time-consuming, as we need to invoke `FindColEvents` repeatedly at many configurations. To improve efficiency, we apply a well-known technique from the collision detection literature and build bounding volume hierarchies over the geometry of the obstacles and the robot. There are many different types of bounding volume hierarchies. See [16] for a survey. We have chosen the sphere tree hierarchy, though other hierarchies, such as the oriented bounding box (OBB) tree, can be used as well. Specifically, we build two sphere trees for S' and S, respectively. Each leaf of a sphere tree contains the convex hull $H(s)$ for a line segment s, and each internal node v contains a sphere that encloses the geometric objects in the children of v (Fig. 3). Clearly $H(s)$ and $H(s')$ can intersect only if their enclosing sphere intersect. It then follows from Theorem 4.1 that s and s' intersect with non-zero probability only if the spheres enclosing $H(s)$ and $H(s')$ intersect. By traversing the sphere trees hierarchically, we can quickly eliminate most of the line segment pairs that have zero intersection probability and reduce the cost of checking a quadratic number of line segment pairs to a much smaller number. We omit the details of constructing sphere tree hierarchies and traversing them for collision detection, as they are well documented elsewhere (see, *e.g.*, [16]).

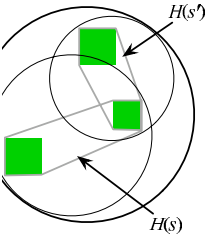


Fig. 3. A sphere tree over two uncertain line segments.

In summary, by exploiting a hierarchical representation, `FindColEvents` efficiently identifies most line segment pairs with intersection probability 0 and reduce the trivial probability bound of $[0,1]$ to $[0,0]$ for all of them together. For the remaining line segment pairs, which are usually small in number, their probability bounds are further refined when necessary (see next subsection).

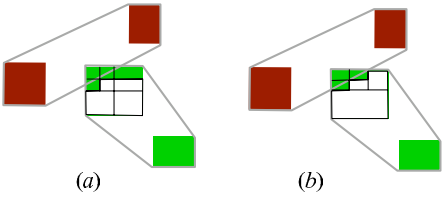


Fig. 4. Decomposing the integration domain for intersection probability calculation. (*a*) The quadtree-based procedure. (*b*) Our procedure that takes into account the geometry of intersecting line segments. The example shows that after roughly a same number of cuts, our procedure identifies a large part of the domain for which no further decomposition is needed.

4.4 Hierarchical Refinement of Collision Probability Bounds

We now consider the problem of refining the bounds on the intersection probability $\mathrm{P}(I_{s,s'(q)})$ for some line segment pair $s \in S$ and $s' \in S'$. To simplify the notation, we will omit the parameter q and assume that s' is translated and rotated suitably.

Computing $\mathrm{P}(I_{s,s'})$ is in essence an integration problem. Let x_1 and x_2 be the endpoints of s, and let x_3 and x_4 be the endpoints of s'. Suppose that x_i has probability density function $f_i(x_i)$ with rectangular support regions R_i. We can calculate the probability that s and s' intersect by integrating over $R_1 \times \cdots \times R_4$:

$$\mathrm{P}(I_{s,s'}) = \int_{R_1 \times \cdots \times R_4} A(x_1, \ldots, x_4)\, f_1(x_1) dx_1 \cdots f_4(x_4) dx_4, \tag{5}$$

where $A(x_1, \ldots, x_4)$ is an index function that is 1 if and only if s and s' intersect. In two-dimensional environments, this integral is 8-dimensional.

To evaluate this integral, we decompose the integration domain $R_1 \times \cdots \times R_4$ hierarchically into a set of subdomains such that in each subdomain, the index function A is constant. By summing up the probability mass associated with all the subdomains where A is 1, we get the value for $\mathrm{P}(I_{s,s'})$. During the hierarchical decomposition process, we maintain three lists of subdomains: (i) subdomains where A is always 1, (ii) subdomains where A is always 0, and (iii) subdomains where A has mixed values (0 or 1). Interestingly, these three lists provide an upper bound and a lower bound on $\mathrm{P}(I_{s,s'})$ at any moment during the decomposition process. Let p_1 and p_2 be the probability mass associated with subdomains in list (i) and (ii), respectively. Clearly, we have $p_1 \leq \mathrm{P}(I_{s,s'}) \leq 1 - p_2$. The probability mass associated with subdomains in list (iii) is $1 - p_1 - p_2$. It represents the gap between the upper and lower bounds. To refine the bounds, we simply take a subdomain from list (iii) and decompose it further until some of the refined subdomains can be assigned to either list (i) or (ii).

To decompose an integration domain $R_1 \times \cdots \times R_4$, we take a horizontal or vertical cut on one or more of the endpoint regions R_i and obtain a set of subdomains $R'_1 \times \cdots \times R'_4$ such that R'_i is rectangular and $R'_i \subseteq R_i$ for $i = 1,2,3,4$. Using Theorem 4.1, we can easily determine whether the index function A has constant value over a subdomain and assign the subdomain to the appropriate list.

There are various strategies to decompose a rectangular integration domain. The main goal is to assign each subdomain to list (i) or (ii) and avoid unnecessarily decomposing into a large number of tiny subdomains. One possible decomposition procedure, based on the quadtree [6], always cuts an endpoint region in the middle either horizontally or vertically (Fig. 4*a*). This is simple to implement, but does not always results in the best decomposition, as it may unnecessarily cut a domain into small pieces. Our decomposition procedure uses the geometry of the two intersecting line segments s and s' to decide where to cut. This results in better decomposition, but the trade-off is that each decomposition step is slightly more expensive. To determine how to cut, our procedure enumerates several cases that depend on the relative positions of the endpoint regions and convex hulls for s and s'. The details are not particularly important. An example decomposition is shown in Fig. 4*b* for illustration. In the experiments, our decomposition procedure usually gives slightly better performance than the quadtree-based procedure.

An alternative way of evaluating the integral in (5) is to perform Monte Carlo integration [11] by sampling from the integration domain $R_1 \times \cdots \times R_4$. Each sample consists of four points $x_1, \ldots, x_4$ with $x_i \in R_i$. Let A_i be the value of the index function A for the ith sample, and p be the value of the integral in (5). Then an estimate of p is given by

$$p_N = \frac{1}{N} \sum_{i=1}^{N} A_i. \tag{6}$$

The values $A_i, i = 1, 2, \ldots, N$ are in fact a set of independent and identically distributed (i.i.d.) random variables. Under a wide range of sampling distributions, the mean of A_i is equal to p. Let V_A be the variance of A_i. By (6), p_N is a random variable with mean p and variance V_A/N. We can then apply Chebychev's inequality and obtain

$$\mathrm{P}\left(|p_N - p)| \geq (1/\delta)(V_A/N)^{-1/2} \right) \leq \delta, \tag{7}$$

which implies that p_N converges to p at the rate $O(N^{-1/2})$. More precisely, for any δ arbitrarily small, we can determine the number of samples, N, needed to ensure that the estimate p_N does not deviate too much from p. So instead of maintaining upper and lower bounds on the collision probabilities, we can choose N large enough to get sufficiently accurate estimates for all the collision probabilities and find a minimum-cost path with high probability. Unfortunately, using Monte Carlo integration this way is not efficient (see Section 5), as it uses the same number of samples for estimation everywhere over the entire environment.

An interesting method is to combine probability bound refinement and Monte Carlo integration. We start by decomposing the integration domain as described earlier. When the probability mass associated with a subdomain is small enough, we apply the Monte Carlo method with a small number of samples to get an estimate and close the gap between the upper and lower bounds. Strictly speaking, if we do this, we cannot guarantee that the algorithm finds a minimum-cost path in U. However, if we use a sufficient number of samples for Monte Carlo integration, we can provide the guarantee with high probability. Furthermore, even when the algorithm fails to find a minimum-cost path, the cost of the resulting path is still a

good approximation to the minimum cost. The reason is that due to the bound in (7), we make a mistake only when two paths have very similar cost. We use this combined method in our implementation of the algorithm, and it achieves better performance better than one that uses pure probability bound refinement.

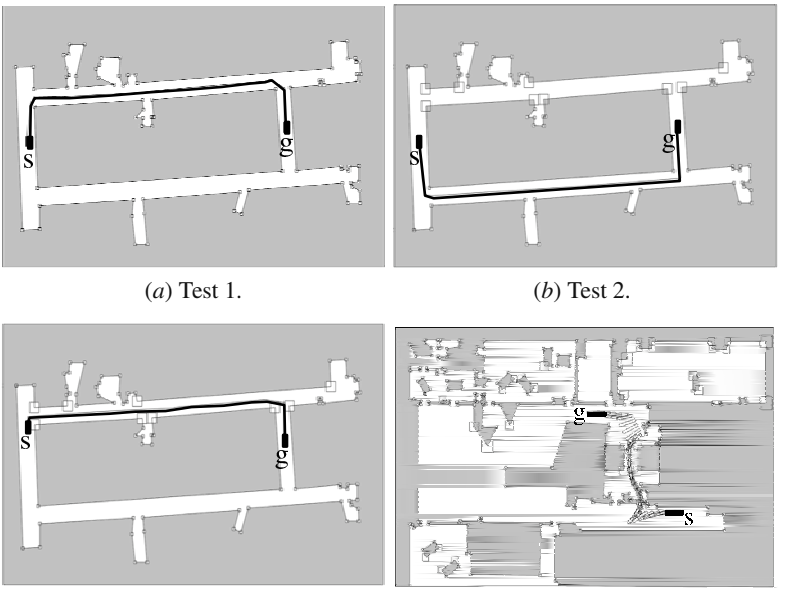
5 Experiments

We made a preliminary implementation of our algorithm and compared it with two alternatives. One algorithm, which we developed for the purpose of comparison, is similar to BURM. It also builds an uncertainty roadmap. However, instead of refining the probability bounds incrementally when necessary, it estimates the exact probabilities using Monte Carlo integration. We call this algorithm MCURM. In our tests, MCURM uses 100 samples to evaluate each intersection probability $\mathrm{P}(I_{s,s(q)})$. The other algorithm that we compared is Lazy-PRM [2], which does not take into account uncertainty during planning.

In our tests, all three algorithms use the same sampling strategy, which is a hybrid strategy consisting of the bridge test and the uniform sampler [7]. We ran the algorithms on each test case and repeated 30 times independently. The performance statistics reported here are the averages of 30 runs. In each run, the three algorithms used the same set of sampled configurations. So the performance difference results from the way they find a minimum-cost path in an (uncertainty) roadmap rather than random variations in sampling the configuration space.

The test results are shown in Fig. 5 and Table 1. In tests 1–3, the robot has a rectangular shape and only translates. In these three tests, the environments are similar. The robot essentially chooses between two corridors to go from the start to the goal position. The main differences among the tests are (i) the level of uncertainty in the obstacle geometry and (ii) the robot start position. In test 1 (Fig. 5*a*), the uncertainty is low and roughly the same everywhere. So the robot chooses the upper corridor, based mainly on the path length consideration. However, it is interesting to observe that although a shortest path with respect to the path length normally touches obstacle boundaries, our minimal-cost path stays roughly in the middle of the corridor. It does so to avoid collision due to the uncertainty in obstacle geometry. In test 2, the upper corridor has substantially higher uncertainty than the lower corridor. On balance, it is better for the robot to choose the slightly longer, but safer lower corridor (Fig. 5*b*). In test 3, the uncertainty in obstacle geometry remains the same as that in test 2, but the start position for the robot moves higher. The robot again decides to go through the upper corridor, because despite the higher collision risk of the upper corridor, it is much shorter than the lower corridor (Fig. 5*c*).

In test 4, the robot can both translate and rotate. To reach its goal, the robot can either take the risk of collision and squeeze through the narrow passage or make a long detour. It is not obvious which choice is better. The answer depends, of course, on the cost of collision. In this case, the robot decides to take the riskier, but shorter path (Fig. 5*d*). Interestingly, our algorithm finds two paths of similar cost, depending on the set of sampled configurations. One path veers to the right (Fig. 5*d*) when it



(*a*) Test 1. (*b*) Test 2.

(*c*) Test 3. (*d*) Test 4

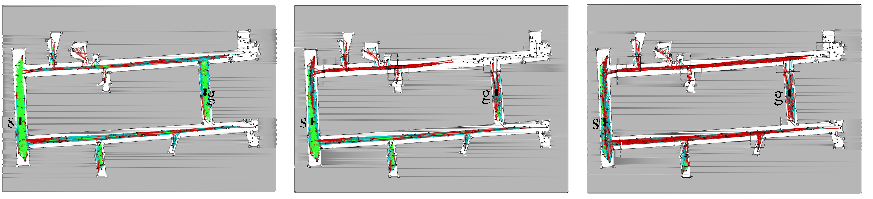
Fig. 5. Test environments and results. The boxes around the vertices mark the support regions of the probability distributions modeling the endpoint positions of line segments forming the obstacle boundaries.

approaches the obstacle near the lower entrance to the narrow passage, and the other veers to the left. This is in fact not surprising, because regardless of whether the path veers to the the left or right, the uncertainty that the path encounters remains similar and the path length does not differ by much.

Now let us look at the performance statistics. For each test case, Table 1 lists the number of nodes in the (uncertainty) roadmap, the running times, and the cost of the paths found by the three algorithms. Note that the absolute running times reported in Table 1 are a little slow, as our implementation is still preliminary and does not optimize the speed of important primitive geometric operations such as the intersection test. We plan to improve the implementation in the future. However, this

Table 1. Performance statistics.

Test Env.	No. Nodes	Cost			Time (s)		
		BURM	MCURM	Lazy-PRM	BURM	MCURM	Lazy-PRM
1	300	699	699	789	59	3,119	42
2	300	741	741	1,059	72	2,871	42
3	300	761	760	891	74	2,994	35
4	500	526	526	668	61	2,534	36



(*a*) (*b*) (*c*)

Fig. 6. Color-coded BURMs. Red, gray, and blue-green marks edges with tight, intermediate, and loose collision probability bounds, respectively. Bright green marks edges with collision probability 0.

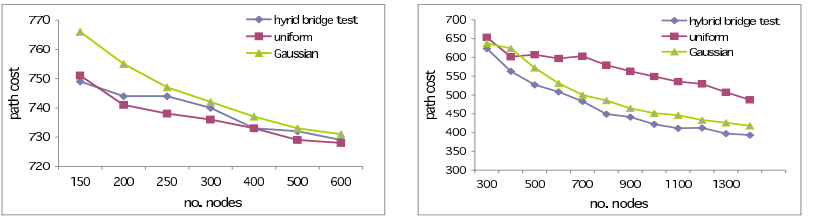
does not significantly affect the comparison of the three algorithms based on their relative performance, as they use the same implementation of primitive operations.

BURM and MCURM find paths with almost the same cost. BURM is, however, 40–50 times faster. This clearly demonstrates the advantage of the approach of evaluating uncertainty hierarchically at multiple resolutions.

The comparison between BURM and Lazy-PRM is even more interesting. As expected, BURM finds paths with lower cost, as it takes uncertainty into account during planning. However, it is somewhat surprising that BURM is not much slower than Lazy-PRM: BURM is only 2 times slower than Lazy-PRM, while it is at least 40 faster than MCURM. The reason is that uncertainty comes into play in deciding the best path when the robot operates in close proximity of the obstacles. This usually happens in localized regions of the configuration space only. BURM takes advantage of this by maintaining bounds on collision probabilities rather than calculating the exact probabilities. It refines these bounds incrementally by exploiting bounding volume hierarchies on the geometry of the obstacles and the robot and by hierarchically decomposing the integration domain for collision probability calculation. The running time comparison with Lazy-PRM provides further evidence on the advantage of our approach.

To better understand the behavior of BURM, we applied it to an environment similar to that in tests 1–3 and varied the uncertainty level in the obstacle geometry. The resulting bounded uncertainty roadmaps are shown in Fig. 6. The edges of the roadmaps are colored to indicate how tight the associated collision probability bounds are. The roadmap in Fig. 6*a* serves as a reference point for comparison. The uncertainty is low in both the upper and lower corridors, and the collision probability bounds are refined to various degrees. As the uncertainty gets higher in the upper corridor, the collision probability bounds there are tightened to differentiate the quality of the paths (Fig. 6*b*). It is also interesting to observe that the upper corridor is not explored as much, because the paths in the lower corridor are far better. Finally, as the uncertainty in the lower corridor also increases, both corridors must be explored, and the collision probability bounds carefully tightened in order to determine the best path (Fig. 6*c*).

Bounded uncertainty roadmaps can be used with any existing sampling strategies. To demonstrate this, we performed additional tests by varying the sampling



(*a*) (*b*)

Fig. 7. The change in the cost of the minimum-cost path found as a function of the sampling strategy and the number of nodes in the uncertainty roadmap.

strategy used and the number of nodes in the uncertainty roadmap. We tried two additional strategies: the uniform sampler and the Gaussian sampler [3]. For all sampling strategies, as the roadmap size increases, the running time increases correspondingly. Two plots of representative results are shown in Fig. 7. They indicate that the cost of the minimum-cost path found decreases with the roadmap size up to a certain point and then stabilizes. So one way of minimizing the path cost is to run our algorithm in an "anytime" fashion by gradually adding more nodes to the roadmap. The effect of different sampling strategies is more pronounced in complex environments. In the simpler environment (see Fig. 5*b*), when the number of roadmap nodes is sufficiently large, the results obtained by the different sampling strategies are comparable. When the number of nodes is small, the Gaussian sampler does not behave very well, as it biases sampling towards the obstacle boundaries, resulting in high collision cost. In the more complex environment (see Fig. 5*d*), which contains several narrow passages, the hybrid bridge test and the Gaussian sampler have clear advantages over the uniform sampler. Just as in classic motion planning, effective sampling strategies for constructing uncertainty roadmaps are important and require further investigation.

6 Conclusion

We have introduced the notion of a bounded uncertainty roadmap and used it to extend sampling-based algorithms for planning under uncertainty in environment maps. By evaluating uncertainty hierarchically at multiple resolutions in different regions of a robot's configuration space, our approach greatly improves planning efficiency. Experimental results, based on a preliminary implementation of our planning algorithm, demonstrate that it is highly effective.

There are many interesting directions for future work. The main idea of our approach, evaluating uncertainty hierarchically at multiple resolutions, is not restricted to the particular path cost function used here. For example, our current path cost function sums up the collision costs for the configurations along a path. Sometimes it may be more suitable to take the maximum of rather than sum up the collision costs. Our idea can be applied to this new path cost function with small modifications. We would also like to understand the effect of sampling strategies on the

running time and the quality of the result for our algorithm with respect to particular classes of path cost functions. One idea is to sample a robot's configuration space adaptively and adjust the sampling distribution based on the collision probabilities of previously sampled configurations. Finally, we will implement our algorithm and test it in three-dimensional environments.

Acknowledgements. This work is supported in part by MoE AcRF grant R-252-000-327-112 and NSF grants CCF-0634803 and FRG-0354543.

Appendix 1 Proof of Theorem 1

Proof. To prove part (1), observe that since $H(s) \cap H(s') = \emptyset$, line segments s and s' has no intersection for all pairs s and s' whose endpoints lie in $R(s)$ and $R(s')$, respectively. This immediately implies that the intersection probability is 0.

Now consider part (2). We start with a simple observation. Let σ be a fixed line segment such that the endpoints of σ lie outside $H(\sigma')$ for some line segment σ' with uncertain endpoint positions and σ does not intersect with $R(\sigma')$. Then either σ intersects with σ' for *all* σ' whose endpoints lie in $R(\sigma')$, or σ intersects with *none* of them. Since $R(s) \cap H(s') = \emptyset$, the endpoints of s must lie outside of $H(s')$. Furthermore, s does not intersect with $R(s')$, as $R(s') \cap H(s) = \emptyset$. From our observation, it follows that s intersects with all s' with endpoints in $R(s')$ or intersects with none of them at all, and this is true for all s with endpoints lying in $R(s)$. Now, since $H(s) \cap H(s') \neq \emptyset$, there exists at least one pair of intersecting line segments s and s' with endpoints in $R(s)$ and $R(s')$. By the convexity of $R(s), R(s'), H(s)$ and $H(s')$, we conclude that all line segment pairs s and s' intersect. Thus the probability of intersection is 1. □

References

1. Alterovitz, R., Siméon, T.T., Goldberg, K.: The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In: Proc. Robotics: Science and Systems (2007)
2. Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 521–528 (2000)
3. Boor, V., Overmars, M.H., van der Stappen, F.: The Gaussian sampling strategy for probabilistic roadmap planners. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 1018–1023 (1999)
4. Burns, B., Brock, O.: Sampling-based motion planning with sensing uncertainty. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 3313–3318 (2007)
5. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations, ch. 7. The MIT Press, Cambridge (2005)
6. de Berg, M., van Kreveld, M., Overmars, M.H., Schwarzkopf, O.: Computaional Geometry: Algorithms and Applications. Springer, Heidelberg (2000)

7. Hsu, D., Jiang, T., Reif, J., Sun, Z.: The bridge test for sampling narrow passages with probabilistic roadmap planners. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 4420–4426 (2003)
8. Hsu, D., Latombe, J.C., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robotics Research 25(7), 627–643 (2006)
9. Hsu, D., Latombe, J.C., Motwani, R.: Path planning in expansive configuration spaces. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 2719–2726 (1997)
10. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)
11. Kalos, M.H., Whitlock, P.A.: Monte Carlo Methods, vol. 1. John Wiley & Sons, New York (1986)
12. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proc. Robotics: Science and Systems (2008)
13. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, Boston (1991)
14. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
15. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 473–479 (1999)
16. Lin, M., Manocha, D.: Collision and proximity queries. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, ch. 35. CRC Press, Boca Raton (2004)
17. Missiuro, P., Roy, N.: Adapting probabilistic roadmaps to handle uncertain maps. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 1261–1267 (2006)
18. Papadimitriou, C., Tsisiklis, J.N.: The complexity of Markov decision processes. Mathematics of Operations Research 12(3), 441–450 (1987)
19. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proc. Int. Jnt. Conf. on Artificial Intelligence, pp. 477–484 (2003)
20. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research 21, 1071–1088 (1973)
21. Smith, T., Simmons, R.: Point-based POMDP algorithms: Improved analysis and implementation. In: Proc. Uncertainty in Artificial Intelligence (2005)
22. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
23. Yu, Y., Gupta, K.: Sensor-based roadmaps for motion planning for articulated robots in unknown environments: Some experiments with an eye-in-hand system. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots & Systems, pp. 1070–1714 (1999)

A Sampling Hyperbelief Optimization Technique for Stochastic Systems

James C. Davidson and Seth A. Hutchinson

1 Introduction

Uncertainty plays a dramatic role not only on the quality of the optimal solution of POMDP system, but also on the computational complexity of finding the optimal solution, with a worst case running time that is exponential in the length of the time horizon for the exact solution. However, given the importance of finding optimal or nearly optimal solutions for systems subject to uncertainty, numerous researchers have developed approaches to approximate POMDP systems to overcome this limitation (refer to [22, Ch. 15 & 16] for a survey of such approaches). The majority of these methods are for discounted, infinite-horizon problems. Moreover, many of these techniques must reperform all the computational effort when the objective function changes.

A central theme of almost all approximation techniques is to reduce the set of possibilities to be evaluated, whether simplifying the representation of the belief or by the simplifying the value function. Drawing on insights offered in [8] about why belief sampling techniques (such as [21, 15, 17, 18, 19, 20]) are so effective, we develop an alternative method that is inspired by graph sampling-based methods (e.g., [10]). In [6], we introduce the notion of hyperfiltering, which evolves forward into future stages the probability function over the belief, or the hyperbelief. We refer to the space of probability functions over the belief as the hyperbelief space. Interestingly, the evolution of a system over the hyperbelief space is deterministic. Thus, we can find the optimal plan in the hyperbelief space using an approach derived from standard search techniques.

James C. Davidson
Beckman Institute, University of Illinois
e-mail: jcdavdsn@illinois.edu

Seth A. Hutchinson
Beckman Institute, University of Illinois
e-mail: seth@illinois.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 217–231.
springerlink.com

To do so, we abstract the problem into a two-level planner. At the high level, a set of points in the hyperbelief space is randomly generated. Edge weights between each ordered pair of samples are then generated using the lower level planner. The lower level planner commissions local, greedy feedback controllers to predict the evolution of the robot from one sampled hyperbelief point to another. Because of the stochastic, partially observed nature of the problem, hyperfiltering is used to estimate the future hyperbelief, under a given policy, from one stage to the next. Instead of requiring that each hyperbelief sample reach the target hyperbelief sample, the distance between the hyperbelief sample and the target sample is included with the edge information between each pair of samples.

At the cost of completeness, this graph representation reduces the set of possibilities to be explored from a continuum in an infinite number of dimensions to a finite set. Moreover, because the evolution is deterministic in the hyperbelief space, optimizing over the graph can be performed in worst case quadratic time complexity in the number of vertices in a complete graph using standard graph optimization algorithms–without the exponential explosion when planning in the belief space. Because the local policy may not attain the exact position of the target hyperbelief sample, a refinement algorithm is used to incrementally identify a possibly better policy and then simulate this policy to determine the actual cost and, thus, if the newly proposed policy is better. The bound on the optimal value can only decreases with every iteration, so that the refinement technique will find the best solution for a given graph in a finite amount of time.

One of the attributes of the proposed sampling-based hyperbelief optimization technique (SHOT) is the capability to generate an abstracted representation of the structure of the hyperbelief space that is independent of the cost function. This way, as with sampling-based methods in general, the majority of work can be performed offline. With relatively little computation expense, the nearly optimal policy for a variety of objectives can be determined based on the current status of the robot as well as changing the initial conditions such as the starting hyperbelief.

The proposed method will be developed in Section 4. However, first related research (Section 2) is explored and background concepts (Section 3) are introduced. Examples are provided in Section 5, and we conclude with some final remarks and comments in Section 6.

2 Related Research

Finding optimal policies for partially observable systems is intractable, with a best known computational time complexity that is exponential in both the time horizon and the number of observations [9]. Thus, many researchers have focused on finding efficient approximation methods (such as [21, 15, 17, 18, 19, 20]). These techniques approximate the POMDP optimization problem by simplifying the expression of the value function while performing value iteration. Each of these methods are generate a set of beliefs that are used to approximate the value function from one stage of the value backup to the next.

Many robotic motion planning approximation approaches are sampling-based as well. In fact sampling-based methods have become one of the dominant methods for planning in the robotics community (refer to [3, Ch. 7] for an overview and survey of sampling-based techniques). As described in [10], probabilistic sampling-based methods generate a series of samples in the configuration space and simple planners are used to link the samples (or vertices) together. In this way a roadmap (or graph) of the samples is created. The majority of sampling-based methods focus on finding feasible, but not necessarily optimal solutions. However, in [11], Kim et al. develop a technique to determine the optimal solution over the roadmap.

The concept of stochastic uncertainty in the process model was first introduced into sampling-based methods in [2]. This method created a discrete approximation of a continuous space by creating a roadmap. Then, probabilities of transitions from one node to another are assigned. By abstracting the problem this way, Apaydin et al. are able to reduce a continuous Markov decision process system to a finite Markov decision process. This method focuses on determining feasible solutions, while recently in [1] the concept of optimizing over the roadmap was introduced via the stochastic motion roadmap, whereby Alterovitz et al. sample a set of points in the configuration space to generate the roadmap. Next, for each node in the roadmap they generate a random set of resulting states for a given action. They then associate an edge weight between the node and any other node according to the number of times, out of the total, that the given action resulted in reaching the other node. Expanding to POMDPs, Prentice and Roy in [16] generate a sampling based approach for linear Gaussian systems. In their approach, a set of mean samples are generated corresponding to points in the configuration space. Next, a traditional probabilistic roadmap is used to generate a roadmap of the system (without consideration of the uncertainty). They generate the transfer function to the belief samples and generate their respective covariances. Next they use a standard graph search (e.g., A^*) to search the graph in a forward manner, while generating the best estimate of the actual covariance as the search progresses.

3 Background

3.1 POMDP Formulation

As a general model of stochastic systems, POMDPs incorporate the possibility of incomplete and uncertain knowledge when mapping states to observations. Such a representation enables the modeling and analysis of systems where sensing is limited and imperfect.

POMDPs include at least the following components: the state space $\mathscr{X}$ representing the finite set of states of the world; the finite set of control actions $\mathscr{U}$ that can be executed; the transition probability function $p_{\mathbf{x}_k|\mathbf{x}_{k-1},u_{k-1}}$ representing the likelihood of the system being in one state x_{k-1} and transferring into another state x_k at stage k given the applied action u_{k-1} at stage $k-1$; the set of all possible observations $\mathscr{Y}$; the observation probability function: $p_{\mathbf{y_k}|\mathbf{x}_k}$ describing the likelihood of a particular

observation y_k occurring given the system is in a specified state x_k; and the cost function $c(\cdot)$, which defines the objective to be optimized for the POMDP.

3.2 Hyperfiltering

Hyperfiltering is a method for systems modeled by POMDPs to propagate the estimate of the belief and its uncertainty forward into future stages for unseen observations and unactualized control inputs. By choosing the probability function over the beliefs, hyperfiltering is able to sequentially evaluate the estimate of the system and its uncertainty forward from one stage to the next. Moreover, by adopting the complete representation of the uncertainty, instead of a limited number of statistics of the belief, a more accurate representation of the evolution of the system is obtained.

The evolution of the probability function $p(x_k|I_k)$ at stage k, also known as the belief b_k, can be determined given the previous belief b_{k-1} and an applied control action $u_k \in \mathscr{U}$ via the belief transition function:

$$b_k = B(b_{k-1}, u_{k-1}, y_k),$$

where the belief transition function describes the Bayesian filtering over all states $x \in \mathscr{X}$, or

$$B(b_k, u_k, y_{k+1})(x_{k+1}) = \frac{p(y_{k+1}|x_{k+1}) \sum_{x_k \in \mathscr{X}} p(x_{k+1}|x_k, u_k) b_k(x_k)}{\sum_{x_{k+1} \in \mathscr{X}} p(y_{k+1}|x_{k+1}) \sum_{x_k \in \mathscr{X}} p(x_{k+1}|x_k, u_k) b_k(x_k)}. \quad (1)$$

The notation $B(\cdot)(x_{k+1})$ is adopted to represent the resulting functional evaluated at a specific state x_{k+1}. The belief at each stage k resides in the belief space $\mathscr{P}_b$, which is the space of all possible beliefs. For discrete state POMDPs, the belief space is represented as an $|\mathscr{X}| - 1$ dimensional simplex $\Delta^{|\mathscr{X}|-1}$, where $|\mathscr{X}|$ is the number of states in the state space.

When predicting future behavior, the observations are unknown and stochastic in nature. The future belief therefore becomes a random variable defined by the stochastic process:

$$\mathbf{b}_{k+1} = B(\mathbf{b}_k, u_k, \mathbf{y}_{k+1}). \quad (2)$$

The evolution from one stage to the next via the stochastic process (2) generates a random variable; thus, a representation of the probability function over the belief is needed to proceed.

The hyperbelief is a probability function over the belief space at each stage. The initial hyperbelief β_1 at stage $k = 1$ is given; for $k > 1$, the hyperbelief is defined as

$$\beta_k \triangleq p_{b_k|\beta_1,\pi}.$$

Each β_k is contained in the *hyperbelief space* $\mathscr{P}_\beta$.

The *belief transition probability function* $p(b_{k+1}|b_k, u_k)$, represents the probability of the outcome b_{k+1} of the stochastic process $B(b_k, u_k, \mathbf{y}_{k+1})$ given b_k and the applied control input u_k. Since both π and b_k are known, the probability function over

the observations can be inferred. The function that transfers a hyperbelief $\beta_k \in \mathscr{P}_\beta$ into the hyperbelief $\beta_{k+1} \in \mathscr{P}_\beta$ given a policy $\pi \in \Pi$ is denoted as the hyperbelief transition function Υ, such that $\Upsilon : \mathscr{P}_\beta \times \Pi \to \mathscr{P}_\beta$, where Π is the set of all information feedback policies. The hyperbelief transition function is represented as

$$\beta_{k+1} = \Upsilon(\beta_k, \pi)$$

where, for each $b_{k+1} \in \mathscr{P}_b$,

$$\Upsilon(\beta_k, \pi)(b_{k+1}) \triangleq \int_{b_k \in \mathscr{P}_b} p\left(b_{k+1} | b_k, \pi\left(b_k\right)\right) \beta_k(b_k) db_k.$$

The notation $\Upsilon(\cdot)(b_{k+1})$ is adopted to represent the resulting function evaluated at a specific belief $b_{k+1} \in \mathscr{P}_b$.

Due to the high nonlinearity of the hyperbelief transition function, we will approximate the hyperbelief transition function using the hyper-particle filter. When the belief transition probability function is chosen as the importance sampling function, the computational complexity of hyper-particle filtering is $O(Knq)$, where K is the desired time-horizon, n is the number samples representing the probability function over the belief space, and q is the number of samples representing a belief. Refer to [6] for a more thorough explanation of both hyperfiltering and the hyper-particle filtering approximation method.

4 Methodology

Our proposed method determines a nearly optimal policy for POMDP systems by approaching the problem from a two-tier, hierarchical approach. A digraph is generated where the vertices represent sampled hyperbeliefs and the edges represent paths, which are generated by using local feedback policies, between ordered pairs of the hyperbelief samples. Before we outline this process in more detail, we will introduce the class of cost functions and policies that we consider in this approach.

4.1 Cost Function and Policy Representation

Unlike typical approaches where the value function is the expected cost over the set of beliefs, we define the value function as a total sum of costs relative to the hyperbelief. We therefore can optimize the hyperbelief cost functions of the form $c : \mathscr{P}_\beta \times \Pi \to \mathbb{R}$, (where Π is the set of all feedback policies) and terminal cost $c_K : \mathscr{P}_\beta \to \mathbb{R}$. The value for a given initial hyperbelief β_0 is defined as

$$V(\beta_0) = \min_{\pi \in \Pi} \sum_{k=0}^{K-1} c(\beta_k, \pi) + c_K(\beta_K).$$

In this hierarchical approach, we restrict the lower level to be a local, greedy policy within some class of local feedback policies Π_l. A digraph $G =< N, E >$ is

constructed, where each vertex is associated with hyperbelief sample that is both the source and target for a set of greedy policies. The upper level policy $\gamma(\cdot)$ selects the local policy based which edge in G is being traversed and on the current stage to produce a closed-loop policy, such that $\gamma : \mathscr{P}_\beta \rightarrow E$. The complete policy is a switching-based controller (see [14] for an overview of hybrid/switching-based methods), whereby the local policy being executed is determined based on some switching surface. In our formulation the switching surface is defined as the inflection point of the derivative of the distance measurement from the current hyperbelief to the target hyperbelief as specified by the target of the current edge. At the point that the derivative becomes positive, the controller switches to the next edge specified, whereby the target in the previous edge is the source in the new edge.

The set of all possible switching policies is denoted as Γ. The value function for a given system becomes $V(\beta_0) \approx \min_{\gamma \in \Gamma} \sum_{k=0}^{K-1} c(\beta_k, \pi_{\gamma(\beta_k)}) + c_K(\beta_k)$. When we derive the optimal solution *over the graph*, we will have to take into consideration the fact that each initial hyperbelief sample may not be able to reach each target hyperbelief sample. To address this issue we will bound, from above and below, the cost based in the distance from terminal hyperbelief to the target hyperbelief sample. Thus, we assume that in addition to the cost function $c(\cdot)$, both a function giving an upper-bound $\bar{c}(\cdot)$, and a lower-bound function $\underline{c}(\cdot)$ are provided so that $\underline{c}(\beta, \pi) \leq c(\beta, \pi) \leq \bar{c}(\beta, \pi)$, $\forall \beta \in \mathscr{P}_\beta, \pi \in \Pi$. One possibility is to select Liptshitz continuous cost functions, whereby there exists a linear bounds between any two points, such that an upper and lower linear bound can be established for the cost based on the distance.

Algorithmically, SHOT proceeds in two steps: 1) generating the digraph $\mathscr{G}$ and 2) optimizing over the digraph $\mathscr{G}$ (or hyperbelief space roadmap) and then refining the results. Both of these steps will be described in more detail in the following sections.

4.2 Generating the Digraph

To approximate a given POMDP system using SHOT, we begin by generating the vertices for the digraph $G =< N, E >$ (or roadmap), whereby each vertex n^i corresponds to a randomly sampled hyperbelief β^i. For notational convenience we will label the vertices with the same label as the hyperbelief samples to minimize confusion. Next the digraph is instantiated by generating the edge information between neighboring hyperbeliefs. Planning between each source and target hyperbelief sample is then performed to generate bounds on the cost and distance from initial hyperbelief sample and terminal hyperbelief sample.

4.2.1 Generating the Vertices: Sampling a Random Hyperbelief Set

Sampling is a difficult problem and an enormous amount of research has gone into sampling for probabilistic roadmap methods [3, Ch. 7]. The difficulties of effectively sampling the hyperbelief space could be exacerbated by sampling in the hyperbelief space because the hyperbelief space is infinite dimensional. However, we

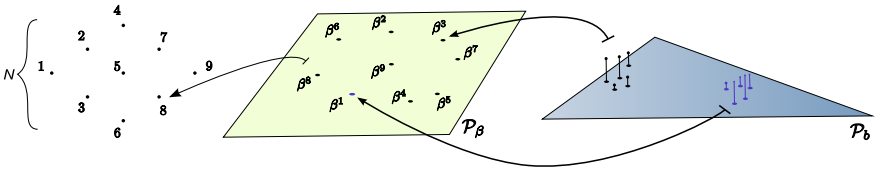


Fig. 1. Random set, N ,of hyperbelief samples in the hyperbelief space $\mathscr{P}_\beta$ and corresponding vertices

conjecture that POMDP systems are often locally insensitive to variations in the starting hyperbelief position, so that two nearby hyperbeliefs converge together under the same feedback policy.

To generate the vertices of the graph we sample a set of random beliefs and then push them through the transition and observation probability functions for random number of stages to generate hyperbelief samples. The motivation for this is that often the initial belief/hyperbelief becomes less influential in the outcome of the hyperbelief as the number of stages increases (or as the size of the information history increase). Undoubtedly this method of sampling is not ideal. In the future, we wish to implement a more intelligent sampling schemes. However, we will demonstrate later in Section 5 that even a naive sampling method performs well for the examples provided.

An illustration of the process of generating belief samples to create a hyperbelief sample, which represents a vertex in the digraph G, is depicted by Figure 1, where a total of 9 hyperbeliefs are sampled. The bottom illustration corresponds to the belief space. There are two clusters of weighted impulses, each of which corresponds to a hyper-particle set, which is an approximated hyperbelief. Each of these hyper-particle sets is then a point in the hyperbelief space: points β^1 and β^8 the center illustration in the figure. The top illustration represents the digraph with each vertex corresponding to a sample in the hyperbelief space (e.g., β^6 is associated with vertex n^6.

4.2.2 Generating the Edge Information: Planning between Hyperbelief Samples

After generating the sample hyperbelief set, which corresponds to the vertices in the digraph G, the edge information is then generated (where each edge β^i to β^j is denoted as $i \rightarrow j$). The edge information comprises the set of intermediary hyperbeliefs from the source hyperbelief (corresponding to the source vertex) and the final distance from the target hyperbelief (corresponding to the target vertex). Intermediate hyperbeliefs comprise the set of hyperbeliefs that represent the system as it evolves from the source hyperbelief sample towards a target hyperbelief sample.

Local policies are employed to plan between hyperbelief samples and are meant to be simple functions that only capture the value landscape locally. To determine the policy for an intermediate hyperbelief, we employ a policy that minimizes the cost of some value function $v_l : \mathscr{P}_\beta \times K \rightarrow \mathbb{R}$.

One choice is for the cost to be the distance to the target hyperbelief. There are various distance measures between beliefs such as the expected graph distance or the Jensen-Shannon divergence (see [7] for a catalog of probability distance functions) that are defined over the belief space. We can then use the probability metric over the hyperbelief space utilizing the belief distance function. The *probability metric* $PM(\cdot)$ [12] between two hyperbeliefs is defined as the expected distance between the two hyperbeliefs, or

$$PM(\beta^i \| \beta^j) = \int_{b^i, b^j \in \mathscr{P}_b} d(b^i, b^j) p(b^i, b^j | \beta^i, \beta^j) db^i db^j.$$

The distance between any two hyperbeliefs, using the probability metric, is bounded by the maximum distance between any two beliefs in the belief space. Thus, any two hyperbeliefs are only a finite distance apart.

Once the local policy is selected, the next set of intermediate hyperbeliefs is generated. Starting for some source hyperbelief sample β^i and some target hyperbelief sample β^j, corresponding to some vertex in G, the policy for the source hyperbelief sample is determined and hyper-particle filtering performed to evolve the system forward one stage into the future to generate an approximation of

$$\beta_k^{i \to j} = \Upsilon(\beta_{k-1}^i, \pi_{i \to j}),$$

where we denote $\beta_0^{i \to j} = \beta^{i \to j}$. This process repeats, generating a set of intermediate hyperbeliefs until a maximum number of iterations is exceeded or until the greedy policy can no longer make any progress towards the target hyperbelief sample. This method is then performed for a specified number of neighbors for each vertex in the digraph. The result a digraph with intermediate hyperbeliefs.

As an example, the distance from the terminal and closest hyperbelief from the target hyperbelief (e.g., $d_{end}^{4 \to 2}$ between source β^4 and target β^2) is illustrated in Figure 2. As can be seen in Figure 2, each vertex attempts to reach β^2. For instance β^4 proceeds to plan into the future for five stages before the greedy policy is unable to make any additional progress towards β^2. Once the local policy terminates, each edge has associated with it the distance from the final intermediate hyperbelief to

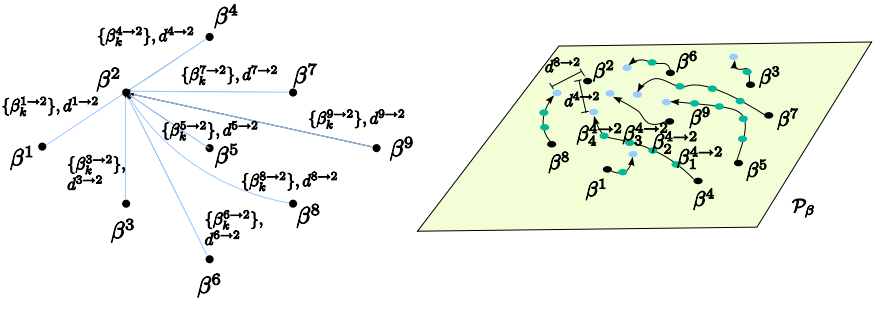


Fig. 2. Graph G and associated edge information directed to vertex β^2

Algorithm 1. Generate hyperbelief sample graph

Require: $\mathscr{P}_\beta$: hyperbelief space, n: number of vertices, K: maximum number of iterations, π: greedy policy
Ensure: G: digraph with vertices N, and edges E (includes edge information (e.g. weights))
  $N = \{\beta^i\}_{i=1}^n \leftarrow$ randomly generate n samples from $\mathscr{P}_\beta$
  for all $i \in N$ **do**
    for all $j \in N$ **do**
      $\beta_0^{i \leftarrow j} \leftarrow \beta^i$
      min_dist $\leftarrow$ distance between $\beta_0^{i \leftarrow j}$ and β^j
      dist $\leftarrow$ -∞
      $k \leftarrow 1$
      while $k \leq K$ and dist $\leq$ min_dist **do**
        $\beta_k^{i \rightarrow j} \leftarrow \Upsilon(\beta_{k-1}^{i \rightarrow j}, \pi)$
        dist $\leftarrow$ distance between $\beta_k^{i \leftarrow j}$ and β^j
        if dist $\leq$ min_dist **then**
          add $\beta_k^{i \leftarrow j}$ to $E^{i \leftarrow j}$ edge information
        end if
        $k \leftarrow k+1$
      end while
    end for
  end for
  return $G \leftarrow < N, E >$

hyperbelief β^2 as well as each of the intermediate hyperbeliefs (which is illustrated for β^4). The algorithm for the digraph generation (including both the hyperbelief sampling and the planning) is described in Algorithm 1.

4.2.3 Generating Distance and Stage Bounds

With the the edge information for the digraph determined, we can determine the sensitivity of a source hyperbelief sample at reaching a target hyperbelief sample. We will generate both a lower bound $\underline{\mathrm{d}}^{i \rightarrow j}(\cdot)$ and an upper bound $\bar{d}^{i \rightarrow j}(\cdot)$ for each edge $i \rightarrow j$ in the graph G. Together the lower and upper bounding functions indicate a range for how close a hyperbelief sample nearby the source hyperbelief is able to reach a target hyperbelief sample. The bounding functions determine the terminal distance relative to how far the starting distance is from the source hyperbelief (such that $\underline{\mathrm{d}} : \mathbb{R} \rightarrow \mathbb{R}$ and $\bar{d} : \mathbb{R} \rightarrow \mathbb{R}$). This information is used below in Section 4.3.2 to generate bounds on the cost function from one edge to the next along a path in the graph.

One method to determine the sensitivity of the hyperbelief space around some source hyperbelief sample is to generate a set of samples around the source hyperbelief sample and then execute the local policy to determine the ability of the set of hyperbeliefs at attaining the target hyperbelief sample. A similar method is then used to determine upper and lower bounding functions for the number of stages.

With all the bounding information determined, we can perform graph optimization to determine the nearly optimal policy.

4.3 Finding the Approximately Optimal Policy

With the vertices and the edge information for the digraph G determined, we can perform digraph optimization on the directed graph to determine the optimal choice of edges to traverse. This set of ordered edges defines the higher level policy.

4.3.1 Generating Edge Costs

We begin the optimization process by generating the weight for each edge is generated based on the intermediate hyperbeliefs. The total weight for a given edge is then computed as $v^{i\to j} = \sum_{k=0}^{K^{i\to j}} c(\beta_k^{i\to j}, \pi_{i\to j}, k) + c_K(\beta^j)$, where $K^{i\to j}$ is the number of intermediate hyperbeliefs and where $\beta_0^{i\to j}$ is the source hyperbelief sample. The terminal cost $c_K(\beta^j)$ is determined by performing simulations of each vertex in the digraph to minimize the the terminal cost associated with each vertex. In this way, each vertex also has a terminating cost of $c_K(\beta^j)$.

For discounted cost functions, the discount applied to the value function for each edge is initialized to start at stage 0. As the evaluation from edge to edge proceeds, the value of each edge is multiplied by the discount factor raised to the total number of stages performed before reaching the said edge.

One of the main attributes of this method is that the value of each edge can be determined on-line for various cost functions without having to re-execute the graph edge algorithm because each edge stores the transient hyperbeliefs and the distance to the target hyperbelief, both of which are cost function independent.

4.3.2 Digraph Optimization

Local policies are used to plan between hyperbelief samples, so all that remains is to establish what is the optimal choice of hyperbelief samples to visit. To do this, we optimize the cost over the digraph. However, as discussed in Section 4.2.2, each source hyperbelief sample may not be able to attain a target hyperbelief sample using the applied greedy policy. To account for the error introduced in the edge weights on the digraph, we utilize $\bar{c}(\cdot)$ to generate an upper-bound on the value function. Because, for some i and j in N, $v^{i\to j}$ is a linear sum of the cost of each hyperbelief, and we can derive bounds for each $i, j \in N$ on $\bar{v}^{i\to j}(\cdot)$, which represents the upper-bound and $\underline{v}^{i\to j}(\cdot)$ as the lower bound on the value function of each node, respectively. Both bounds are a function of the distance between any hyperbelief and the source vertex of the edge. Thus, where $d^{j\to l} = d(\beta_{K^{i\to j}}^{i\to j}, \beta_0^{j\to l})$, the bounds on the value between path $i \to j \to l$, are given as

$$v^{i\to j} + \underline{v}^{j\to l}(d^{j\to l}) \le v^{i\to j\to l} \le v^{i\to j} + \bar{v}^{j\to l}(d^{j\to l}).$$

When more there is more than one intermediate vertex in a path $n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_t$, the upper bound on the cost along the path becomes

$$\bar{v}^{n_1 \rightarrow n_t} = v^{n_1} + v^{n_2}(d^{n_1 \rightarrow n_2}) + \sum_{s=3}^{t-1} \max_{d_{min}^{n_{s-1} \rightarrow n_s} \leq d \leq d_{max}^{n_{s-1} \rightarrow n_s}} \bar{v}^s(d) + \max_{d_{min}^{n_{t-1} \rightarrow n_t} \leq d \leq d_{max}^{n_{t-1} \rightarrow n_t}} v_K^t(d), \tag{3}$$

where

$$d_{max}^{n_{s-1} \rightarrow n_s} = \max_{d_{min}^{n_{s-2} \rightarrow n_{s-1}} \leq d \leq d_{max}^{n_{s-2} \rightarrow n_{s-1}}} \bar{d}^{n_{s-1} \rightarrow n_s}(d)$$

and

$$d_{max}^{n_1 \rightarrow n_2} = d_{end}^{n_1 \rightarrow n_2}. \tag{4}$$

The lower bound distance can be defined by substituting max with min in (4). Similarly, a lower bound on the cost can be found by substituting max for min and $\bar{v}$ for $\underline{v}$ into (3). The bound on the distance is determined at each stage and then passed along the path to the next vertex to be used to estimate the bounds on the distance at the next stage. These distance bounds are then used to determine both the upper and lower bounds on the cost for the specified path.

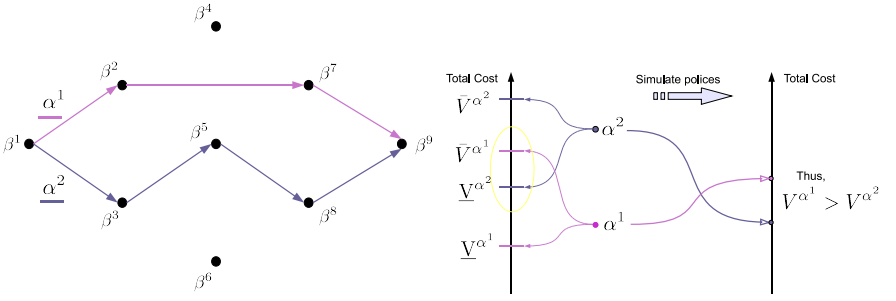
We derive the graph optimization method based Bellman Ford's algorithm [4, Ch. 24], which can handle negative weight cycles. The basic idea is to start with each vertex $i \in N$. Update the minimal value for that vertex as the cost $c_K(\beta^i)$. Then for each edge $i \rightarrow j \in E$, evaluate the minimum upper bound and the minimum lower-bound on the cost from vertex i to adjacent vertex j and update the cost of the edge. We then repeat this step for the number of vertices. This method has a computational complexity of $O(|N|^2 |E|)$.

This process is performed to determine both the least upper-bound cost and path as well as the the lower-bound cost associated with that path. We then simulate the system using to least upper-bounded path on the digraph because the graph optimization provides a bound on the cost. The result of the simulation is the actual cost of the specified policy. With the actual cost and a bound on the lowest cost for the system, we can provide the range that the optimal solution may reside.

One of the deficiencies of this approach is that the resulting policy, and the the associated total cost, is not necessarily optimal for the specified hyperbelief set. What is achieved is a bound on the optimal policy and cost. To overcome this shortcoming we present an iterative refinement algorithm, which will find the exact optimal solution for a given digraph in a finite amount of time and that each iteration of the method will only improve the quality of the solution.

4.3.3 Refinement

The quality of the bounding functions (both the upper and lower bounds) potentially plays a critical role in producing a suitable solution in the previously outlined optimization technique. We present a refinement algorithm to mitigate the effects of the bounding function and also to guarantee the optimal solution for the given digraph will be found if enough time is given.



(a) Example least lower-bound and least upper-bound optimal switching policy

(b) Possible configuration of the cost for every possible policy

Fig. 3. Example least lower-bound and least upper-bound optimal switching policy

The process initiates just as the method without refinement. Then the optimization technique described above is performed to obtain the least upper-bound path with its associated upper-bound $\bar{V}^{\alpha^1}$ and lower-bound $\underline{V}^{\alpha^1}$ costs. An example least upper-bound path is depicted in Figure 3a as α^1. Then the least upper-bound path is

Algorithm 2. Refinement method

Require: optimal-cost, optimal-policy, verified-policies: a list policies evaluated so far, G: hyperbelief graph, k: refine iteration
Ensure: optimal-cost, optimal-policy
  check-path ← the $k-1$'th verified-policies
  while check-path is not empty **do**
    i ← pop off vertex from the end of check-path
    for all neighbors j of i **do**
      value-list(j) ← $v^i.end + \bar{v}$(i,j)
    end for
    append to v-list second lowest vertex in value list
  end while
  insert v-list policy into sorted verified-policies
  least-bound ← min of least-bound and lower bound for v-list policy
  k-policy ← determine k'th lowest policy from verified-policies
  if bounded cost of k-policy < optimal-cost **then**
    new-cost ← simulate system with k-policy
    if new-cost < verified-cost **then**
      optimal-policy ← new-policy
    end if
  end if
  if $k <$ maximum refine iterations and (optimal-cost - least-bound) $\geq$ threshold **then**
    optimal-policy ← refine(optimal-cost, optimal-policy, verified-policies, G, k + 1)
  end if
  return optimal-cost, optimal-policy

simulated to obtain the actual cost V^{α^1}. Then the refinement begins by determining the second least upper-bound path, e.g., α^2 in Figure 3a. The upper-bound cost $\bar{V}^{\alpha^2}$ and lower-bound cost $\underline{V}^{\alpha^2}$ induced by path α^2 is then determined. The resulting cost V^{α^1} is compared to the second least upper-bounded cost. If the lower bound $\underline{V}^{\alpha^2}$ of second to lowest cost is above the actual cost of the already simulated cost, then the next (third) least-upper bound path is determined. However, if the actual cost V^{α^1} for α^1 resides within the bound of this second to least upper-bound path, then the second to least upper-bounded path is simulated. The cost V^{α^2} of the resulting simulation is then compared to the previous lowest cost. If the newly simulated cost is the lowest then it is selected as the minimum. In Figure 3b, after simulation of α^2, it is determined that the cost V^{α^1} is greater than $\underline{V}^{\alpha^2}$. Thus α^2 is simulated and it is discovered that α^2 achieves a lower cost than α^2.

This process continues until a maximum number of iterations (less than or equal to the number of possible paths in the digraph) or minimum threshold on the bound of the cost is met. At any stage of this process the bound on the cost is evaluated as the minimum actual simulated cost to the least lower-bounded cost of the remaining (non-simulated) paths. The resulting minimum path imposes a switching order for the local policies. The set of local policies and the switching order is the approximately optimal policy. The algorithm describing this process is outlined in Algorithm 2.

5 Results

To verify the SHOT technique, we applied it to several of the benchmark problems found in the literature: namely, 4*x*4 and hallway2. Both of these methods are discounted infinite horizon problems. To simulate the system we adapted k to represent each stage of execution. For the presented examples, we generate upper bounds experimentally by measuring the sensitivity of the cost associated with the starting distance of each neighboring edge. For example, given edge $i \rightarrow j$, we determine the relationship of the cost for $v^{l \rightarrow i}$ relative to the distance of edge l to i for each l such that edge $l \rightarrow i$ exists. We then generate a piecewise linear, Lipshitz upper and lower bound for the cost. In a similar manner, bounds for the distance from one edge to the next were determined as well as the bounds on the number of elapsed iterations.

The limited results produced so far seem to verify that that this method is comparable to other POMDP approximation methods. For the 4*x*4 example with average of 18 hyperbeliefs, SHOT achieved an average cost of 3.703 with no variance in the 10 iterations run. The method HSVI2 presented in [18] achieved a reward of 3.75 for this example. With the he hallway2 example similar relative performance was achieved: 0.416, which is on par with the the expected reward 0.48 achieved by SARSOP [13]. The hallway2 example averaged 127.6 hyperbelief samples.

One point to note is that each of these problems comprise a set of states the once entered results in the system being uniformly distributed over the state space at the next iteration. This reset mechanism is pivotal in finding the optimal solution and

if none of the hyperbelief sample are able to place some probability over this state, then the resulting policy fails to achieve the target and receives an inferior reward. This sensitivity is one of the motivating factors of behind the SHOT approach.

The examples presented were chosen to demonstrate the validity of the SHOT method. However, the presented method was developed with large observation spaces and planning for systems where the number of stages required to achieve a satisfactory result is prohibitive for most current POMDP approximation methods. To more fully address this class of problems and the performance of presented method, a more exhaustive set of experimental results for the SHOT method can be found at [5].

6 Conclusion

A method for finding nearly optimal policies for POMDPs with total cost or finite time horizons was presented. The proposed method (SHOT) was a sampling-based technique using a two-level hierarchical planner, whereby the lower level planner executes local, greedy feedback policies and the higher level planner coordinates the order of hyperbelief samples that are visited. This method attempts to capture the structure of the POMDP system, which is independent of the starting hyperbelief (or belief) and the objective function, so that an efficient multi-query technique can be utilized for any initial hyperbelief or objective function for a given POMDP system.

Future research includes evaluating alternate sampling schemes, such as generating event samples from the observation space. Sampling from the observation space has the potential to alleviate some of the issues that arise from attempting to sample from the hyperbelief space directly.

References

1. Alterovitz, R., Simeon, T., Goldberg, K.: The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In: Proceedings of Robotics: Science and Systems, Atlanta, GA, USA (June 2007)
2. Apaydin, M., Brutlag, D., Guestrin, C., Hsu, D., Latombe, J., Varm, C.: Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. Journal of Computational Biology 10, 257–281 (2003)
3. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2002)
5. Davidson, J.: Experimental Results for the SHOT technique (2008), http://robotics.ai.uiuc.edu/~jcdavid/SHOT
6. Davidson, J.C., Hutchinson, S.A.: Hyper-particle filtering for stochastic systems. In: IEEE International Conference on Robotics and Automation, pp. 2770–2777 (2008)
7. Gibbs, A.L., Su, F.E.: On choosing and bounding probability metrics. International Statistical Review 70, 419–435 (2002)

8. Hsu, D., Lee, W.S., Rong, N.: What makes some POMDP problems easy to approximate? In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems. MIT Press, Cambridge (2008)
9. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101, 99–134 (1998)
10. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
11. Kim, J., Pearce, R., Amato, N.: Extracting optimal paths from roadmaps for motion planning. In: IEEE International Conference on Robotics and Automation, ICRA 2003. Proceedings, September 14-19, vol. 2, pp. 2424–2429 (2003)
12. kukaszyk, S.: A new concept of probability metric and its applications in approximation of scattered data sets. Computational Mechanics 33(4), 299–304 (2003)
13. Kurniawati, H., Hsu, D., Lee, W.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proc. Robotics: Science and Systems (2008)
14. Liberzon, D.: Switching in Systems and Control. Birkhäuser, Boston (2003)
15. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1025–1032 (2003)
16. Prentice, S., Roy, N.: The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. In: Proceedings of the 13th International Symposium of Robotics Research (ISRR), Hiroshima, Japan (November 2007)
17. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, pp. 520–527 (2004)
18. Smith, T., Simmons, R.: Point-based POMDP algorithms: Improved analysis and implementation. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence (2005)
19. Spaan, M.T., Vlassis, N.: A point-based POMDP algorithm for robot planning. In: IEEE International Conference on Robotics and Automation, pp. 2399–2404 (2004)
20. Spaan, M.T., Vlassis, N.: PERSEUS: Randomized point-based value iteration for POMDPs. Journal of Artificial Intelligence Research 24, 195–220 (2005)
21. Thrun, S.: Monte Carlo POMDPs. In: Advances in Neural Information Processing Systems, pp. 1064–1070 (2000)
22. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT Press, Cambridge (2005)

Part V
Computational Minimalism

On the Value of Ignorance: Balancing Tracking and Privacy Using a Two-Bit Sensor

Jason M. O'Kane

Abstract. We consider a target tracking problem in which, in addition to sensing some information about the position of a mobile target, the tracker must also ensure that the *privacy* of that target is preserved, even in the presence of adversaries that have complete access to the tracker's sensor data. This kind of problem is important for many kinds of robot systems that involve communication systems or agents that cannot be fully trusted. In this paper, we (1) introduce a formal, quantitative definition for privacy, (2) describe algorithms that allow a robot to maintain conservative estimates of its performance in terms of tracking and privacy, (3) give strategies for the tracker to maximize its tracking performance, subject to constraints on the allowable privacy levels, and (4) present an implementation of these methods along with some experimental results.

1 Introduction

Robots must deal continually with uncertainty. The general problem of sensing and acting to reduce uncertainty is well-studied and continues to receive attention. This paper considers the related but complementary problem of sensing and acting in order to *maintain* uncertainty, rather than eliminating it. To motivate this (perhaps counterintuitive) idea, consider the following problem (inspired by [8, 9]):

> **Panda tracker problem:** A giant panda moves unpredictably through a wilderness preserve. A mobile robot tracks the panda's movements, periodically sensing partial information about the panda's whereabouts and transmitting its findings to a central base station. At the same time, poachers attempt to exploit the presence of the tracking robot — either by eavesdropping on its communications or by directly compromising the robot itself — to locate the panda. We assume, in the worst case, that the poachers have access to any

Jason M. O'Kane
Department of Computer Science and Engineering, University of South Carolina
e-mail: jokane@cse.sc.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 235–249.
springerlink.com

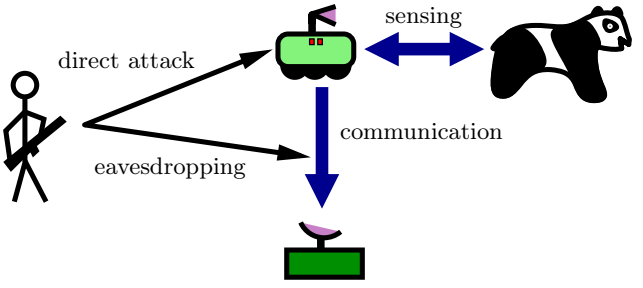


Fig. 1. Basic components of the panda tracker problem. The tracking robot needs to sense the panda's location and communicate this information to the base station, while preventing the poacher from exploiting this information to locate the panda.

information collected by the tracking robot, but they cannot control its motions. The problem is to design the tracking robot so that the base station can record coarse-grained information about the panda's movements, without allowing the poachers to obtain the fine-grained position information they need to harm the panda. See Figure 1.

In this problem, the tracking robot must collect some information about the panda's location, but cannot collect too much information without endangering the *privacy* (and therefore, the safety) of the panda. More generally, this sort of privacy concern is relevant in at least three kinds of robotics applications:

1. Applications which require communication over an untrusted channel.
2. Applications which require communication with an untrusted recipient.
3. Applications in which the robot itself cannot be fully trusted.

For each of these classes of problems, it is essential to design a robot and its programming to ensure that the information it collects or transmits is not harmful to disclose. We emphasize that the goal is to ensure that privacy is preserved even when the best possible use is made of all the available information. This approach specifically precludes the possibility of getting a "free ride" by intentionally forgetting. It also allows us to avoid explicitly considering the actions and knowledge of the poachers — we simply ensure that it is acceptable for them to know everything that the tracker knows.

In this paper, we consider one version of the panda tracker problem, in which a *target* moves in the plane unpredictably but with bounded velocity. A *tracker* monitors that target's motions using a sensor that reports very coarse information about the direction from the tracker to the target. A crucial first observation is that the ability to maintain privacy is closely related to the informative value of the robot's sensors. A tradeoff exists between strong sensors (which make tracking easy but privacy difficult) and weak sensors (which make privacy easy to maintain at the expense of tracking accuracy). We present results suggesting that a reasonable balance between tracking and privacy can be obtained using sensors that provide relatively little information.

Our approach is based on analyzing the tracker's *information states* as they change over time. In this context, the information state at a particular time is simply the set of states that are consistent with the history of actions executed and sensor readings obtained by the tracker up to that time. A distinctive feature of this work is that we maintain upper and lower bounds on the extent of the information state, without needing to compute the information states themselves.

The primary goal of this work is to investigate the role that privacy concerns can play in robotic tracking problems. After reviewing related prior work in Section 2, this paper makes several new contributions.

1. We give quantitative definitions for tracking and privacy, in terms of the tracker's information states. These definitions appear in Section 3.
2. We describe representations and algorithms for the tracker to maintain a lower bound on privacy and an upper bound on the tracking accuracy achieved throughout its execution. These algorithms, which are introduced in Section 4, require $O(1)$ storage and $O(1)$ update time, so they are well-suited to implement, for example, on extremely simple mobile sensor platforms.
3. We derive reactive strategies, in Section 5, for planning the motions of the tracker that keep privacy and tracking within acceptable bounds. Implementations of these algorithms are presented.

Concluding discussion appears in Section 6.

2 Relationship to Prior Work

The panda tracker problem was first proposed in the context of wireless sensor networks. Both temporal [8] and spatial [9] privacy of an observed target can be protected by careful design of the network's routing protocols. Crucial to both of these works is their application of Kerckhoffs' Principle, under which it is assumed that the adversaries have complete knowledge of the protocol used by the network. This view is paralleled by our assumption that the adversaries have access to all of the information the tracker does. We encompass both forms of privacy by considering how position information changes over time. To our knowledge, the present work is the first to directly examine the effects of sensing and action on privacy.

Our work is also closely related to the lines of research that seek to understand tasks by solving them in spite of severe sensing limitations. Problems in manipulation [3, 5], localization [4, 17], navigation [10, 12], mapping [7], and pursuit-evasion [6] have been solved for such systems. These examples are meant to be representative, rather than exhaustive. A common thread through much of this work is a two-step approach, based on solving the passive information state update problem before considering how to choose actions actively. In each case, the solution depends on representing and updating the robot's knowledge in ways that make this active planning manageable. This work follows the same approach.

More specifically, target tracking problems have also been studied in the literature. Much of this work is focused on maintaining visibility between the tracker and the target within a cluttered environment. Specific methods have used dynamic

programming [11], sampling-based [15], and reactive [14] approaches. The novelty of our work is that we are the first to explicitly include privacy in the formulation. One closely related variation is the stealth tracking problem, in which the tracker must maintain visibility of the target, while remaining near the boundary of the target's visibility polygon to avoid possible detection [1]. Our work differs primarily because we are concerned with the privacy of the target's location, rather than that of tracker's location, and because we consider a much weaker sensor.

3 Problem Statement

This section formalizes the panda tracker problem. We begin in Section 3.1 with a general formulation for which we give a quantitative definition of privacy, then in Section 3.2 apply this formulation to the specific version of the panda tracker problem solved in this paper.

3.1 General Formulation

Consider a formulation with the following elements:

- A division of time into a sequence of discrete, but not necessarily equal length, *stages*, numbered $k = 1, 2, \ldots$. In each stage, the robot can both sense and act.
- A *state space* X. In stage k, the relevant information about the situation in the world is modeled by a *state* $x_k \in X$.
- A *distance function* $d : X \times X \to \mathbb{R}$ for pairs of states, under which X is a metric space.
- An *initial condition* $\eta_0 \subseteq X$ representing a set of possible starting states. Unless η_0 is a singleton set, the actual starting state is unknown to the robot.
- An *observation space* Y, so that $y_k \in Y$ models the sensor information collected by the robot at stage k.
- An *observation function* $h : X \to Y$, under which $y_k = h(x_k)$. That is, h describes how the observation y_k is determined by the current state x_k.
- An *action space* U. The robot chooses one action $u_k \in U$ to execute in each stage.
- A *nature action space* Θ. The nature action $\theta_k \in \Theta$ at stage k models the effects of noise, actions by other decision makers, or both.
- A *state transition function* $f : X \times U \times \Theta \to X$ that describes how the state changes. The state x_{k+1} at stage $k+1$ is given by $x_{k+1} = f(x_k, u_k, \theta_k)$.

From these basic ingredients, we can make two additional definitions for convenience. First, define an iterated transition function to apply multiple transitions at once: $f(x, u_1, \theta_1, \ldots, u_k, \theta_k) = f(\cdots f(f(x, u_1, \theta_1), u_2, \theta_2) \cdots, u_k, \theta_k)$. Second, denote the *preimage* of each observation y by $H(y)$, so that $H(y) = \{x \in X \mid h(x) = y\}$. The robot does not necessarily know its state, but instead must rely on its sensor-action history to make its decisions. At stage k, this information consists of $y_1, \ldots, y_k$ and $u_1, \ldots, u_{k-1}$. This history can be used to determine the set of possible states at stage k. The following two definitions make this notion precise.

Definition 3.1. A state $x \in X$ is *consistent with* a sensor-action history $(y_1, u_1, \ldots, y_{k-1}, u_{k-1}, y_k)$ if there exists some $x_1 \in \eta_0$ and a sequence of nature actions $\theta_1, \ldots, \theta_{k-1}$ such that $x = f(x_1, u_1, \theta_1, \ldots, u_{k-1}, \theta_{k-1})$ and $y_j = h(f(x_1, u_1, \theta_1, \ldots, u_{j-1}, \theta_{j-1}))$ for each $j = 1, \ldots, k$.

Definition 3.2. The *information state* (I-state) η_k at stage k is the set of all states consistent with the robot's sensor-action history. The *information space* (I-space) $\mathscr{I}$ is the powerset of X, which contains all possible I-states.

The intuition is that a state x is a member of an I-state η_k if and only if the robot cannot conclusively rule out x as a possible state at stage k. Such I-states are useful because the robot can always keep track of its I-state, even when there is insufficient information to determine the underlying true state. Let $F : \mathscr{I} \times U \times Y \to \mathscr{I}$ denote an *information transition function* that describes how the I-state changes over time. Given an information state η_k, and action u_k, and an observation y_{k+1}, we have $\eta_{k+1} = F(\eta_k, u_k, y_{k+1})$. We can describe this information transition in terms of f and H:

$$F(\eta_k, u_k, y_{k+1}) = \{f(x, u_k, \theta) \mid x \in \eta_k, \theta \in \Theta\} \cap H(y_{k+1}). \tag{1}$$

How are these I-states related to privacy and tracking? Informally, privacy is preserved whenever the I-state is a relatively large set; conversely good tracking depends on keeping the I-state relatively small. This intuition leads us to the following quantitative definitions of privacy and tracking.

Definition 3.3. A closed ball B is a *privacy disk* for an I-state η_k if $B \subseteq \mathrm{cl}(\eta_k)$. The *privacy margin* of an I-state η_k is the largest radius over all privacy disks of η_k.

Definition 3.4. A closed ball B is a *tracking disk* for an I-state η_k if $\mathrm{cl}(\eta_k) \subseteq B$. The *tracking margin* of an I-state η_k is the smallest radius over all tracking disks of η_k.

The intuition is that the privacy and tracking disks form inner and outer circles around the boundary of η_k. A privacy disk is a region within which no states can be ruled out; a tracking disk is a region outside of which every state can be ruled out. The radii of these two disks provide quantitative measures of privacy and tracking. Note that the privacy disk of maximal radius is not necessarily unique (for example if η_k is the region between two concentric circles), but the minimal tracking disk *is* unique.

One might imagine an alternative to Definitions 3.3 and 3.4 based directly on the area or volume of the I-state. Such an approach would not use separate measures for privacy and tracking, but instead quantify the robot's uncertainty by the volume of its I-states and work to ensure that this value remains within given bounds. Our decision to use Definitions 3.3 and 3.4 rather than this type of volume-based approach is motivated by several considerations. First, our definitions are more strict than the volume-based alternative — an I-state will, in general, have volume greater than or equal to that of its largest privacy disk and less than or equal to that of its smallest tracking disk. Therefore, the performance bounds we obtain are valid for the volume-based definitions as well. Second, the fact that we are concerned only

with the largest ball contained within and the smallest ball around the I-state allows us to maintain bounds on tracking and privacy that do not require computing the I-state's exact shape. Finally, for tracking in particular, the volume of the I-state may not represent the tracking accuracy very well. In extreme cases, the I-state may consist of several small regions distant from one another. Such an I-state would have small volume, but would require a large amount of travel to search completely.

3.2 The Panda Tracker Problem

Now we can state the panda tracker problem as a specific instance of the general formulation from Section 3.1.

A single target moves unpredictably in an obstacle-free plane. Its motions are continuous and are constrained by a known maximum velocity, but nothing else is known about its trajectory. A single tracker, modeled by a point with orientation in the plane, monitors the movements of the target using a low-speed, low-resolution bearing sensor that reports, in a coordinate system attached to the tracker, which quadrant contains the target. The tracker is able to move very quickly in comparison to the target. Each stage models a period of motion by both the target and the tracker, followed by a sensor reading by the tracker. Formally, we can describe this system as follows:

- The state space is $X = \mathbb{R}^2$. A single state represents the target's position relative to the tracker, in coordinate frame with the tracker's position at its origin and its positive y-axis in the direction the tracker faces. We denote the individual coordinates of a state using superscripts, so that $x = (x^{(1)}, x^{(2)})$ for each $x \in X$. Note, however, that it will sometimes be more straightforward to consider a global frame, specifying the position and orientation of the tracker and the position of the target relative to a stationary reference point.
- The distance metric d is the standard Euclidean metric over $\mathbb{R}^2$.
- The initial condition η_0 is a disk known to contain the target. This includes, as a special case in which the disk is centered at the origin, the situation where the tracker starts with an upper bound on the distance to the target.
- The action space U is the set of rigid body transformations achievable by the tracker within one stage (that is, between two consecutive sensor readings). For concreteness, we consider a robot that can translate and rotate freely with maximal velocity v_{trk} and maximal angular velocity ω_{trk}.
- The nature action space Θ is the set of translations of magnitude at most v_{tgt}, the displacement achievable by the target in a single stage.
- The state transition function $f : X \times U \times \Theta \to X$ applies the motion u_k of the tracker and the motion θ_k of the target to the current state x_k to compute the new state $x_{k+1} = f(x_k, u_k, \theta_k)$.
- The observation space is $Y = \{0,1\} \times \{0,1\}$.
- The observation function $h : X \to Y$ returns the quadrant containing the target, according to

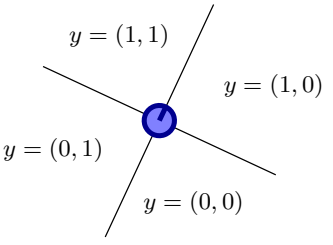


Fig. 2. Observation preimages for the panda tracker problem. The sensor reports which quadrant contains the target.

$$h(x_k) = h\left(x_k^{(1)}, x_k^{(2)}\right) = \left(\left[x_k^{(1)} \geq 0\right], \left[x_k^{(2)} \geq 0\right]\right),$$

in which $[\cdot]$ is the indicator function that returns 1 if its argument is true and 0 otherwise. See Figure 2.

For purposes of comparison, we consider two specific, mutually independent choices for the tracker's goal:

G1: *Minimize tracking margin* — Keep the tracking margin as small as possible, without regard for the privacy margin.
G2: *Maintain privacy bound* — Keep the tracking margin as small as possible, while ensuring that the privacy margin is no smaller than a given minimum, denoted ρ.

In each case, we are concerned with the worst-case tracking and/or privacy margins, taken over all possible trajectories of the target. We discuss strategies to achieve these two goals in Section 5.

4 Passive Updates

This section presents algorithms for maintaining a lower bound on the privacy margin and an upper bound on the tracking margin for the panda tracker problem defined in Section 3.2. These techniques are passive in the sense that we defer (to Section 5) the question of how to choose actions for the tracker, considering for now only how to estimate the performance during a given execution. We present two approaches to this problem: an optimal algorithm based on computing the I-state itself, and a second approach that is more efficient, but produces only upper and lower bounds on the tracking and privacy margins, respectively.

4.1 Updates Directly from the I-State

For the first approach, consider how the tracker might maintain an explicit representation of the I-state η_k. The initial I-state η_0 is a disk. In subsequent stages, to

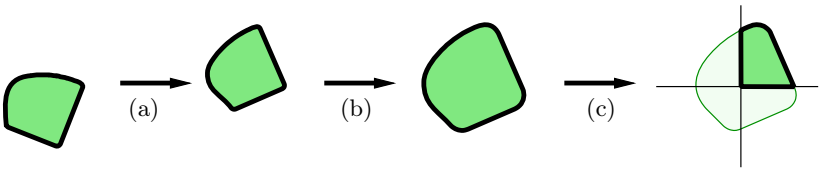


Fig. 3. Computing explicit information transitions for the panda tracker problem. (a) Rigid body transformation by u_k. (b) Minkowski sum with Θ. (c) Intersection with $H(y_{k+1})$.

compute η_{k+1} from η_k, we must perform three transformations on η_k (recalling that states are represented in a coordinate frame attached to the tracker):

1. Rigid body transformation by u_k, reflecting the motion of the tracker.
2. Minkowski sum of the resulting region with Θ, which is a disk of radius v_{tgt}, reflecting all possibilities for the unknown motion θ_k of the target.
3. Intersection of the resulting region with a quarterplane[1] (the observation preimage $H(y_{k+1})$), reflecting the new information supplied by the sensor.

Brute force algorithms for these operations would take time linear in the complexity of boundary of η_k. In practice, the circular arcs arising in Step 2 can be approximated by polygonal chains. With this information, the exact privacy margin can be computed by finding the largest disk inside η_k in $O(n)$ time [2]. Likewise, the exact tracking margin can be found by computing, in $O(n)$ time [13], the smallest enclosing disk around η_k.

4.2 *Indirect Updates with Limited Memory*

For some applications, including those in which the tracker is a very simple mobile sensor platform, computation and memory resources are at a premium. This section presents passive update algorithms appropriate for such situations, which maintain estimates on the tracking and privacy margins but require only relatively small, constant amounts of time and space. The central idea is to define two sequences of disks, $P_1 \ldots, P_k$, and $T_1, \ldots, T_k$, so that, at each stage i, we have $P_i \subseteq \eta_i \subseteq T_i$. Note that these need not necessarily be the largest privacy disks or the smallest tracking disks for their respective I-states. In the initial condition, the I-state itself is a disk, so we start with $P_1 = T_1 = \eta_0$. In subsequent stages, we use P_k to compute P_{k+1} and T_k to compute T_{k+1}, as described below. After the tracker computes P_{k+1} and T_{k+1}, it discards P_k and T_k.

First, we consider how to compute a new privacy disk P_{k+1} given a privacy disk $P_k \subseteq \eta_k$, an action u_k, and an observation y_{k+1}. Accounting for the motions of the tracker and the target is straightforward. As in Section 4.1, we perform a rigid body transformation on P_k by u_k, followed by a Minkowski sum with Θ. Since P_k is a disk, these operations can be realized by a translation of the center of P_k and an increase

[1] We use the term quarterplane to refer to the planar intersection of two halfplanes with orthogonal boundary lines.

of its radius by v_{tgt}. Let P'_k denote the disk resulting from these operations. It remains to consider the effects of the observation y_{k+1}. At stage k, no states within P_k can be ruled out. Therefore, we choose for P_{k+1} the largest disk inside of $P'_k \cap H(y_{k+1})$, resulting in a region in which no states in P_{k+1} can be ruled out for stage $k+1$. The problem is thereby reduced to find the largest disk inside the region of intersection between a disk and a quarterplane.

Computing a new tracking disk T_{k+1} from T_k, u_k, and y_{k+1} follows a similar procedure, but instead requires finding the smallest disk around the quarterplane/disk intersection region. Due to space constraints, details about both of these algorithms appear in a separate technical report [16].

5 Active Tracking

In this section we build on the passive update methods from Section 4 by introducing techniques to actively control the tracker to achieve goal conditions G1 (Section 5.1) and G2 (Section 5.2). In Section 5.3 we discuss an implementation of these algorithms and present some experimental results.

Throughout this section, we use a global coordinate frame, considering the motions of the tracker and target relative to an arbitrary but fixed external reference point. Since the observation received is determined by the tracker's position and orientation, we can view the active tracking problem as a problem of deciding where, and in what orientation, to place the tracker, thereby indirectly positioning the boundaries of the observation preimages. Throughout this section, we assume that the tracker has access at each stage to the privacy disk P_k and the tracking disk T_k, computed as described in Section 4.2. The strategies we derive are expressed by prescribing the destination of the tracker, relative to P_k and T_k. Note, however, that if the tracker has sufficient computation power, it can instead use the explicit passive update method described in Section 4.1, and replace P_k and T_k with the smallest enclosing and largest enclosed disks of η_k. In either case, we retain from Section 4 the convention that P'_k denotes a circle with the same center as P_k, with a radius larger by v_{tgt} (the translation is not needed in this coordinate frame); T'_k is defined similarly in terms of T_k.

5.1 Minimizing the Tracking Margin (G1)

Goal condition G1 requires the tracker to keep the tracking margin as small as possible, without regard for privacy. Thus, G1 refers to a typical target-tracking application. Our approach to achieving this goal is based on the following observation:

Lemma 5.1. *For a given T_k, the worst-case tracking margin for stage $k+1$ is minimized when the tracker moves to the center of T_k.*

Proof. If the tracker positions itself at the center of T_k, the sensor preimage boundaries divide T_k into four parts that are identical up to a rotation. Regardless of the target's position, the resulting T_{k+1} will have the same radius. In contrast, if the

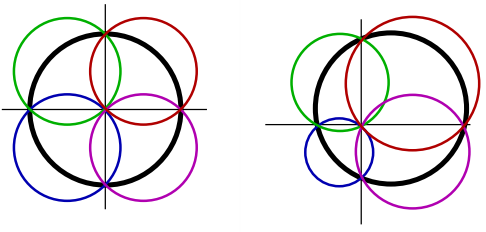


Fig. 4. [left] Positioning the tracker at the center of T_k minimizes the worst-case tracking margin at stage $k+1$. [right] Other positions for the tracker lead to larger worst-case tracking margins. In this case, the tracker is positioned below and to the left of the center of T_k, and the worst case occurs if the target is in the top right quadrant.

tracker is not at the center of T_k, then for at least one of the four possible observations, the resulting T_{k+1} will be larger, thereby degrading the worst-case tracking margin. See Figure 4.

This observation leads directly to a strategy to achieve G1:

Strategy for G1. *Move to the center of T_k.*

To execute this strategy, the tracker must be fast enough to move in a single stage between the centers of successive tracking disks T_k and T_{k+1}.

5.2 Maintaining a Privacy Bound (G2)

Next, we consider G2, in which the tracker wants to keep the tracking margin as small as possible, subject to the constraint that the privacy margin can become no smaller than a given minimum, denoted ρ. Our basic approach is to compute at each stage a "safe region" of destinations for the tracker, within which the privacy bound is maintained, while collecting some information. If this region is empty, the tracker instead chooses a position from which only one observation is possible, preventing any reduction of the privacy margin. The following lemma makes this idea more precise.

Lemma 5.2. *For given T_k and P_k, the privacy margin at stage $k+1$ will be at least ρ if, at the end of stage k, either*

(a) the radius of P'_k is at least $(1+\sqrt{2})\rho$, and the tracker is within distance $\sqrt{\text{radius}(P'_k)^2 - 2\,\text{radius}(P'_k)\rho} - \rho$, of the center of P_k, facing the center of P_k, or

(b) the tracker is farther than $\sqrt{2}\,\text{radius}(T'_k)$ from the center of T_k, oriented so that none of the observation preimage boundaries cross the interior of T'_k.

Proof. First consider condition (a). Notice that for a fixed distance from the center of P_k, the worst-case privacy is maximized when the tracker faces the center of P_k. Assuming the tracker takes this orientation, we consider (as in Section 4.2) a

coordinate system in which the center of P_k lies on the vertical axis, and the horizontal axis bisects $H(y_{k+1})$. Let $(0,c)$ denote the center of P_k and let $(s,0)$ denote the vertex of $H(y_{k+1})$. Note that because the tracker faces $(0,c)$, we have $c = s$. It remains to find the value of c such that the resulting largest enclosed disk has radius ρ. Let $(a,0)$ denote the position of the center of the largest enclosed disk. The distance from this point to observation preimage boundary is $(a-c)/\sqrt{2}$; the distance to the boundary of P_k is $\text{radius}(P'_k) - \sqrt{c^2+a^2}$. The radius of the largest enclosed disk is maximized when these values are equal, which occurs at $c = \sqrt{\text{radius}(P'_k)^2/2 - \text{radius}(P'_k)\rho} - \rho/\sqrt{2}$. Finally, for the tracker to generate this c (and, therefore, to maintain privacy margin ρ), it must have distance $\sqrt{2}c$ of the center of P_k. For condition (b), notice that both P'_k and T'_k will be fully contained within $H(y_{k+1})$. In this case, only one observation is possible, and neither P_{k+1} nor T_{k+1} will be affected the observation.

Lemma 5.2 provides two options for the tracker: To stay near the center of P_k (condition (a)), or to move far from the center T_k (condition (b)). Recalling the proof of Lemma 5.1, note that whenever the former option is available, it results in a smaller tracking margin in stage $k+1$. As a result, we can state the following motion strategy for the tracker:

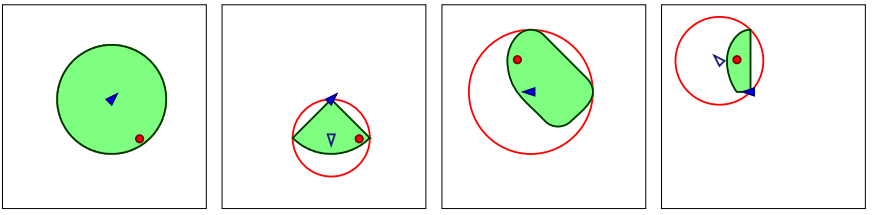
Strategy for G2. *If $r > (1+\sqrt{2})\rho$, move to the point in the disk described in condition (a) of Lemma 5.2 closest to the center of T_k, facing the center of P_k. Otherwise, move distance $\sqrt{2}\,\text{radius}(T'_k)$ from the center of T_k, facing an angle of $\pi/4$ away from the line between the center of T_k and the tracker's destination.*

In either case, the tracker's destination is straightforward to compute using circle-line intersection methods.

5.3 Simulation and Experimental Results

To evaluate our approach, we have implemented these algorithms in simulation. Figures 5 and 6 show several stages of example executions of G1 and G2, respectively. In these figures, the target's position is shown with a small circle. The tracker's current position and orientation are shown with a shaded triangle; the destination for the tracker is shown as an unshaded triangle. The large outer circle is the boundary of T_k, and the shaded region is the I-state η_k. Figure 6 also shows the boundary of P_k.

One natural question is to ask how much the tracking performance is degraded when the tracker's goal is G2 with nonzero ρ, compared to the "pure tracking" performance of G1. That is, how much tracking accuracy must be sacrificed in order to guarantee a given level of privacy? To answer this question, we performed experiments with $v_{tgt} = 2.5$ and values of ρ varying between 0.1 and 2.5 in increments of 0.1. In each trial, the target moved through a sequence of randomly-selected destination points and we recorded the tracking radius achieved for each of these 25 values of ρ. Each trial lasted for 1,000 stages, and we averaged the results over 1,000 trials. Figure 7 summarizes the results of this experiment. The results, as one might



(a) (b) (c) (d)

Fig. 5. Execution snapshots the Strategy for G1 described in Section 5.1. (a) The initial condition, in which the tracking disk and the true information state are identical. (b) After the first observation is received, the tracking disk encloses the information state (which the tracker does not compute). The tracker's destination is the center of this tracking disk. (c) Before sensing in stage 28 of this execution. The information state η_{28} and tracking disk T'_{28} are shown. (d) After sensing in stage 28. Notice that in this example, the tracking disk T_k is only a poor approximation of the smallest tracking disk.

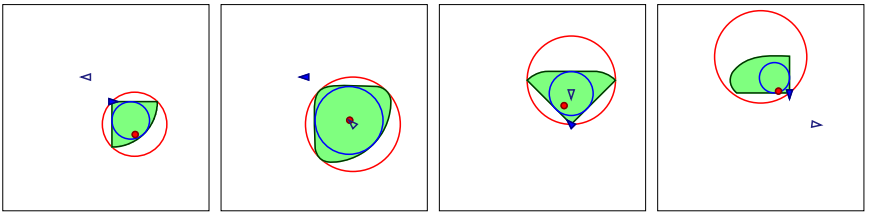


Fig. 6. Execution snapshots the Strategy for G2 described in Section 5.2. Shown left to right are the situations after sensing for $k = 1, 2, 3, 4$. Notice that the safe region is empty for $k = 1$ and $k = 4$.

expect, illustrate a tradeoff between tracking and privacy. When $\rho = 0.1$, the average results are similar to those for G1, and for larger values of ρ, the tracking radii increase roughly linearly. More substantial differences can be seen when comparing the maximum tracking radius that occurred in each trial.

We also performed experiments to assess the performance advantage gained by using the explicit updates of Section 4.1 compared to the constant time and space indirect updates of Section 4.2. We used $v_{tgt} = 1.5$, two different motion patterns for the target: a random walk and repeated circuits around a square of radius 10, and three motion strategies for the tracker: random movements, our strategy for G1, and our strategy for G2 with $\rho = 1.5$. For each of these six combinations, we performed 100 trials of 1,000 stages each, computed the ratios of tracking radii for explicit compared to implicit updates, and computed similar ratios for the privacy margins. For both tracking and privacy, we divided the smaller value by the larger one, so that a result of 1.0 would indicate optimal performance. Figure 8 shows the results. Privacy results are shown only for the cases in which the tracker's goal is G2; for the other two goals, P_k is quickly eliminated, leading to average ratios very close to zero.

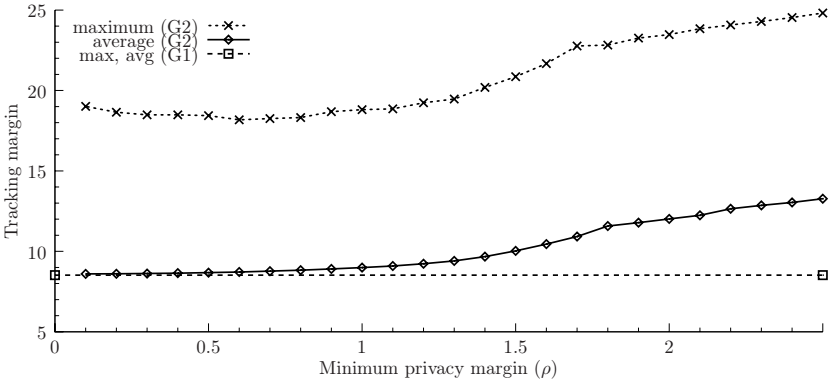


Fig. 7. Comparison of tracking radii for various tracker goals, including G1, and G2 with varying values of ρ.

Tracking	Tracker: random	Tracker: G1	Tracker: G2
Target: random	0.596	0.838	0.619
Target: square	0.352	0.770	0.386

Privacy	Tracker: G2
Target: random	0.789
Target: square	0.696

Fig. 8. Comparison of implicit versus explicit update methods for various motion strategies. In each table, a value of 1 would indicate that the performance of the implicit algorithm perfectly matches the explicit (optimal) algorithm. [left] Ratio of exact tracking margins for explicit updates to tracking margin upper bounds for implicit updates [right] Ratio of privacy margin lower bounds for implicit updates to exact privacy margins for explicit updates.

Several interesting phenomena can be observed in the results. For tracking, implicit updates generate the best results when the tracker's goal is G1. This reflects the fact that, in general, the I-states achieved in this condition have the approximate shape of a right isosceles triangle, with the center of T_k on the hypotenuse. This leads to similar results for both implicit and explicit updates. Observe also that, across all tracker strategies, the approximation is superior for random motions of the target than for the square pattern. This reflects that fact that, whenever the target makes a long motion in a single direction, implicit updates repeatedly make the same kinds of over- or underestimates. The issue is especially visible for G2, in which the tracker is frequently "held back" by a need to stay within P_k.

6 Discussion and Conclusion

This paper presented an initial investigation of the role that a geometric view of privacy can play in robotic tracking problems. Unsurprisingly, many important questions have been left unanswered.

In this paper we considered only one particular kind of sensor for the tracker, one in which the preimages are quarterplanes. Alternatives we contemplated in the

development of this work include sensors whose preimages are disks and their planar complements (that is, the plane with a disk deleted), halfplanes, and annuli. Each requires its own method for passively updating the tracker's information and its own unique active motion strategies. Note, however, that depending on the sensor models used, it may be challenging or impossible to design active strategies to satisfy G1 or G2. For example, if the observation preimages are a disk and its complement (such as would be the case if the sensor reported only whether the target was within a certain radius), then regardless of the tracker's strategy, the worst-case tracking margin would increase without bound, corresponding the case in which the target moves far away from the tracker throughout its execution. Similar difficulties occur if the observation preimages are halfplanes.

Recall also that the sensor models used in this work are deterministic, not allowing for sensor noise. The primary effect that sensor noise would have on our formulation is that the observation preimages would overlap, allowing the observation to be chosen in some unknown way whenever the state is in an overlap region. This enlargement of the preimages can be expected to make privacy easier to maintain at the expense of increased difficulty in tracking. For the particular case of our quadrant sensor, the update algorithms would require only slight generalizations, and we expect the resulting active strategies to be quite similar as well.

Finally, several other extensions merit additional research, the most interesting of which include the presence of obstacles; related problems with multiple trackers, multiple targets, or both; and nontrivial mobility constraints.

Acknowledgements. The author is grateful to Wenyuan Xu for the introduction to the panda tracker problem, and for helpful discussions. This work is partially supported by a grant from the University of South Carolina, Office of Research and Health Sciences Research Funding Program.

References

1. Bandyopadhyay, T., Li, Y., Ang, M.H., Hsu, D.: Stealth tracking of an unpredictable target among obstacles. In: Proc. Workshop on the Algorithmic Foundations of Robotics, pp. 43–58 (2004)
2. Chin, F.Y., Snoeyink, J., Wang, C.A.: Finding the medial axis of a simple polygon in linear time. In: Staples, J., Katoh, N., Eades, P., Moffat, A. (eds.) ISAAC 1995. LNCS, vol. 1004, pp. 382–391. Springer, Heidelberg (1995)
3. Erdmann, M., Mason, M.T.: An exploration of sensorless manipulation. IEEE Transactions on Robotics and Automation 4(4), 369–379 (1988)
4. Erickson, L., Knuth, J., O'Kane, J.M., LaValle, S.M.: Probabilistic localization with a blind robot. In: Proc. IEEE International Conference on Robotics and Automation (2008)
5. Goldberg, K.Y.: Orienting polygonal parts without sensors. Algorthmica 10, 201–225 (1993)
6. Guilamo, L., Tovar, B., LaValle, S.M.: Pursuit-evasion in an unkown environment using gap navigation trees. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (2004)

7. Huang, W.H., Beevers, K.R.: Topological mapping with sensing-limited robots. In: Proc. Workshop on the Algorithmic Foundations of Robotics, pp. 235–250 (2004)
8. Kamat, P., Xu, W., Trappe, W., Zhang, Y.: Temporal privacy in wireless sensor networks. In: Proc. IEEE International Conference on Distributed Computing Systems (2007)
9. Kamat, P., Zhang, Y., Trappe, W., Ozturk, C.: Enhancing source-location privacy in sensor network routing. In: Proc. IEEE International Conference on Distributed Computing Systems (2005)
10. Kamon, I., Rivlin, E.: Sensory-based motion planning with global proofs. IEEE Transactions on Robotics and Automation 13(6), 814–822 (1997)
11. LaValle, S.M., González-Baños, H.H., Becker, C., Latombe, J.-C.: Motion strategies for maintaining visibility of a moving target. In: Proc. IEEE International Conference on Robotics and Automation, pp. 731–736 (1997)
12. Lazanas, A., Latombe, J.C.: Landmark-based robot navigation. In: Proc. National Conference on Artificial Intelligence (AAAI) (1992)
13. Meggido, N.: Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. SIAM Journal on Computing 12, 759–776 (1983)
14. Murrieta, R., Sarmiento, A., Bhattacharya, S., Hutchinson, S.A.: Maintaining visibility of a moving target at a fixed distance: The case of observer bounded speed. In: Proc. IEEE International Conference on Robotics and Automation (2004)
15. Murrieta-Cid, R., Tovar, B., Hutchinson, S.: A sampling-based motion planning approach to maintain visibility of unpredictable targets. Autonomous Robots 19(3), 285–300 (2005)
16. O'Kane, J.M.: On the value of ignorance: Balancing tracking and privacy using a two-bit sensor. Technical Report USC CSE TR-2009-001, University of South Carolina (2009)
17. O'Kane, J.M., LaValle, S.M.: Localization with limited sensing. IEEE Transactions on Robotics 23, 704–716 (2007)

On the Existence of Nash Equilibrium for a Two Player Pursuit-Evasion Game with Visibility Constraints

Sourabh Bhattacharya and Seth Hutchinson

Abstract. In this paper, we present a game theoretic analysis of a visibility based pursuit-evasion game in a planar environment containing obstacles. The pursuer and the evader are holonomic having bounded speeds. Both the players have a complete map of the environment. Both the players have omnidirectional vision and have knowledge about each other's current position as long as they are visible to each other. The pursuer wants to maintain visibility of the evader for maximum possible time and the evader wants to escape the pursuer's sight as soon as possible. Under this information structure, we present necessary and sufficient conditions for surveillance and escape. We present strategies for the players that are in *Nash Equilibrium*. The strategies are a function of the *value* of the game. Using these strategies, we construct a value function by integrating the *adjoint equations* backward in time from the termination situations provided by the corners in the environment. From these value functions we recompute the control strategies for the players to obtain optimal trajectories for the players near the termination situation. As far as we know, this is the first work that presents the necessary and sufficient conditions for tracking for a visibility based pursuit-evasion game and presents the equilibrium strategies for the players.

1 Introduction

Consider a situation in which a group of mobile pursuers having bounded speeds are trying to keep sight of an unpredictable evader in a cluttered environment. In order to

Sourabh Bhattacharya
Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign
e-mail: sbhattac@illinois.edu

Seth Hutchinson
Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign
e-mail: seth@illinois.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 251–265.
springerlink.com

deploy minimum number of pursuers needed to track the evader it would be useful to know the best strategy that can be used by a single pursuer. In this work, we analyze the problem of a mobile pursuer trying to track a mobile evader in an environment containing obstacles. Both the pursuer and the evader are holonomic with bounded speeds and can see each other at the beginning of the game. The players do not have knowledge of each other's future actions. We formulate the problem of tracking as a game in which the goal of the pursuer is to keep the evader in its field-of-view for maximum possible time and the goal of the evader is to escape the pursuer's field-of-view in minimum time by breaking the line of sight around a corner.

An interesting application of this problem is in security and surveillance systems. It may be useful for a security robot to track a malicious evader that is trying to escape. Also, an "evader" may not be intentionally trying to slip out of view. A pursuer robot may simply be asked to continuously follow and monitor at a distance an evader performing a task not necessarily related to the target tracking game[6]. The pursuer may also be monitoring the evader for quality control, verifying the evader does not perform some undesired behavior, or ensuring that the evader is not in distress. The results are useful as an analysis of when escape is possible. If it is impossible to slip away, it may be desirable for the evader to immediately surrender or undertake a strategy not involving escape. In home care settings, a tracking robot can follow elderly people and alert caregivers of emergencies. Target-tracking techniques in the presence of obstacles have been proposed for the graphic animation of digital actors, in order to select the successive viewpoints under which an actor is to be displayed as it moves in its environment [19].

In the past, we have addressed tracking problems similar to the one in this paper. In [7], we address the problem of a pursuer trying to track an antagonistic evader around a single corner. We partition the visibility region of the pursuer into regions based on the strategies used by the players to achieve their goals. Based on these partitions we propose a sufficient condition of escape for the evader in general environments. In [8], given the initial position of the evader in a general environment, we use the sufficient condition to compute an approximate bound on the initial positions of the pursuer from which it might track the evader. The bound depends on the ratio of the maximum speed of the evader to that of the pursuer. If the initial position of the pursuer lies outside this bound, the evader can escape the pursuer's sight. Moreover, we provide strategies for the evader to escape irrespective of pursuer's actions. In this work, we formulate the target-tracking problem as a game in which the pursuer wants to maximize the time for which it can track the evader and the evader wants to minimize it. We compute the strategies for the players that are in *Nash equilibrium*. If a player follows its equilibrium strategy, it is guaranteed of a minimum outcome without any knowledge of the other player's future actions. Moreover when a pair of strategy for the players is in *Nash equilibrium* then any unilateral deviation of a player from its equilibrium strategy might lead to a lower outcome for it. Consider a situation in which the pursuer can keep the evader in sight for time t_f when the players follow their equilibrium strategies. If the evader deviates from its equilibrium strategy then the pursuer has a strategy to track it for a time greater than t_f. On the other hand, if the pursuer deviates from its equilibrium

strategy then the evader can escape in time less than t_f. Hence there is no motivation for either of the players to deviate from their equilibrium strategies due to the lack of knowledge of the other player's future actions. For a pair of equilibrium strategies for the players either the evader can escape the pursuer's sight in finite time or the pursuer can track the evader forever. Hence computing them gives us the strategies sufficient for tracking or escape, whichever holds at a given point in the state space. As far as we know, this is the first work that addresses the necessary and sufficient conditions for tracking and provides equilibrium strategies for the players. We use these strategies to integrate the kinematic equations of the system backward in time from the termination situations to obtain the optimal trajectories for the players.

Prior work regarding the problem of tracking is based on discretizing the motion models of the players or the state-space in which the game is being played[17, 5]. These techniques lead to approximate numerical solutions that become computationally inefficient with increasing time horizon of the game. Moreover they assume a prior model of uncertainty for the evader's future actions. Contrary to these works, we use continuous time motion models for the players and provide closed form solutions to the coupled non-linear differential equations that govern our system kinematics. Hence no error is introduced in the solutions due to discretizations of any form. Further, our results are valid for scenarios in which the players have no knowledge about each others future actions.

Some variants of the tracking problem have also been addressed. In [11], the authors take into account the positioning uncertainty of the robot pursuer. Game theory is proposed as a framework to formulate the tracking problem, and an approach is proposed that periodically commands the pursuer to move into a region that has no localization uncertainty in order to re-localize and better track the evader afterward. [10] presents an off-line algorithm that maximizes the evader's minimum time to escape for an evader moving along a known path. Since the entire trajectory of the evader is known beforehand, the problem reduces to a single player optimization problem. In [14] and [4], gradient descent algorithms have been proposed by formulating a local risk function for a pursuer having the local map of the evader. [4] deals with the problem of *stealth target tracking* where a robot equipped with visual sensors tries to track a moving target among obstacles and, at the same time, remain hidden from the target. Obstacles impede both the tracker's motion and visibility, and also provide hiding places for the tracker. A tracking algorithm is proposed that applies a local greedy strategy and uses only local information from the tracker's visual sensors and assumes no prior knowledge of target tracking motion or a global map of the environment. In [23], the problem of target tracking has been analyzed at a fixed distance between the pursuer and evader. Optimal motion strategies are proposed for a pursuer and evader based on critical events.

In this work, we use differential games to analyze a pursuit-evasion problem. The theory of deterministic pursuit-evasion was single-handedly created by R.Isaacs that culminated in his book [15]. A general framework based on the concepts in classical game theory and the notion of tenet of transition was used to analyze pursuit-evasion problems. Problems like the *Lady in the Lake*, *Lion and the Man*, *Homicidal chauffer* and *Maritime Dogfight Problem* were introduced in this book. A modification to

the classical problems involves the consideration of discrete-time versions of these problems and the application of a proper information structure to compute the value of the game [13, 12]. An exhaustive analysis of solved and partly solved zero-sum differential games is provided in [3] and [18]. Most of the classical problems in pursuit-evasion deal with players in obstacle-free space having either constraints on their motion or constraints on their control due to under-actuation. In the recent past, researchers have analyzed pursuit-evasion problems with constraints in the state space. In [20, 21, 22], a pursuit-evasion game is analyzed with the pursuer and the evader constrained to move on a two-dimensional conical surface in a three-dimensional space. Our work belongs to this category of problems. In our problem, the state constraints arise due to the presence of obstacles that obstruct visibility as well as motion of the players in the workspace and the control constraints arise as a result of the bounded speed of the players. Apart from these problems, researchers have also analyzed pursuit-evasion in $\mathbb{R}^n$ [16], in non-convex domains of arbitrary dimension [1], in unbounded domains[2] and in graphs [24].

In Section II, we present the formulation of the game. In Section III, we analyze the termination situations presented by the obstacles around any corner in the environment. In Section IV, we present the strategies for the players that are in *Nash equilibrium*. In Section V we present the construction of the optimal trajectories. In Section VI, we present the conclusions and the future work.

2 Formulation of the Game

We consider a mobile pursuer and an evader moving in a plane with velocities $u = (u_p, \theta_p)$ and $v = (u_e, \theta_e)$ respectively. u_p and u_e are the speeds of the players that are bounded by $\overline{v}_p$ and $\overline{v}_e$ respectively. θ_p and θ_e are the direction of the velocity vectors. We use r to denote the ratio of the maximum speed of the evader to that of the pursuer $r = \frac{\overline{v}_e}{\overline{v}_p}$. They are point robots with no constraints in their motion except for bounded speeds. The workspace contains obstacles that restrict pursuer and evader motions and may occlude the pursuer's line of sight to the evader. The initial position of the pursuer and the evader is such that they are visible to each other. The visibility region of the pursuer is the set of points from which a line segment from the pursuer to that point does not intersect the obstacle region. Visibility extends uniformly in all directions and is only terminated by workspace obstacles (omnidirectional, unbounded visibility). The pursuer and the evader know each others current position as long as they can see each other. Both the players have a complete map of the environment. In this setting, we consider the following game. The pursuer wants to keep the evader in its visibility region for maximum possible time and the evader wants to break the line of sight to the pursuer as soon as possible. If at any instant, the evader breaks the line of sight to the pursuer, the game terminates. Given the initial position of the pursuer and the evader, we want to know the optimal strategies used by the players to achieve their respective goals. Optimality refers to the strategies used by the players that are in *Nash equilibrium*.

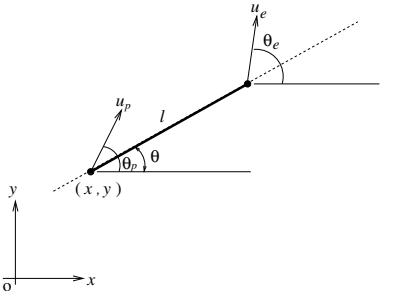


Fig. 1. State variables and Control inputs

We model the system as a non-rigid bar of variable length representing the line of sight between the pursuer and the evader. The bounded velocities of the pursuer and the evader are modeled as control inputs at opposite ends of the bar. Any occlusion between the pursuer and the evader leads to a situation in which the bar intersects the obstacles. Hence the pursuer's goal is to keep the bar in free space for the maximum possible time and the evader's goal is to force the bar to intersect some obstacle as soon as possible. In this work we assume that the line of sight is not blocked due to grazing contact with the boundary. Hence visibility is retained even if a vertex in the environment is incident on the bar.

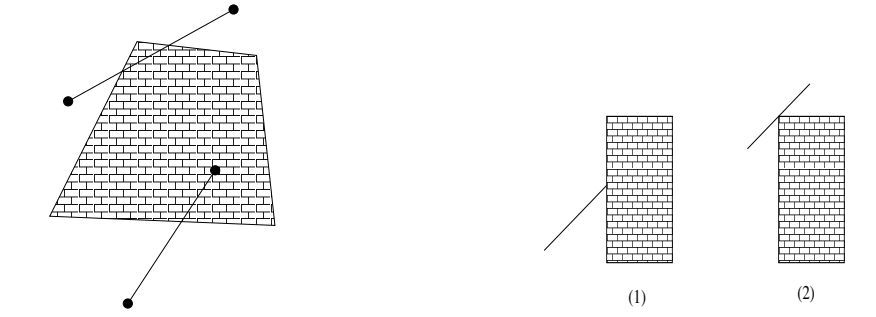
Figure 1 shows the configuration of the system along with the state variables and the control inputs. (x,y) is the position of the end of the bar controlled by the pursuer. l is the length of the bar and θ is the angle made by the bar with the horizontal line. The configuration of the system can be expressed as (x,y,l,θ) and hence it is $\mathbb{R}^3 \times S^1$. In the rest of the paper, $\mathbf{x}(\in \mathbb{R}^3 \times S^1)$ will be used to represent the state of the bar. The pursuer controls the velocity, u, of one end of the bar and the evader controls the velocity, v, of the other end of the bar. The differential equation describing the kinematics of the system is given by the following equation.

$$\begin{pmatrix} \dot{x} \\ y \\ l \\ \theta \end{pmatrix} = \begin{pmatrix} u_p \cos\theta_p \\ u_p \sin\theta_p \\ u_e \cos(\theta_e - \theta) - u_p \cos(\theta_p - \theta) \\ \frac{u_e}{l} \sin(\theta_e - \theta) - \frac{u_p}{l} \sin(\theta_p - \theta) \end{pmatrix}$$

The above equation can also be expressed in the form $\dot{\mathbf{x}} = f(\mathbf{x}, u, v)$.

3 State Constraints and Termination Situations

In this section, we present a description of the obstacles in the configuration space. The workspace contains polygonal obstacles in the plane that obstruct the visibility and motion of the players. Since the system is modeled as a bar representing the line of sight between the players, the obstruction of mutual visibility as well as the motion of the players caused due to obstacles in the workspace can be expressed as a state constraint in $\mathbb{R}^3 \times S^1$. These state constraints can be expressed as



(a) Configuration space obstacles (b) Types of contacts on the boundary

Fig. 2. Configuration Space Obstacles

configuration space obstacles. In $\mathbb{R}^3 \times S^1$, the configuration space obstacles are the set of all (x, y, l, θ) such that the bar has a non-empty intersection with some obstacle in the workspace. Figure 2 shows two such configurations of the bar that lies in configuration space obstacles. In one configuration the obstacle blocks the line-of-sight between the pursuer and the evader. In the other configuration a player is inside the obstacle which is forbidden according to the rules of the game.

The *game set* is the set of all points in $\mathbb{R}^3 \times S^1$ that belong to the free space. Hence the boundary of the game set is same as the boundary of the configuration space obstacles. The boundary of the game set consists of two kinds of contact between the bar and the obstacles. Refer to Figure 2(b). The first kind of contact occurs when at least one end of the bar touches an obstacle in the plane. At no point in time, the state of the game can cross the boundary at such a point as this is equivalent to either of the players penetrating into an obstacle in the workspace. The second kind of contact occurs when a vertex of an obstacle is incident on the bar and these set of points on the boundary of the game set is called the *Target set*. At any point in time, if the current state of the game lies on the target set, then it can cross the boundary according to the rules of the game since in the workspace this is equivalent to breaking the mutual visibility between the players which results in the termination of the game. Since we are interested in situations where the mutual visibility between the players can be broken, we are only interested in the part of the boundary that forms the target set.

In this game, termination occurs only when the evader can break the line of sight to the pursuer around a corner. Every corner in the environment presents an opportunity for the evader to break the line of sight. Hence every corner presents a termination situation for the game.

If the state of the system lies on the target set then a vertex of some obstacle is incident on the bar. The evader cannot guarantee termination at every point on the target set. Figure 3 shows a configuration of the bar in which the system is on the target set. Let d_p denote the distance of the vertex from (x, y) which is same as the distance of the pursuer from the vertex. Let l denote the length of the bar which is same as the distance between the pursuer and the evader. The evader can force

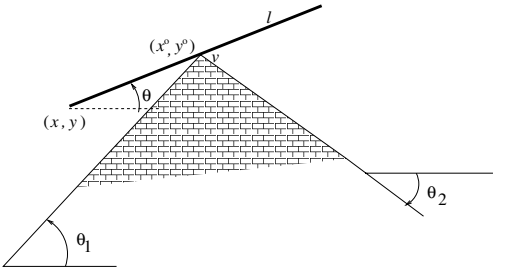


Fig. 3. A configuration of the bar on the target set.

termination if and only if the maximum angular velocity of the evader around the corner is greater than the maximum angular velocity achievable by the pursuer around the corner. This can happen if and only if $\frac{d_p}{l} > \frac{1}{1+r}$. Hence we can further subdivide the target set depending on whether the evader can guarantee termination at that point. The part of the target set where evader can guarantee termination regardless of the choice of the controls of the pursuer is called the *usable part* (UP). The remaining part of the target set outside the UP is called the *non-usable part*(NUP) and the game will never terminate on the NUP. Given any initial position of the pursuer and the evader, the game will always terminate on the UP.

Now we present the equations characterizing the target set around a vertex of an obstacle. Refer to Figure 3. The figure shows a configuration of the bar in which a vertex, v, lies on the bar. Hence the current state of the bar lies on the target set. We want the equation of the hyperplane that characterizes the target set generated by v. Let (x, y, l, θ) be the configuration of the bar and (x^o, y^o) be the coordinates of the vertex of the obstacle. Let $\lambda \in (0, 1)$ be a variable that determines the fraction of the length of the bar between (x, y) and the corner (x^o, y^o). We can write the following equations of constraints for the bar.

$$x^o - x = \lambda l \cos\theta$$

$$y^o - y = \lambda l \sin\theta$$

In the above equation, as λ changes, the point of contact between the bar and the vertex changes. Hence the target set is characterized by the following equation.

$$\Rightarrow F(x, y, l, \theta) = (y^o - y)\cos\theta - (x^o - x)\sin\theta = 0 \tag{1}$$

Since the above equation applies to any $\lambda \in (\frac{r}{1+r}, 1)$, Equation (1) also characterizes the usable part of the target set.

Given a vertex, the target set generated by it in the configuration space has the following boundaries.

- The pursuer lies on the corner$\Rightarrow (x, y) = (x^o, y^o)$.
- The evader lies on the corner$\Rightarrow (x + l\cos\theta, y + l\sin\theta) = (x^o, y^o)$.
- The bar is parallel to either of the edges incident on the vertex : $\theta = \theta_2$ or $\theta = \theta_1$.

Every vertex will generate a target set. The final boundary of the target set generated by a vertex will depend on the position of the other vertices and edges in the environment. But the equation of the target set will be given by (1).

The unit normal to a point (x,y,l,θ) on the target set is given by

$$n(x,y,l,\theta) = \nabla F = \frac{1}{\sqrt{1+(x^o-x)^2\sec^2\theta}}[\sin\theta \quad -\cos\theta \quad 0 \quad -(x^o-x)\sec\theta]^T \tag{2}$$

4 Optimal Strategies

In this section we present the optimal controls for the players. Before we define the concept of optimality we need to define the payoff for the players in the game. Consider a play that terminates at time t_f. Since the pursuer wants to increase the time of termination its payoff function can be considered as t_f. On the other hand since the evader wants to minimize the time of termination its payoff can be considered to be $-t_f$. Since the payoff functions of the players add to zero, this is a *zero sum* game. Another way to show that it is a zero sum game is to observe that the pursuer's gain is equal to the evader's loss and vice-versa. The time of termination is a function of the initial state $\mathbf{x_0}$ and the control history during the play, u and v. Let π denote the functional $\pi : (\mathbf{x_0}, u, v) \rightarrow t_f \in \mathbb{R}$. π is called the *outcome functional* and is given by the following expression.

$$\pi[\mathbf{x}_0, u, v] = \int_0^{t_f} L[\mathbf{x}(\tau), u(\tau), v(\tau)]d\tau + G[\mathbf{x}(t_f)]$$

In the above expression $L[\mathbf{x}(\tau), u(\tau), v(\tau)]$ is called the *running cost function* and $G[\mathbf{x}(t_f)]$ is called the *terminal cost function*. The running cost function is the cost incurred while the game is being played. The terminal cost function is the cost incurred for reaching a particular terminal state on the target set. In this game, $L[\mathbf{x}(\tau), u(\tau), v(\tau)] = 1$ and $G[\mathbf{x}(t_f)] = 0$. The pursuer wants to maximize the outcome functional and the evader wants to minimize it.

For a point $\mathbf{x}$ in the state space, $J(\mathbf{x})$ represents the outcome if the players implement their optimal strategy starting at the point $\mathbf{x}$. It is the time of termination of the game when the players implement their optimal strategies. It is also called the *value* of the game at $\mathbf{x}$. Any unilateral deviation from the optimal strategy by a player can lead to a better payoff for the other player. For example, for a game that starts at a point $\mathbf{x}$, if the evader deviates from the optimal strategy then there is a strategy for the pursuer in which its payoff is greater than $J(\mathbf{x})$ and if the pursuer deviates from the optimal strategy then there is a strategy for evader in which its payoff is greater than $-J(\mathbf{x})$. Since this is a *zero sum* game, any strategy that leads to a higher payoff for one player will reduce the payoff for the second player.

$\nabla J = [J_x \quad J_y \quad J_l \quad J_\theta]^T$ denotes the gradient of the value function. The Hamiltonian of any system is given by the following expression.

$$H(\mathbf{x}, \nabla J, u, v) = \nabla J \cdot f(\mathbf{x}, u, v) + L(\mathbf{x}, u, v)$$

Let $u^* = (u_p^*, \theta_p^*)$ and $v^* = (u_e^*, \theta_e^*)$ be the optimal controls used by the pursuer and the evader respectively. The Hamiltonian of the system satisfies the following conditions along the optimal trajectories[15]. These are called the *Isaacs* conditions.

1. $H(\mathbf{x}, \nabla J, u, v^*) \leq H(\mathbf{x}, \nabla J, u^*, v^*) \leq H(\mathbf{x}, \nabla J, u^*, v)$
2. $H(\mathbf{x}, \nabla J, u^*, v^*) = 0$

Condition 1 implies that when the players implement their optimal strategies any unilateral deviation by the pursuer leads to a smaller value for the Hamiltonian and any unilateral deviation by the evader leads to a larger value of the Hamiltonian. Moreover condition 2 implies that when the players implement their optimal controls the Hamiltonian of the system is zero. The *Isaacs* conditions are an extension of the *Pontryagin's principle* in optimization to a game.

The Hamiltonian of our system is given by the following expression.

$$\begin{aligned} H(\mathbf{x}, \nabla J, u, v) &= \nabla J \cdot f(\mathbf{x}, u, v) + L \\ &= u_p[J_x \cos\theta_p - J_l \cos(\theta_p - \theta) - \frac{J_\theta}{l} \sin(\theta_p - \theta) + J_y \sin\theta_p] \\ &\quad + u_e[J_l \cos(\theta_e - \theta) + \frac{J_\theta}{l} \sin(\theta_e - \theta)] + 1 \end{aligned}$$

Since the evader wants to minimize the time of escape and the pursuer wants to maximize the time of escape, Isaacs first condition requires the following to be true along the optimal trajectories.

$$(u_e^*, \theta_e^*, u_p^*, \theta_p^*) = \min_{u_e, \theta_e} \max_{u_p, \theta_p} H(x, \nabla J, u, v) \tag{3}$$

We can see that the Hamiltonian is *separable* in the controls u_p and u_e i.e., it can be written in the form $u_p f_1(\mathbf{x}, \nabla J) + u_e f_2(\mathbf{x}, \nabla J)$. Hence the optimal controls for the players are given by the following expressions in terms of the gradient of the value function.

$$\begin{gathered} (\cos\theta_p^*, \sin\theta_p^*) \mid\mid (J_x - J_l \cos\theta + \frac{J_\theta}{l} \sin\theta, J_y - J_l \sin\theta - \frac{J_\theta}{l} \cos\theta) \\ (\cos(\theta_e^* - \theta), \sin(\theta_e^* - \theta)) \mid\mid (-J_l, -\frac{J_\theta}{l}) \\ u_e^* = \overline{v}_e \\ u_p^* = \overline{v}_p \end{gathered} \tag{4}$$

Due to lack of space, the derivation is presented elaborately in [9]. In the first and the second equation $\mid\mid$ is used to denote parallel vectors. In case $J_x - J_l \cos\theta + \frac{J_\theta}{l} \sin\theta = 0$ and $J_y - J_l \sin\theta - \frac{J_\theta}{l} \cos\theta = 0$ then θ_p^* can take any value and the pursuer can follow any control strategy. Similarly if $J_l = 0$ and $\frac{J_\theta}{l} = 0$, then θ_e^*

can take any value and the evader can follow any control strategy. These conditions represent *singularity* in the Hamiltonian.

The entire game set can be partitioned into two regions depending on the value of the game. For all the initial positions of the pursuer and the evader for which the value of the game $J(\mathbf{x})$ is finite, the evader can break the line of sight in finite time by following the strategies in Equation(4). For all the initial positions of the pursuer and the evader for which the value of the game is infinite, the pursuer can track the evader forever if it follows the controls given in Equation (4). Hence Equation (3) are the necessary and sufficient conditions for pursuer to track the evader in terms of the Hamiltonian of the system.

The analysis done in this section implies that if we are given the value function $J(\mathbf{x})$ then we can compute the optimal strategies for the players by using equation (4).

5 Construction of Optimal Trajectories

In this section we present the trajectories generated by the optimal control laws that terminate on the UP. We use the following theorem to construct the optimal trajectories.

Theorem 5.1. *[15]Along the optimal trajectory, the following equation holds.*

$$\frac{d}{dt}\nabla J[\mathbf{x}(t)] = -\frac{\partial}{\partial x}H(\mathbf{x}, \nabla J, u^*, v^*)$$

The above equation is called the adjoint equation and the components of $\nabla J(\mathbf{x})$ are called adjoint variables. The retro-time(time-to-go) form of the adjoint equations is

$$\frac{d}{d\tau}\nabla J[\mathbf{x}(\tau)] = \frac{\partial}{\partial x}H(\mathbf{x}, \nabla J, u^*, v^*)$$

where $\tau = t_f - t$ is called the retro-time. t_f is the time of termination of the game.

The adjoint equation is a differential equation for the gradient of the value function $J(\mathbf{x})$ along the optimal trajectories in terms of the optimal controls. Since Equation (4) gives the optimal controls of the players as a function of $\nabla J(\mathbf{x})$, we integrate the adjoint equations backward in time from the UP to obtain $\nabla J(\mathbf{x})$ in terms of the state variables. Substituting $\nabla J(\mathbf{x})$ into the optimal controls gives a feedback control strategy for the players. Substituting the feedback control laws for the players into the kinematic equation leads to the optimal trajectories. Due to lack of space, the construction of the optimal trajectories is provided elaborately in [9].

From the analysis done in [9], we present the optimal trajectories of the players. Let $(x_f, y_f, l_f, \theta_f)$ denote the configuration of the bar at the termination situation. The optimal trajectory of the pursuer as a function of retro-time is given by the following equations.

$$x_p(\tau) = x_f + \tau \overline{v}_p \sin\theta_f$$
$$y_p(\tau) = y_f - \tau \overline{v}_p \cos\theta_f \tag{5}$$

The optimal trajectory of the evader as a function of retro-time is given by the following equations.

$$x_e(\tau) = x_f + l_f \cos\theta_f - \overline{v}_e \tau \sin\theta_f$$
$$y_e(\tau) = y_f + l_f \sin\theta_f + \overline{v}_e \tau \cos\theta_f \tag{6}$$

The optimal trajectories for the pursuer and the evader are straight lines. Moreover the trajectories are perpendicular to the orientation of the bar at the termination situation and hence parallel to each other. The players move in opposite directions as they follow the optimal trajectories. Figure 4 shows the optimal trajectories for the pursuer and the evader that terminate at a corner at the origin. The evader is shown by the red dots and the pursuer is shown by green dots. The black line joining the pursuer and the evader represents the orientation of the bar(line-of-sight) at different time instants. The value of the speed ratio, r, is 0.5. At the termination situation, the bar is oriented at an angle of $\frac{\pi}{4}$ with respect to the x, the position of the pursuer is $(-3,-3)$ and the position of the evader is $(1,1)$. The payoff for both the players at any point on the optimal trajectory is given by the variable τ since it is the time required for termination. In the figure, the payoff for an orientation of the bar is shown on the side of the bar. The bar with $\tau = 0$ represents the termination situation. If the pursuer deviates from its optimal strategy then the evader has a strategy for which it can escape around the corner in time less than τ. If the evader deviates from its

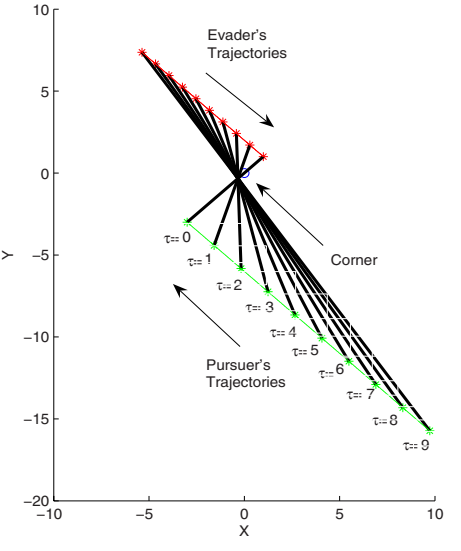
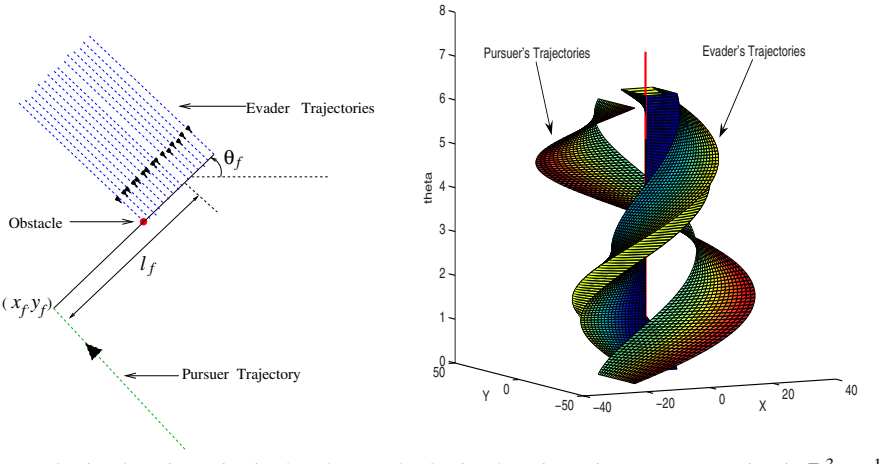


Fig. 4. Optimal trajectories for a terminating situation around a corner

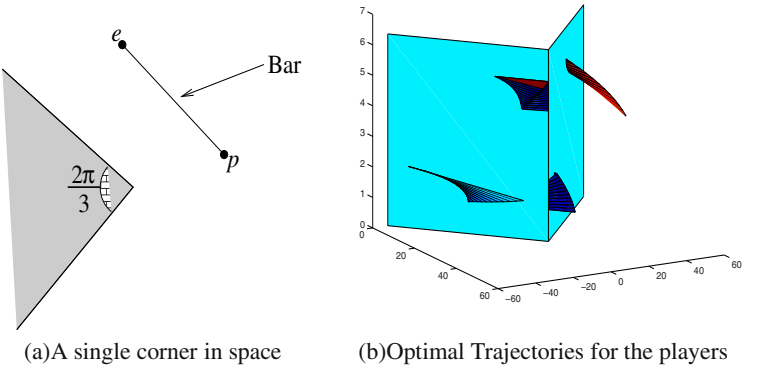


(a) Optimal Trajectories in the plane (b) Optimal Trajectories across a section in $\mathbb{R}^3 \times S^1$

Fig. 5. Optimal trajectories for an environment having a single point obstacle

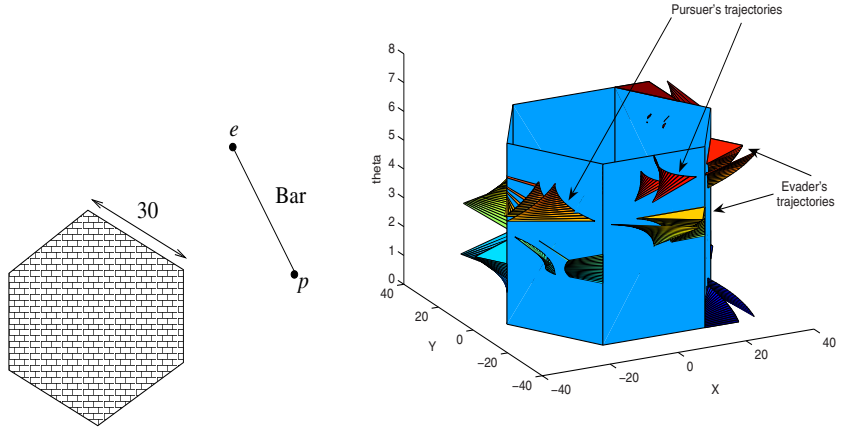
optimal trajectory then the pursuer has a strategy for which it can track the evader for a time greater than τ. This is due to the fact that the trajectories are obtained from strategies that are in Nash equilibrium. Hence there is no motivation for either of the players to deviate from their optimal strategies.

For a general environment in the plane, the optimal trajectories lie in $\mathbb{R}^3 \times S^1$. In order to depict them in $\mathbb{R}^3$, we need to consider a subspace of the optimal paths terminating at a corner. In the following examples, for each corner in the environment we show the subspace of the optimal paths that have a fixed distance of the pursuer from the corner at the termination situation. The value of the speed ratio, r, is 0.66 in all the following examples. Figure 5 shows the optimal trajectories for the players



(a)A single corner in space (b)Optimal Trajectories for the players

Fig. 6. Optimal trajectories of the players for a corner in space



(a)A Hexagonal obstacle in space (b)Optimal trajectories for the players

Fig. 7. Optimal trajectories of the players for a hexagonal obstacle in space

in a simple environment containing a point obstacle at the origin. The line of sight between the pursuer and the evader is broken if it passes through the origin. The evader wants to minimize the time required to break the line of sight and the pursuer wants to maximize it. Let $(x_f, y_f, l_f, \theta_f)$ represent the orientation of the bar at the termination situation. Figure 5(a) shows the optimal trajectories of the players for all possible values of l_f for a constant value of x_f, y_f and θ_f. Figure 5(b) shows the optimal trajectories for every orientation of the bar at the termination situation. The z axis represents the angle of the bar at the termination situation. A cross-section parallel to the xy-plane gives the optimal trajectories of the players in a plane for a given θ_f. The red line in the middle denotes the point obstacle. The inner spiral is formed by the optimal trajectories of the evader and the outer spiral is formed by the optimal trajectory of the pursuer. The color of a point is a representative of the value of the game, $J(\mathbf{x})$, at that point. The value of the game increases as the color changes from blue to red.

Figure 6(a) shows a single corner in the plane. The internal angle at the corner is $\frac{2\pi}{3}$. Figure 6(b) shows the optimal trajectories of the players for the corner. The symmetry in the trajectories is due to the fact that termination situations occur symmetrically around a corner.

Figure 7(a) shows a regular hexagon in the plane. Figure 7(b) shows the optimal trajectories of the players for the hexagonal obstacle.

6 Conclusion and Future Work

In this paper, we address a visibility based pursuit-evasion game in an environment containing obstacles. The pursuer and the evader are holonomic having bounded speeds. The pursuer wants to maintain visibility of the evader for maximum

possible time and the evader wants to escape the pursuer's sight as soon as possible. Both the players have knowledge about each others current position. Under this information structure, we present necessary and sufficient conditions for surveillance and escape. We present strategies for the players that are in Nash Equilibrium. The strategies are a function of the value of the game. Using the strategies, we construct a value function by backward integration of the adjoint equations from the termination situations provided by the corners in the environment. From the value functions we recompute the control strategies for the players to obtain optimal trajectories for the players near the termination situation. We show that the optimal strategy for the players is to move on straight lines parallel to each other in opposite directions towards a termination situation. We show a subspace of the optimal trajectories for a point obstacle, a corner and a hexagonal obstacle in space.

In the future, we would like to provide complete solutions for a general polygonal environment. This would include analysis of various kinds of singular surfaces.

References

1. Alexander, S., Bishop, R., Ghrist, R.: Pursuit and evasion in non-convex domains of arbitrary dimensions. In: Proceedings of Robotics: Science and Systems, Philadelphia, USA (August 2006)
2. Alexander, S., Bishop, R., Ghrist, R.: Capture pursuit games on unbounded domains (2008)
3. Başar, T., Olsder, G.J.: Dynamic Noncooperative Game Theory, 2nd edn. SIAM Series in Classics in Applied Mathematics, Philadelphia (1999)
4. Bandyopadhyay, T., Li, Y., Ang Jr., M., Hsu, D.: Stealth Tracking of an Unpredictable Target among Obstacles. In: Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (2004)
5. Bandyopadhyay, T., Li, Y., Ang Jr., M., Hsu, D.: A Greedy Strategy for Tracking a locally Predicatable Target among Obstacles. In: IEEE International Conference on Robotics and Automation, ICRA 2002. Proceedings, pp. 2342–2347 (2006)
6. Becker, C., Gonzalez-Banos, H., Latombe, J., Tomasi, C.: An intelligent observer. In: Proceedings of International Symposium on Experimental Robotics, pp. 94–99 (1995)
7. Bhattacharya, S., Candido, S., Hutchinson, S.: Motion strategies for surviellance. In: Robotics: Science and Systems - III (2007)
8. Bhattacharya, S., Hutchinson, S.: Approximation schemes for two-player pursuit evasion games with visibility constraints. In: Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland (June 2008)
9. Bhattacharya, S., Hutchinson, S.: From strategies to trajectories (2008), http://www-cvr.ai.uiuc.edu/~sbhattac/comp.pdf
10. Efrat, A., Gonzalez-Banos, H., Kobourov, S., Palaniappan, L.: Optimal strategies to track and capture a predictable target. In: IEEE International Conference on Robotics and Automation, ICRA 2003. Proceedings, vol. 3 (2003)
11. Fabiani, P., Latombe, J.: Tracking a partially predictable object with uncertainty and visibility constraints: a game-theoretic approach. Technical report, Univeristy of Stanford (December 1998), http://underdog.stanford.edu/ (cited on page 76)
12. Fleming, W.H.: The convergence problem for differential games. Journal for Mathematical Analysis and Applications 3, 102–116 (1961)

13. Fleming, W.H.: The convergence problem for differential games. Advances in Game Theory. Annals of Mathematics Studies, vol. 52, pp. 195–210 (1964)
14. Gonzalez-Banos, H., Lee, C., Latombe, J.: Real-time combinatorial tracking of a target moving unpredictably among obstacles. In: IEEE International Conference on Robotics and Automation, ICRA 2002. Proceedings, vol. 2 (2002)
15. Isaacs, R.: Differential Games. Wiley, New York (1965)
16. Kopparty, S., Ravishankar, C.V.: A framework for pursuit evasion games in rn. Inf. Process. Lett. 96(3), 114–122 (2005)
17. LaValle, S.M., Gonzalez-Banos, H.H., Becker, C., Latombe, J.C.: Motion strategies for maintaining visibility of a moving target. In: 1997 IEEE International Conference on Robotics and Automation, 1997. Proceedings, Albuquerque, NM, USA, April 1997, vol. 1, pp. 731–736 (1997)
18. Lewin, J.: Differential Games: Theory and Methods for Solving Game Problems with Singular Surfaces. Springer, London (1994)
19. Li, T., Lien, J., Chiu, S., Yu, T.: Automatically generating virtual guided tours. In: Computer Animation Conference, pp. 99–106 (1997)
20. Melikyan, A.A., Hovakimyan, N.V.: Game problem of simple puruit on a two-dimensional cone. Journal of Applied Mathematics and Mechanics 55(5), 607–618 (1991)
21. Melikyan, A.A., Hovakimyan, N.V.: Singular trajectories in the game of simple pursuit in the manifold. Journal of Applied Mathematics and Mechanics 55(1), 42–48 (1991)
22. Melikyan, A.A., Hovakimyan, N.V.: A differential game of simple approach in melikyan. Journal of Applied Mathematics and Mechanics 57(1), 47–57 (1993)
23. Murrieta-Cid, R., Muppirala, T., Sarmiento, A., Bhattacharya, S., Hutchinson, S.: Surveillance strategies for a pursuer with finite sensor range. International Journal of Robotics Research, 1548–1553 (2007)
24. Parsons, T.: Pursuit-evasion in a graph. Theor. Appl. Graphs, Proc. Kalamazoo (1976)

On the Topology of Plans

Michael Erdmann

Abstract. This paper explores a topological perspective of planning. A series of examples and theorems establishes a fundamental coupling between tasks on graphs and simplicial complexes. Planning under uncertainty is one application. The paper introduces *strategy* and *loopback* complexes. The paper's main theorem shows that tasks specified by goal states in nondeterministic graphs have guaranteed solutions if and only if their loopback complexes are homotopic to spheres.

1 Introduction

This paper explores a topological perspective of planning, focusing on graphs with nondeterministic actions. Such graphs capture a wide variety of robotics problems, including motion in the presence of control and sensing uncertainty. The research was motivated by Robert Ghrist's technology transfer between topology and robotics [4, 12, 10, 11], by Steve LaValle's work on information spaces [13, 22, 24, 23], and by workshops on topology and robotics organized by Michael Farber at ETH Zürich in 2003 and 2006 [20].

Topology

Let us frame the word "topology." Sometimes (not here) the term refers to the contact topology of an assembly, or the topology of the configuration space of a robot, or the topology of a network, or the topology of amino acid connections in a protein, to name a few possibilities. In these examples, a *physical structure* has some topology that researchers seek to describe abstractly. While methods from topology certainly have much to say about such problems, they are *not* the focus of this paper. Instead, this paper explores topological descriptions of *tasks* and topological characterizations of *task solvability*.

Michael Erdmann
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, 15213
e-mail: me@cs.cmu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 267–282.
springerlink.com

An analogy might be the relationship of linear algebra to vectors written with specific coordinates. Linear algebra provides abstract techniques for representing and manipulating vectors, independent of coordinates. Anything one can do with matrix and vector notation one can do as well by writing out arrays of coordinates and manipulating the coordinates directly. Indeed, ultimately computations in a computer must work with numbers in some coordinate system. However, the numbers are like trees that obscure the forest. By thinking instead at the abstract level of linear subspaces, kernels, eigenvectors, and so forth, one can recognize fundamental structure. How easy it is to say that a high-dimensional positive definite system always decomposes into orthogonal one-dimensional systems; how cumbersome it would be to convey that truth by numbers or coordinates alone.

Result Flavor

This perspective has precedent in other areas of computer science, such as pioneering work of Herlihy on asynchronous computation. For instance, [15] shows that an asynchronous decision task has a wait-free protocol if and only if there exists a certain color-preserving continuous function between two chromatic simplicial complexes determined by the task. In other words, a computational problem is equivalent to a topological problem. The simplicial complexes reflect the structure of the input and output spaces of the task. The continuous function is a topological constraint between those spaces and thus a constraint on solvability of the decision task.

In topological robotics, Farber initiated a line of work to describe the topological complexity of motion planning problems on a space in terms of the cohomology of that space [9]. This complexity reflects the discontinuities inherent to any controller that maps start and goal configurations of a robot to trajectories between those configurations.

The current paper focuses on tasks that may be specified by goal states in some nondeterministic graph; the task is to attain a goal state from anywhere within the graph. The paper's key result says (roughly) that such a task has a guaranteed solution if and only if a certain simplicial complex associated with the task is homotopic to a sphere of a certain dimension. This observation is motivated by similar results describing the structure of complete directed graphs [3] and strongly connected directed graphs [16]. Indeed, our proof techniques build on the foundations of those two papers.

Contributions

The primary contribution of this paper is the previously unseen structure it reveals in planning problems. The paper shows how the details of a nondeterministic graph can be abstracted away leaving a purely topological description of the task: One can reason about task solvability by thinking in terms of spheres and contractible spaces.

A second contribution is the introduction of *strategy complexes*. A single strategy on a graph is a nondeterministic control law for moving acyclically within some portion of the graph. A strategy complex consists of all possible strategies on a graph. Strategy complexes are useful for reasoning about alternate strategies, such as

backup strategies that a system might need if a selected strategy for accomplishing a task fails unexpectedly.

Third, although beyond the scope of this paper, a large number of ancillary results flow from our characterization of task solvability, among them: (i) One can answer purely topologically the question whether a set of actions is essential to solving a task. (ii) One can recast various questions about tasks into measurements by the Euler characteristic. (iii) One can describe graph composition in terms of the joining of spheres. (iv) One can view Bellman-style backchaining as the repeated enlargement of an existing sphere.

We anticipate that yet more structure exists to be discovered in planning problems via the lens of topology. In particular, we already know that the key theorems of this paper extend, fairly readily, to stochastic graphs. The details, however, are beyond the scope of this short paper. We will instead convey the core ideas by focusing on nondeterministic graphs.

2 An Example: Nondeterminism, Cycles, and Strategies

Imagine an ambulance rescue in an old complicated city, perhaps after an earthquake. There are many opportunities for nondeterminism: Entries into the city might be blocked, maps might be wrong, navigation might lead to circular paths. The left panel of Fig. 1 shows such a toy scenario; let us focus on the final step in which ambulance workers must pass through a narrow opening to reach their patient, as in the right panel. Something might go wrong, a collapse of some sort, forcing the rescue workers to either side, as in Fig. 2. The rescuers may then take additional steps around buildings bounding the original narrow opening to reach their patient.

We can model this scenario using the graph[1] in the left panel of Fig. 3. The action to move from `A` (ambulance) to `X` (patient) might nondeterministically lead to `X` but perhaps also to `B` or `C`, depending on whether and how a collapse occurs on the direct path from `A` to `X`. In the example, there then are deterministic actions from either `B` or `C` to `X`. Such a graph is essentially a compressed AND/OR graph [1].

We can now represent the *strategy* just described as a triangle (right panel of Fig. 3). The triangle or strategy is effectively a control law. The vertices of the triangle are the individual actions to be executed at any particular location during the rescue operation. So, for instance, the control law says "When at location `B`, execute the action `B` → `X`."

Strategies Should Avoid Cycles

Perhaps it is also possible to move from `B` to `C`, as in Fig. 4, and vice versa. We can augment our graph to include these actions, as shown in the left panel of Fig. 5.

One must be careful *not* to include both of these new actions, `B` → `C` and `C` → `B`, together in a control law. Otherwise, the rescuers (who could be robots, not humans)

[1] This paper depicts nondeterministic actions as follows: in a graph, by directed edges tied together with a circular arc; in a strategy complex, by multiple outcomes to the right of an "arrow" (→).

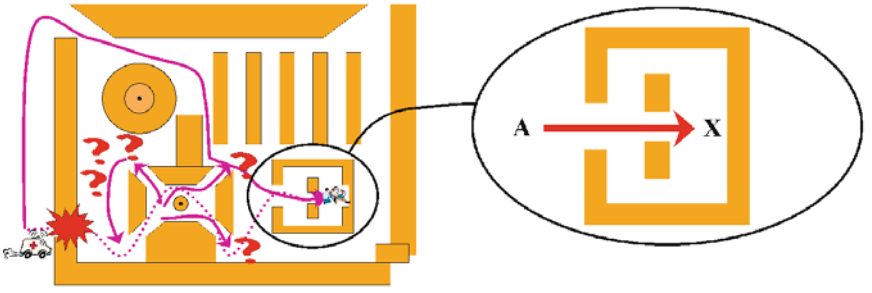


Fig. 1. Nondeterministic events hamper rescue effort.

Fig. 2. Direct access might be blocked.

might cycle forever between B and C, never reaching their patient at X. Instead, there now exist several distinct strategies for reaching X. These are represented by the two tetrahedra in the right panel of Fig. 5. Observe that the two tetrahedra intersect in a triangle that is the original triangle from Fig. 3, but there is no simplex that simultaneously includes the two actions B → C and C → B.

Strategy Complex

The *strategy complex* of a nondeterministic graph consists of all sets of actions of the graph that cannot give rise to cyclic paths in the underlying directed graph. We will refer to such sets of actions as *acyclic*. Each acyclic set of actions is a *simplex* of

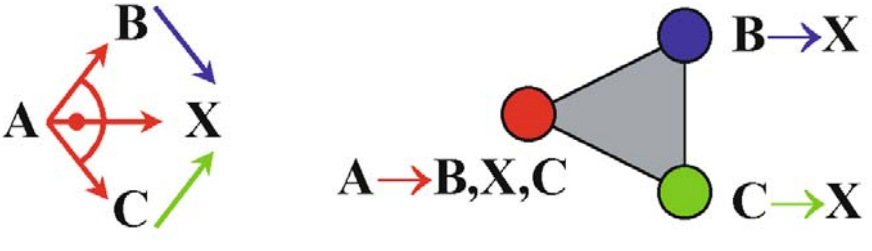


Fig. 3. Nondeterministic graph and strategy complex modeling the motions of Fig. 2.

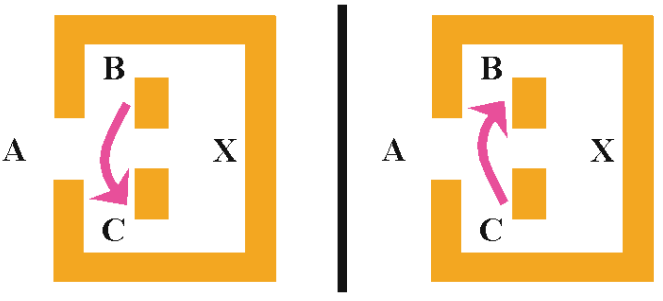


Fig. 4. Actions that move between the far locations B and C.

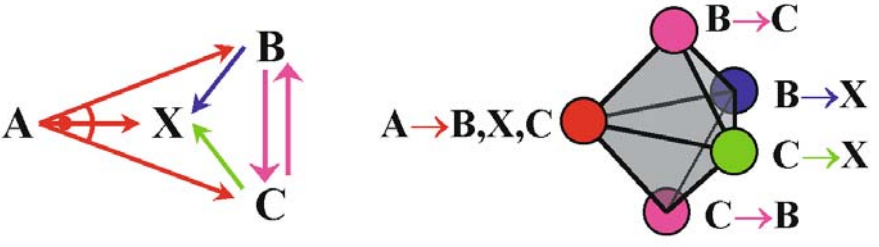


Fig. 5. With additional (potentially cycle-inducing) actions, the strategy complex now contains two tetrahedra.

the strategy complex. For background on simplicial complexes see [21, 2]. Section 5 contains a brief summary as well.

The two tetrahedra we just mentioned generate the strategy complex of the rescue graph of Fig. 5. Each tetrahedron represents a strategy (or control law or plan) consisting of actions that may be executed without accidentally creating cycles in the graph. Each triangle or edge or vertex of one of these tetrahedra, formed from a subset of the actions comprising the tetrahedron, also represents some strategy (perhaps with a different goal).

The semantics of the top tetrahedron of Fig. 5 are:

- When at A, execute the action $\mathtt{A} \to \mathtt{B}, \mathtt{X}, \mathtt{C}$.
- When at B, execute either the action $\mathtt{B} \to \mathtt{X}$ or the action $\mathtt{B} \to \mathtt{C}$. It does not matter which; pick one, perhaps nondeterministically.
- When at C, execute the action $\mathtt{C} \to \mathtt{X}$.

Nondeterminism appears in both the outcomes and choices of the strategy:

1. Nature acts as an adversary during execution of the action $\mathtt{A} \to \mathtt{B}, \mathtt{X}, \mathtt{C}$, making the outcome uncertain. This can be bad.
2. The system has available multiple actions at location B. This is good. The bigger a strategy simplex, the better. The system can simply choose to ignore the extra action $\mathtt{B} \to \mathtt{C}$ if it wishes and instead always move $\mathtt{B} \to \mathtt{X}$, perhaps appropriate if speed of rescue is important.

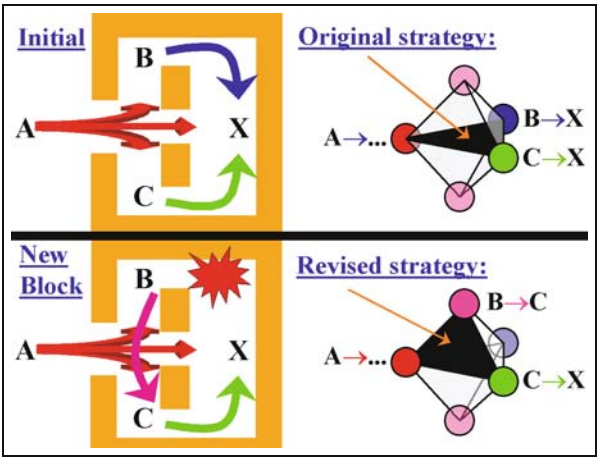


Fig. 6. The original triangular strategy of Fig. 3 is a face of both tetrahedra in the enlarged strategy complex of Fig. 5. If one of the original actions fails unexpectedly, it may be possible to find a revised backup strategy in the strategy complex.

This second observation, that the tetrahedral strategy has extra actions, is useful. It tells us that if an unmodeled event occurs and the action `B` → `X` from the original strategy of Fig. 3 no longer seems appealing, then there is a *backup*, via the sequence of actions `B` → `C`; `C` → `X`. See Fig. 6 for a comparison.

Comment: The graph just described is ubiquitous in planning under uncertainty. For instance, it captures the essential difficulty of the peg-in-hole problem [19].

3 More Examples: Graphs, Loopbacks, and Spheres

3.1 Two Graphs and Their Strategy Complexes

Let us consider an even simpler example, to build intuition. The graph on the left of Fig. 7 is a standard directed graph. Each edge of the directed graph is a possible "action" the system could perform, moving it from some state[2] of the graph to some other state.

The strategy complex of this graph is shown in the right panel of Fig. 7. The biggest simplex one could possibly expect to see in the strategy complex would be a tetrahedron, consisting of all four actions present in the directed graph. However, two of the actions, namely `1` → `2` and `2` → `1`, could give rise to a cycle in the graph, so no simplex of the strategy complex can contain both these actions. The complex is in fact generated by two triangles.

[2] We refer to a graph node as a "state" and reserve the term "vertex" for singleton simplices in simplicial complexes.

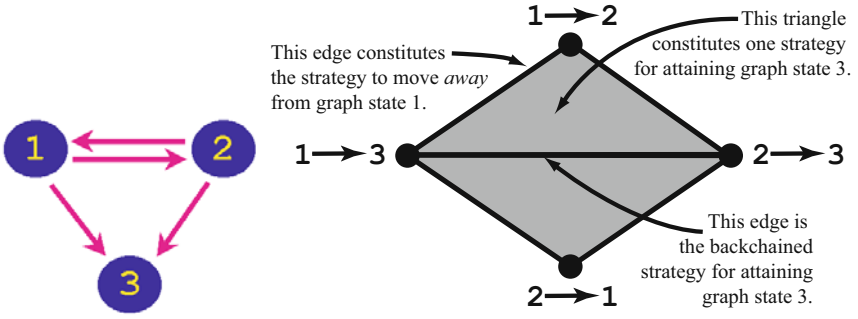


Fig. 7. The graph on the left defines the strategy complex shown on the right.

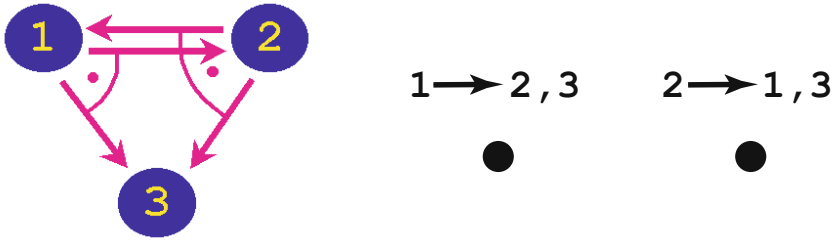


Fig. 8. The graph on the left has two nondeterministic actions that could create a cycle, so the strategy complex on the right consists of two isolated points.

The two triangles, as well as three of the five edges in the complex, constitute strategies for attaining state 3 in the graph. The central edge, consisting of the actions $\{1 \to 3;\ 2 \to 3\}$, is the strategy one would obtain by backchaining from state 3.

Observe that a strategy complex may contain strategies for a variety of goals. For instance, the top left edge of the complex in Fig. 7, comprising the set of actions $\{1 \to 3;\ 1 \to 2\}$, is a strategy that simply says "move *away* from state 1."

For contrast, consider the graph of Fig. 8. It contains two actions, one each at states 1 and 2. Each action has two nondeterministic outcomes. The two actions cannot appear together as a simplex since, depending on the actual nondeterministic transitions at runtime, these actions could cause cycling in the graph between states 1 and 2. As a result, the strategy complex consists of two isolated points, representing the two strategies "move away from state 1" and "move away from state 2." In particular, *there are no strategies guaranteed to attain state* 3 from the other two states.

3.2 Loopback Graphs and Complexes

Now let us modify the graph of Fig. 7. We will add artificial deterministic transitions from state 3 to each of states 1 and 2, which we call *loopback actions*. Think of

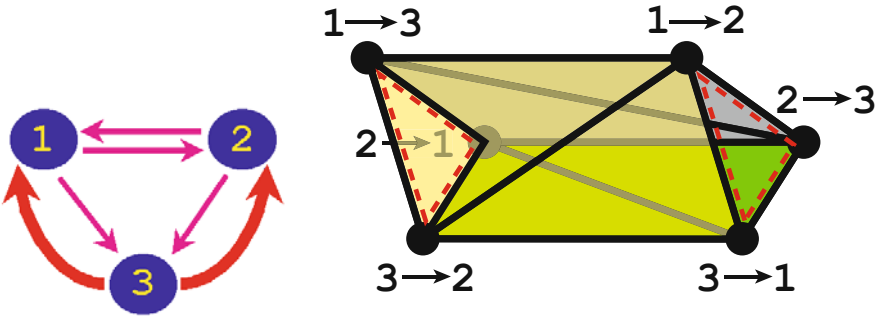


Fig. 9. A loopback graph and loopback complex associated with the graph of Fig. 7. The complex contains 6 vertices, 12 edges, and 6 triangles (shaded). The two triangular endcaps outlined in dashed red are *not* part of the complex, since each gives rise to a cycle in the graph. The complex is homotopic to S^1.

these actions as "topological electrodes" that allow us to measure whether the graph contains a guaranteed strategy for attaining state 3 from all states.

The left panel of Fig. 9 shows the resulting *loopback graph*. Now imagine constructing the strategy complex associated with that graph, as shown in the right panel of the figure. We refer to it as a *loopback complex* of our original graph. The complex in this case looks roughly like a polygonal cylinder. The complex is homotopic to a circle[3], as represented by either open end of the cylinder. One says that the loopback complex has the *homotopy type of a circle*. Notice that one cannot continuously deform the complex (within itself) into a point. This is crucial. Homotopy type is an equivalence relation on topological spaces. Spaces that have the same homotopy type as a point are said to be *contractible*. Spheres are not contractible. (A circle is a one-dimensional sphere.)

In contrast, suppose we add both possible loopback actions at state 3 to the graph of Fig. 8. The resulting graph and loopback complex are show in Fig. 10. Now the loopback complex is homotopic to a point; one can continuously deform it within itself to a point.

The Punch Line: No matter how complicated the nondeterministic graph, if we add all loopback actions to it that transition from some state `s` to the remaining states, then the resulting loopback complex will *always* be homotopic either to a sphere or to a point. A sphere tells us that there is a strategy guaranteed to attain state `s` from all states in the graph; a point tells us that no such strategy exists. See Theorem 1 of Section 5.

[3] "Homotopic" means, in this case, that the complex, viewed as a topological space, can be continuously deformed within itself into a subspace that is topologically a circle. For a more precise and general definition, see [21, 2].

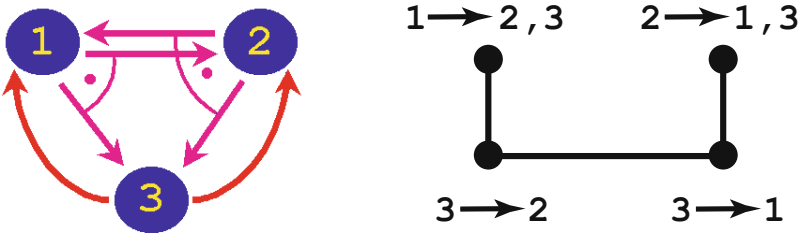


Fig. 10. A loopback graph and loopback complex associated with the graph of Fig. 8. The complex is contractible.

3.3 A Loopback Complex Homotopic to S^2

As a third example, the graph of Fig. 11 contains three deterministic actions transitioning to state 4, one each from states 1, 2, and 3. Adding the loopbacks to this graph at state 4 adds three more actions. The associated loopback complex is a triangulation of the two-dimensional sphere. Each of the six actions of the loopback graph is a vertex in this triangulation. The complex further contains 12 edges and 8 triangles. The triangle $\{1 \to 4;\ 2 \to 4;\ 3 \to 4\}$ represents the strategy for attaining state 4 from all states. The triangle $\{4 \to 1;\ 4 \to 2;\ 4 \to 3\}$ represents the (loopback) strategy "move away from state 4."

Removing an action, say $1 \to 4$, from the original graph is the same as puncturing the sphere, thereby producing a contractible loopback complex. Contractibility is consistent with the fact that the modified graph would no longer contain a strategy for attaining state 4 from all states.

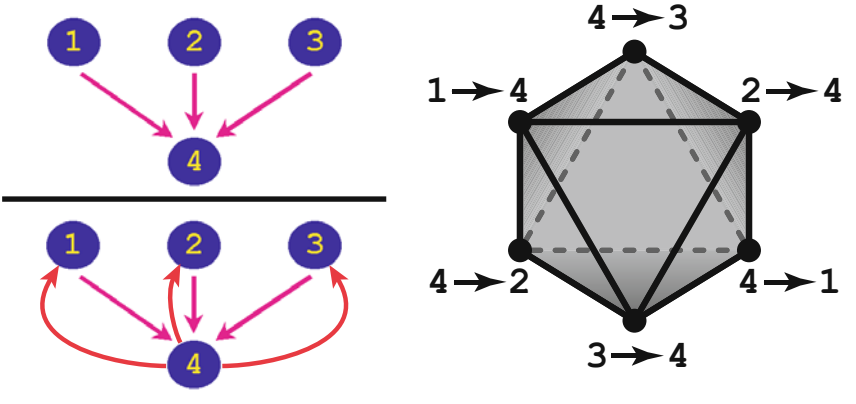


Fig. 11. The graph in the top left contains three deterministic transitions to state 4, one each from the other three states. The graph in the bottom left is this same graph along with loopback actions from state 4 to the other three states. The resulting loopback complex shown on the right is homotopic to S^2.

4 Topological Thinking in Partially Observable Spaces

This section shows by example how the topological characterization of task solvability may help decide whether a task has a guaranteed solution.

4.1 Inferring Task Unsolvability from Duality

This example shows how the topology of strategy complexes may imply task unsolvability.

The spherical characterization of task solvability has a dual statement (Theorem 2 of Section 5), with the following necessary condition: If the loopback complex associated with a particular goal is homotopic to a sphere, then the subcomplex formed by *all* actions at *any* nonempty set of nongoal states must be contractible. Intuitively, this subcomplex is obtained by slicing the sphere into two well-behaved parts, then discarding the part containing the loopback actions. What remains must be contractible. Failure of this condition is evidence of a potentially *inescapable cycle*, indicating that the task has no guaranteed solution.

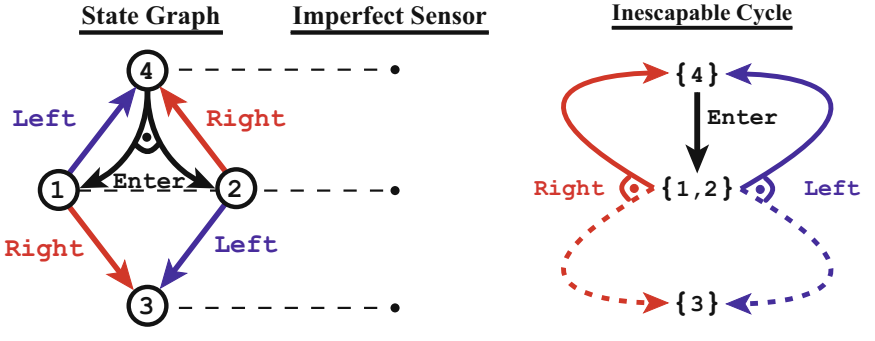


Fig. 12. A graph, an imperfect sensor, and a potentially inescapable cycle between knowledge states {4} and {1,2} (indicated by the solid arrows).

The state graph of Fig. 12 might model a robot moving in either of two corridors (states 1 and 2). In any one corridor the robot can move RIGHT or LEFT. Atriums connect the corridors at either ends. The task is to reach one particular atrium (state 3). Entry into the corridors from the other atrium (state 4) is imprecise. The gray triangle in the strategy complex of Fig. 13 constitutes a strategy for accomplishing this task, assuming perfect sensing. The strategy is to ENTER from state 4, move RIGHT from 1, LEFT from 2.

Now imagine a robot controller unable to distinguish the two corridors (states 1 and 2) based on sensing alone. The task no longer has a guaranteed solution. One could see this in a variety of ways, for instance, by explicitly constructing the robot's

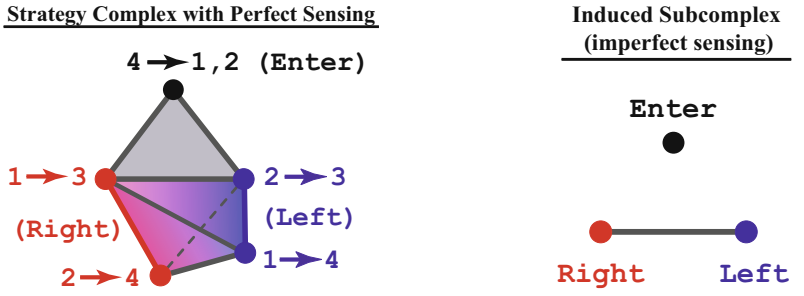


Fig. 13. With perfect sensing, the strategy complex of the state graph of Fig. 12 consists of a triangle joined to an edge of a tetrahedron. With sensing ambiguity at states 1 and 2, this strategy complex collapses, inducing a non-contractible subcomplex of strategies in knowledge space that signals an inescapable cycle.

knowledge space[4], part of which is shown in the right panel of Fig. 12. Let us take a related but more topological perspective.

One does not need to construct the knowledge space directly. Instead, one can reason about strategies. Consider the strategy complex of the original graph (left panel, Fig. 13). As we will see, this complex induces a non-contractible subcomplex of strategies in knowledge space (right panel, Fig. 13) that violates the duality theorem.

Let us focus on the actions at two key knowledge states, namely $\{4\}$, representing certainty that the system is at state 4, and $\{1,2\}$, representing uncertainty as to whether the system is at state 1 or 2. The tetrahedron of the original complex describes the strategies possible with perfect sensing at the graph states 1 and 2. The tetrahedron collapses to an edge under sensing ambiguity. This edge represents the two actions, RIGHT and LEFT, possible at state $\{1,2\}$ in knowledge space. In the original complex, only a portion of the tetrahedron joins with the action ENTER; action RIGHT at state 2 and action LEFT at state 1 do not join with ENTER. In the knowledge space complex, the edge $\{$RIGHT, LEFT$\}$ consequently *cannot join* with the action ENTER.

We have thus exhibited a non-contractible complex describing the strategies available at a subset of knowledge space. This means, *no matter what the surrounding knowledge space might look like*, there can exist no guaranteed solution to the task of attaining state 3 in the presence of control uncertainty at state 4 and sensing confusion between states 1 and 2.

4.2 Hypothesis-Testing and Sphere Suspension

This example shows how the topology of loopback complexes may imply task solvability.

[4] *Knowledge space* is the graph whose states are sets of potential robot locations consistent with the robot's sensing and action history at runtime [8, 5, 7, 18].

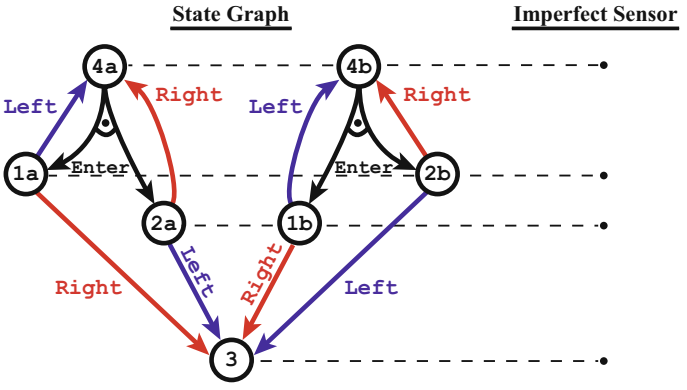


Fig. 14. An imperfect sensor stratifies a graph into two subgraphs (the "a wing" and the "b wing"), over each of which sensing is effectively perfect.

Fig. 14 shows a variant of the previous example. Once again there is some control uncertainty, once again the sensor cannot distinguish certain corridors. The system cannot move reliably to state 3 using pure feedback control (that is, using sensing alone, without history). For some sensor values, neither of the actions RIGHT or LEFT will make progress toward the goal 3 at all states consistent with the sensor value.

Fortunately, this time the confusable corridors lie in different "wings" of the building. The sensor stratifies the graph into two subgraphs, both containing the goal. Within each subgraph the sensor is effectively perfect and each subgraph contains a strategy guaranteed to attain the goal from anywhere within that subgraph.

Whenever a sensor stratifies a graph into subgraphs in this manner, there exists a hypothesis-testing strategy for attaining the goal from anywhere in the overall graph. Hypothesis-testing means: The system assumes it is in one of the subgraphs; it commands actions and interprets sensor readings as if it really were in that subgraph, but also verifies consistency between predicted motions and observed sensor readings. If an inconsistency occurs, the hypothesis of being in that subgraph has been falsified and the system moves on to another hypothesis. Intuitively, this strategy eventually converges at the goal.

Hypothesis-testing is a strategy in knowledge space, but one does not need to construct knowledge space. In general, knowledge space may contain additional, possibly shorter, strategies. For the example of Fig. 14, hypothesis-testing is effectively the only strategy.

There is a short topological argument that hypothesis-testing converges: Hypothesis-testing amounts to repeated sphere suspension.[5] Further details: The graph H_i describing motions under the i^{th} hypothesis is equivalent to the subgraph

[5] A *suspension* of a complex is another complex formed by joining each simplex with each of two new vertices [21, 14]. For instance, the complex in Fig. 5 is a suspension of the complex in Fig. 3. The key property relevant here is that the suspension of a sphere of any dimension is another sphere, of one higher dimension.

G_i being hypothesized, except that some actions may move nondeterministically to a new state $\perp_i$, signaling falsification of the hypothesis. We also add an "action" from $\perp_i$ to the goal, as explained below. Without loss of generality[6], every action of G_i, and thus H_i, contains a *nondeterministic* transition to the goal. The loopback complex Γ_{H_i} of H_i is then a suspension, formed by joining the loopback complex Γ_{G_i} of G_i with the loopback action $\texttt{goal} \rightarrow \perp_i$ and the action $\perp_i \rightarrow \texttt{goal}$. Since G_i contains a strategy guaranteed to attain the goal from any state in G_i, Γ_{G_i} is homotopic to a sphere. Consequently so is Γ_{H_i}, proving that hypothesis-testing converges.

What is the mysterious action $\perp_i \rightarrow \texttt{goal}$? It is the inductive-hypothesis that there exists a strategy for attaining the goal once the i^{th} graph-hypothesis has been falsified! (The base case is similar, except that there is no need for a state $\perp_i$. The final graph-hypothesis is certain.)

Two comments: (1) One can make a very similar argument directly at the graph level, further underscoring the parallel between spheres and task solvability. That parallel shows as well that backchaining is much like repeated sphere suspension. (2) Hypothesis-testing is related to randomization [5, 6, 7].

5 Mathematical Details

An *(abstract) simplicial complex* Σ is a collection of finite sets, such that if σ is in Σ then so is every subset of σ [21]. The elements of Σ are called *simplices*; the elements of a simplex and singleton simplices are both called *vertices*. We permit the empty simplex $\emptyset$, for combinatorial simplicity [3, 17]. The complex $\{\emptyset\}$, consisting solely of the empty simplex, is the *empty complex*. It is also the sphere of dimension -1. The complex $\emptyset$, consisting of no simplices, is the *void complex*. All complexes in this paper are finite. Any nonvoid finite complex has a geometric realization in some Euclidean space, with relative topology the same as its polytope topology [21]. Thus we view Σ as a topological space.

A *nondeterministic graph* $G = (V, \mathscr{A})$ is a set of *states* V and a collection of *actions* $\mathscr{A}$. Each $A \in \mathscr{A}$ is a nonempty set of *directed edges* $\{(v,u_1),(v,u_2),\ldots\}$, with v and all u_i in V; v is A's *source* and each u_i is a nondeterministic *target*. Distinct actions may have overlapping or identical edge sets. All graphs and actions in this paper are finite. A *possible path of length* k in G is a sequence of states $v_0, v_1, \ldots, v_k$ such that $(v_i, v_{i+1}) \in A_i$, for some actions $\{A_i\}_{i=0}^{k-1} \subseteq \mathscr{A}$. G is *acyclic* if none of its possible paths have $v_0 = v_k$ with $k \geq 1$.

Any $\mathscr{B} \subseteq \mathscr{A}$ defines a nondeterministic *subgraph* $H_{\mathscr{B}} = (V, \mathscr{B})$ of G. Given a desired *stop* state $s \in V$, we say that G *contains a complete guaranteed strategy for attaining* s if there is some set of actions $\mathscr{B} \subseteq \mathscr{A}$ such that $H_{\mathscr{B}}$ is acyclic and $\mathscr{B}$ contains at least one action with source v for every $v \in V \backslash \{s\}$. Observe that $\mathscr{B}$ cannot contain actions with source s and that every possible path in $H_{\mathscr{B}}$ with $v_k \neq s$ can be extended to a longer path. Iterating, this process converges at s, since $H_{\mathscr{B}}$ is

[6] Adding these transitions does not affect the existence of strategies for attaining the goal, but focuses on the topologically significant simplices in the complexes.

acyclic. We refer interchangeably to both $H_{\mathscr{B}}$ and $\mathscr{B}$ as being *acyclic*, *a complete guaranteed strategy*, etc.

Given a nondeterministic graph $G = (V, \mathscr{A})$ with $V \neq \emptyset$, let Δ_G be the simplicial complex whose simplices are the acyclic sets of actions $\mathscr{B} \subseteq \mathscr{A}$. If $V = \emptyset$, let $\Delta_G = \emptyset$. Observe that no action of G with a self-loop can appear in Δ_G. Given desired stop state $s \in V$, let $G_{\leftarrow s}$ be the nondeterministic graph identical to G except that all actions with source s have been discarded, replaced instead by $(|V|-1)$-many *loopback* actions $\{(s,v)\}$, each consisting of a single edge from s to some v, with v ranging over $V \backslash \{s\}$. Define $\Delta_{G_{\leftarrow s}}$ accordingly. Δ_G is the *strategy complex* and $\Delta_{G_{\leftarrow s}}$ is a *loopback complex* of G.

Theorem 1. *Let $G = (V, \mathscr{A})$ be a nondeterministic graph and $s \in V$.*

If G contains a complete guaranteed strategy for attaining s, then $\Delta_{G_{\leftarrow s}}$ is homotopic to the sphere S^{n-2}, with $n = |V|$. Otherwise, $\Delta_{G_{\leftarrow s}}$ is contractible.

Proof. I. Let $\mathscr{B}$ be a complete guaranteed strategy for attaining s and let $\mathscr{A}'$ be the actions of $G_{\leftarrow s}$. We may assume $V = \{1, \ldots, n\}$ and $s = n$. For each $A \in \mathscr{A}'$ define the open polyhedral cone $U_A = \bigcap_{(i,j) \in A} \{\mathbf{x} \in \mathbf{R}^n \mid x_i > x_j\}$. Observe that a set of actions $\{A_1, \ldots, A_k\}$ is acyclic if and only if $U_{A_1} \bigcap \cdots \bigcap U_{A_k}$ is not empty. When nonempty, the intersection is contractible. By the Nerve Lemma [14], $\Delta_{G_{\leftarrow s}}$ therefore has the homotopy type of $\bigcup_{A \in \mathscr{A}'} U_A$. We claim that this union covers all of $\mathbf{R}^n$ except for the line on which all coordinates are equal. Thus it is homotopic to S^{n-2}.

To see coverage: Clearly no point with all coordinates equal can be in the union. The cones determined by the loopback actions cover all points $\mathbf{x} \in \mathbf{R}^n$ for which $x_n > x_i$, some i. Suppose some $\mathbf{x}$ in $\mathbf{R}^n \backslash \{x_1 = \cdots = x_n\}$ is not inside any U_A. Then $x_i \geq x_n$ for all i, with at least one $x_i > x_n$. Some action $B \in \mathscr{B}$ has that i as a source. $\mathscr{B} \subseteq \mathscr{A}'$, so $\mathbf{x} \notin U_B$, meaning there is some target j of B such that $x_j \geq x_i > x_n$. Repeating this argument with j, etc., produces an arbitrarily long and thus cyclic possible path in $H_{\mathscr{B}}$. Contradiction.

II. If G does not contain a complete guaranteed strategy for attaining s, then no simplex of $\Delta_{G_{\leftarrow s}}$ contains actions at all states of $V \backslash \{s\}$. For every simplex $\sigma \in \Delta_{G_{\leftarrow s}}$ there is therefore a unique nonempty maximal set τ_σ of loopback actions such that $\sigma \bigcup \tau_\sigma \in \Delta_{G_{\leftarrow s}}$. A standard collapsing argument now shows that $\Delta_{G_{\leftarrow s}}$ is contractible (Lemma 7.6 of [3] is useful). □

The following theorems, stated here without proof, further characterize the topology of strategies on nondeterministic graphs. Notation and definitions are as before. Also, given $G = (V, \mathscr{A})$ and any subset $W \subseteq V$, define the subgraph $G|W = (V, \mathscr{A}|W)$ with $\mathscr{A}|W$ all actions of G whose sources lie in W.

Theorem 2. *G contains a complete guaranteed strategy for attaining $s \in V$ if and only if $\Delta_{G|W}$ is contractible for every nonempty W contained in $V \backslash \{s\}$.*

Theorem 3. *Suppose $n = |V| > 0$. Δ_G is homotopic to S^{n-2} if and only if, for every $v \in V$, G contains a complete guaranteed strategy for attaining v.*

Theorem 4. *For any finite simplicial complex Σ, there is a nondeterministic graph G such that Δ_G is isomorphic as a simplicial complex to* $\mathrm{sd}(\Sigma)$ *(the first barycentric subdivision of Σ).*

Theorems 1 and 2 are dual topological perspectives. Theorem 3 provides a "graph controllability" condition: There exist actions to attain any desired goal state in a graph *despite control uncertainty* (modeled by nondeterminism) precisely when the graph's strategy complex is homotopic to a sphere of a specific dimension. We may therefore view graphs satisfying this condition as nondeterministic analogues of strongly connected directed graphs. Theorem 4 shows that nondeterministic graphs and simplicial complexes are essentially the same topologically.

6 Conclusions

This paper has drawn a parallel between planning and the topology of simplicial complexes. Theorem 1 characterizes the existence of plans purely topologically. Theorem 4 suggests that methods from algebraic topology may offer further new approaches for planning in the presence of uncertainty. The CONTRIBUTIONS section of the INTRODUCTION mentions additional results. In particular, our key theorems generalize to stochastic graphs.

Acknowledgments. This work was sponsored by DARPA under contract HR0011-07-1-0002. This work does not necessarily reflect the position or the policy of the U.S. Government. No official endorsement should be inferred.

Many thanks to the entire "SToMP" group for their feedback and encouragement, particularly to Benjamin Mann, Robert Ghrist, Steven LaValle, and Matt Mason for their leadership and advice, as well as to Robert Ghrist and Shmuel Weinberger for answering my topology questions so quickly and enthusiastically. Many thanks to the WAFR Co-Chairs for all their efforts and support.

References

1. Barr, A., Cohen, P., Feigenbaum, E. (eds.): The Handbook of Artificial Intelligence. William Kaufmann, Inc., Los Altos (1981–1989)
2. Björner, A.: Topological methods. In: Graham, R., Grötschel, M., Lovász, L. (eds.) Handbook of Combinatorics, vol. V.II, pp. 1819–1872. North-Holland, Amsterdam (1995)
3. Björner, A., Welker, V.: Complexes of directed graphs. SIAM J. Discrete Math. 12(4), 413–424 (1999)
4. de Silva, V., Ghrist, R.: Coordinate-free coverage in sensor networks with controlled boundaries via homology. Intl. J. Rob. Res. 25(12), 1205–1222 (2006)
5. Erdmann, M.: On Probabilistic Strategies for Robot Tasks. PhD thesis, EECS, MIT, Cambridge, MA (1989), Also available as Technical Report AI-TR-1155
6. Erdmann, M.: Randomization in robot tasks. Intl. J. Rob. Res. 11(5), 399–436 (1992)
7. Erdmann, M.: Randomization for robot tasks: Using dynamic programming in the space of knowledge states. Algorithmica 10(2-4), 248–291 (1993)

8. Erdmann, M., Mason, M.: An exploration of sensorless manipulation. IEEE J. Robotics and Automation 4(4), 369–379 (1988)
9. Farber, M.: Topological complexity of motion planning. Discrete and Computational Geometry 29(2), 211–221 (2003)
10. Ghrist, R., LaValle, S.: Nonpositive curvature and Pareto-optimal coordination of robots. SIAM J. Control & Opt. 45(5), 1697–1713 (2006)
11. Ghrist, R., O'Kane, J., LaValle, S.: Computing Pareto optimal coordinations on roadmaps. Intl. J. Robotics Research 24(11), 997–1010 (2005)
12. Ghrist, R., Peterson, V.: The geometry and topology of reconfiguration. Adv. Appl. Math. 38, 302–323 (2007)
13. Guilamo, L., Tovar, B., LaValle, S.: Pursuit-evasion in an unknown environment using gap navigation trees. In: Proc. IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 3456–3462 (2004)
14. Hatcher, A.: Algebraic Topology. Cambridge University Press, Cambridge (2002)
15. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. J. ACM 46(6), 858–923 (1999)
16. Hultman, A.: Directed subgraph complexes. Elec. J. Combinatorics 11(1), R75 (2004)
17. Jonsson, J.: Simplicial Complexes of Graphs. PhD thesis, Department of Mathematics, KTH, Stockholm, Sweden (2005)
18. LaValle, S.: Planning Algorithms. Cambridge University Press, New York (2006)
19. Lozano-Pérez, T., Mason, M., Taylor, R.: Automatic synthesis of fine-motion strategies for robots. Intl. J. Robotics Research 3(1), 3–24 (1984)
20. Mackenzie, D.: ROBOTICS: Topologists and Roboticists Explore an 'Inchoate World'. Science 301(5634), 756 (2003)
21. Munkres, J.: Elements of Algebraic Topology. Addison-Wesley, Menlo Park (1984)
22. O'Kane, J., LaValle, S.: On comparing the power of robots. Intl. J. Robotics Research 27(1), 5–23 (2008)
23. Tovar, B., Murrieta, R., LaValle, S.: Distance-optimal navigation in an unknown environment without sensing distances. IEEE Transactions on Robotics 23(3), 506–518 (2007)
24. Tovar, B., Yershova, A., O'Kane, J., LaValle, S.: Information spaces for mobile robots. In: Proc. Intl. Workshop on Robot Motion and Control (2005)

Part VI
Geometric Sensing

Mirror-Based Extrinsic Camera Calibration

Joel A. Hesch, Anastasios I. Mourikis, and Stergios I. Roumeliotis

Abstract. This paper presents a method for determining the six degree-of-freedom transformation between a camera and a base frame of interest. A planar mirror is maneuvered so as to allow the camera to observe the environment from several viewing angles. Points, whose coordinates in the base frame are known, are observed by the camera via their reflections in the mirror. Exploiting these measurements, we determine the camera-to-base transformation analytically, without assuming prior knowledge of the mirror motion or placement with respect to the camera. The computed solution is refined using a maximum-likelihood estimator, to obtain high-accuracy estimates of the camera-to-base transformation and the mirror configuration for each image. We validate the accuracy and correctness of our method with simulations and real-world experiments.

1 Introduction

Cameras are utilized in a wide variety of applications ranging from surveillance and crowd monitoring, to vision-based robot localization. In order to obtain meaningful geometric information from a camera, two calibration procedures must be completed. The first is intrinsic calibration, that is, determining the internal camera parameters (e.g., focal length, principal point, and skew coefficients), which affect the image measurements. The second is extrinsic calibration, which is the process of computing the transformation between the camera and a base frame of reference. In a surveillance application, the base frame may be the room or building coordinate system, whereas on a mobile robot, the base frame could be the

Joel A. Hesch and Stergios I. Roumeliotis
University of Minnesota, Minneapolis, MN 55455, USA
e-mail: {joel,stergios}@cs.umn.edu

Anastasios I. Mourikis
University of California, Riverside, CA 92521, USA
e-mail: mourikis@ee.ucr.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 285–299.
springerlink.com

robot-body frame. Several authors have addressed extrinsic calibration of a camera to another sensor (e.g., for odometry-to-camera [13], inertial measurement unit (IMU)-to-camera [14], or laser scanner-to-camera [1, 21]). These exploit measurements from both sensors to determine their mutual transformation. However, very little attention has been devoted to determining the *camera-to-base* transformation, for a generic base frame.

In this paper, we deal exclusively with extrinsic camera calibration. Our objective is to determine the camera-to-base transformation from observations of points whose coordinates in the base frame are known. We consider the most limiting case, in which the known points do not lie within the camera's field of view but can only be observed using a planar mirror. We maneuver the mirror in front of the camera to provide multiple views of the points. In our formulation, no prior information about the mirror motion or placement with respect to the camera is assumed. The configuration of the mirror and the camera-to-base transformation are treated as unknowns to be computed from the observations. The main contribution of this paper is an algorithm for determining the camera-to-base transformation *analytically*, which requires a minimum of 3 non-collinear points tracked in 3 images.

A direct approach to extrinsic camera calibration is to utilize all of the measurements in a maximum-likelihood estimator (MLE) for computing the unknown transformation [7]. This takes the form of a nonlinear least-squares problem, which seeks to iteratively minimize a nonconvex function of the unknown variables. While appealing for its ease of implementation, this method has two drawbacks. First, without an accurate initial guess, the minimization process may take several iterations to converge, or even fail to find the correct solution. Second, the MLE provides no framework for studying the minimal measurement conditions required to compute a solution. To address the first issue, in the method presented in this paper we first determine the transformation analytically, and then employ an MLE to refine the computed solution (cf. Section 3). Moreover, we determine the minimal number of measurements required for a unique solution. Finally, in Appendix 2 we comment on the extension of this work to robot-body 3D reconstruction.

2 Related Work

Before presenting our method, we first review the related work, which falls into two categories: (i) hand-eye calibration, and (ii) catadioptric systems. Hand-eye calibration is the process of determining the six degree-of-freedom (6 d.o.f.) transformation between a camera and a tool, which are both mounted on a robot manipulator [20, 2]. The hand-eye problem is solved by correlating the measurements of the camera and the encoders, which measure displacements of the robot joints. This process determines the pose of the camera with respect to the robot base. Subsequently, the camera-to-tool transformation is calculated by combining the estimated camera-to-robot-base transformation, and the robot-base-to-tool transformation, which is assumed to be known. This necessitates the availability of precise technical drawings,

and limits the applicability of these methods, since they cannot determine the camera-to-base transformation for a generic base frame.

Next, we turn our attention to catadioptric systems, which are employed to perform synthetic multiple-view vision. Methods have been presented utilizing a single camera and planar [3, 8], or conic mirrors [9]. Others accomplish stereo vision with reflections from free-form surfaces [22], and a trinocular mirror-based vision system also exists [17]. Additionally, stereo is achieved with a static camera and a moving mirror [11], or with a moving camera and two stationary spherical mirrors of known radii [15]. While the use of mirror reflections relates these approaches to our work, the key difference is that we do *not* perform synthetic stereo, i.e., only a single observation of each point is available in each image.

Jang *et al.* demonstrate a system for 3D scene reconstruction using a moving planar mirror [10]. Exploiting a combination of fiducial points on the mirror and vanishing points in the reflections, they solve for the position of the mirror with respect to the camera. The 3D scene is determined based on synthetic stereo from multiple reflections. In contrast to this approach, we do not assume that the dimensions of the mirror, or its position with respect to the camera, are available. Finally, Kumar *et al.* determine the transformations between multiple cameras with non-overlapping fields of view, using mirror reflections of a calibration grid [12]. They require 5 views (per camera) of the calibration pattern to form a set of linear constraints, which are solved for the unknown transformations. In contrast to [12], our method requires only 3 images, each containing observations of 3 known points, to determine the camera-to-base transformation analytically.

3 Computing the Transformation

In this section, we describe our approach for analytically determining the transformation between the camera frame, $\{C\}$, and a frame of interest, $\{B\}$, from observations of 3 points whose coordinates in $\{B\}$ are known. Frame $\{B\}$ is arbitrary and without loss of generality, we will refer to $\{B\}$ as the "base frame." Example base frames vary by application, and may include: (i) the robot-body frame, if the camera is mounted on a robot, (ii) the room or building frame, if the camera is utilized in a surveillance application, and (iii) the rig mount, if the camera is part of a stereo pair.

We address the most limiting scenario in which the points are only visible through reflections in a planar mirror that is moved in front of the camera to provide multiple views of the scene. We exploit these observations to compute the transformation between $\{B\}$ and $\{C\}$, without knowledge of the mirror's placement or motion with respect to the camera (cf. Algorithm 1). In what follows, we present the measurement model and discuss its relation with the three-point pose estimation problem (P3P). We comment cases where a unique solution does not exist, and present an analytical method to compute the unknown transformation from a minimum of 3 points observed in 3 images that differ by rotations about two axes. Lastly, we summarize a maximum-likelihood approach for refining the computed transformation, a detailed discussion of which is available in [7].

Algorithm 1. Computing the Camera-to-Base Transformation

Input: Observations of 3 points tracked in N_c images
Output: Camera-to-base transformation $\{{}^C_B\mathbf{R}, {}^C\mathbf{p}_B\}$
  for each image in N_c **do**
    Convert to three-point pose estimation problem (P3P)
    Solve P3P to obtain combined homogeneous/reflection transformation: $\{\mathbf{A},\mathbf{b}\}_k$
  end for
  for each triplet of solutions $\{\mathbf{A},\mathbf{b}\}_k$, $\{\mathbf{A},\mathbf{b}\}_{k'}$, $\{\mathbf{A},\mathbf{b}\}_{k''}$ **do**
    Compute mirror configurations from (14)
    Compute camera-to-base rotation ${}^C_B\mathbf{R}$ from (23)
    Compute camera-to-base translation ${}^C\mathbf{p}_B$ from (16)
  end for
  Utilize clustering to select the correct solution $\{{}^C_B\mathbf{R}, {}^C\mathbf{p}_B\}$
  Refine the solution using a maximum-likelihood estimator

3.1 Measurement Model

First, we present the measurement model that describes each of the camera observations. To simplify the presentation, in this section we focus on the case of a single point, observed in a single image. Consider a point $\mathbf{p}$, whose position with respect to frame $\{B\}$, ${}^B\mathbf{p}$, is known[1]. We seek to express the point $\mathbf{p}$ in the camera reference frame $\{C\}$. From geometry (cf. Fig. 1) we have two constraint equations:

$$ {}^C\mathbf{p}' = {}^C\mathbf{p} + 2d_p\,{}^C\mathbf{n} \tag{1} $$

$$ d_p = d - {}^C\mathbf{n}^{\mathrm{T}}\,{}^C\mathbf{p} \tag{2} $$

where ${}^C\mathbf{p}'$ is the reflection of ${}^C\mathbf{p}$, ${}^C\mathbf{n}$ is the mirror normal vector expressed in the camera frame, d is the distance between the mirror and the camera, and d_p is the distance between the mirror and the known point (both distances are defined along the mirror normal vector). Note also that

$$ {}^C\mathbf{p} = {}^C_B\mathbf{R}\,{}^B\mathbf{p} + {}^C\mathbf{p}_B \tag{3} $$

where ${}^C_B\mathbf{R}$ is the matrix which rotates vectors between frames $\{B\}$ and $\{C\}$, and ${}^C\mathbf{p}_B$ is the origin of $\{B\}$ with respect to $\{C\}$. We substitute (2) and (3) into (1), and rearrange the terms to obtain:

$$ \begin{aligned} {}^C\mathbf{p}' &= \left(\mathbf{I}_3 - 2\,{}^C\mathbf{n}\,{}^C\mathbf{n}^{\mathrm{T}}\right){}^C\mathbf{p} + 2d\,{}^C\mathbf{n} \\ &= \left(\mathbf{I}_3 - 2\,{}^C\mathbf{n}\,{}^C\mathbf{n}^{\mathrm{T}}\right)\left({}^C_B\mathbf{R}\,{}^B\mathbf{p} + {}^C\mathbf{p}_B\right) + 2d\,{}^C\mathbf{n} \end{aligned} \tag{4} $$

[1] Throughout this paper, ${}^X\mathbf{y}$ denotes a vector $\mathbf{y}$ expressed with respect to frame $\{X\}$, ${}^X_W\mathbf{R}$ is the rotation matrix rotating vectors from frame $\{W\}$ to $\{X\}$, and ${}^X\mathbf{p}_W$ is the origin of $\{W\}$, expressed with respect to $\{X\}$. $\mathbf{I}_n$ is the $n\times n$ identity matrix, and $\mathbf{0}_{m\times n}$ is the $m\times n$ matrix of zeros.

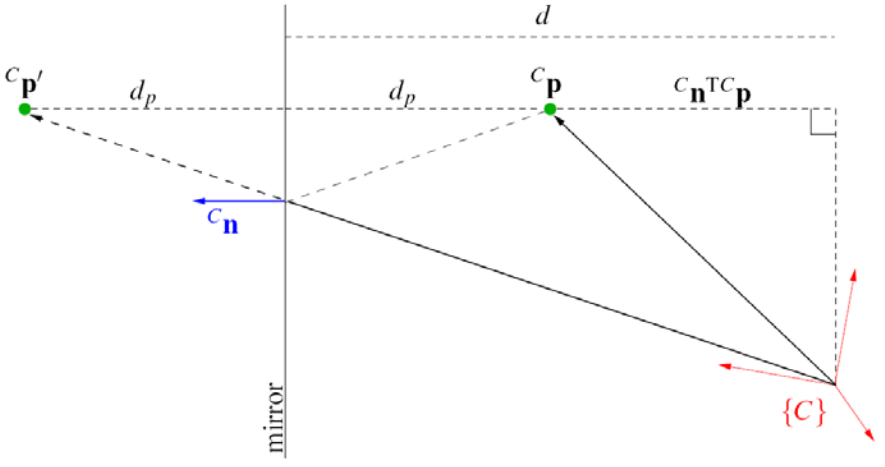


Fig. 1. Observation of the point ${}^C\mathbf{p}'$ which is the reflection of ${}^C\mathbf{p}$. In this figure, the mirror plane is perpendicular to the page. Only the reflected point is in the camera's field of view; the real point is not observed directly by the camera.

which can be written in homogeneous coordinates as:

$$\begin{bmatrix} {}^C\mathbf{p}' \\ 1 \end{bmatrix} = \begin{bmatrix} \left(\mathbf{I}_3 - 2{}^C\mathbf{n}{}^C\mathbf{n}^{\mathrm{T}}\right) & 2d\,{}^C\mathbf{n} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} {}^C_B\mathbf{R} & {}^C\mathbf{p}_B \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{p} \\ 1 \end{bmatrix}. \tag{5}$$

The reflection of $\mathbf{p}$ is observed by the camera, and this measurement is described by the perspective projection model:

$$\mathbf{z} = \frac{1}{p_3}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \eta = \mathbf{h}({}^C\mathbf{p}') + \eta, \quad {}^C\mathbf{p}' = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^{\mathrm{T}} \tag{6}$$

where η is the pixel noise, assumed to be a zero-mean, white Gaussian process with covariance matrix $\sigma_\eta^2\mathbf{I}_2$. Equations (4) and (6) define the measurement model, which expresses the observed image coordinates, $\mathbf{z}$, of the point as a function of the *known* position vector, ${}^B\mathbf{p}$, the *unknown* camera-to-base transformation, $\{{}^C_B\mathbf{R}, {}^C\mathbf{p}_B\}$, and the *unknown* configuration of the mirror with respect to the camera, $\{{}^C\mathbf{n},\ d\}$. Note that the transformation between the mirror and camera has 6 d.o.f., however, only 3 d.o.f. appear in the measurement equation. These are expressed by the vector $d\,{}^C\mathbf{n}$, which has 2 d.o.f. from the mirror normal, ${}^C\mathbf{n}$, and 1 d.o.f. from the camera-to-mirror distance, d. The remaining 3 d.o.f., which correspond to rotations about ${}^C\mathbf{n}$ and translations of the mirror-frame origin in the mirror plane, do not affect the measurements.

3.2 Three-Point Perspective Pose Estimation Problem

We now briefly review the three-point perspective pose estimation problem (P3P) and discuss how it relates to our problem. The goal of P3P is to determine the 6

d.o.f. transformation, $\{{}^C_B\mathbf{R}, {}^C\mathbf{p}_B\}$, between a camera frame, $\{C\}$, and a base frame, $\{B\}$, given the known coordinates of 3 non-collinear points, ${}^B\mathbf{p}_i$, $i = 1\ldots3$, in $\{B\}$, and their corresponding perspective projections, $\mathbf{z}_i$, in $\{C\}$, defined as:[2]

$$\mathbf{z}_i = \frac{1}{p_{3i}}\begin{bmatrix} p_{1i} \\ p_{2i} \end{bmatrix}, \quad {}^C\mathbf{p}'_i = \begin{bmatrix} p_{1i} & p_{2i} & p_{3i} \end{bmatrix}^{\mathrm{T}} \tag{7}$$

$$\begin{bmatrix} {}^C\mathbf{p}'_i \\ 1 \end{bmatrix} = \begin{bmatrix} {}^C_B\mathbf{R} & {}^C\mathbf{p}_B \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{p}_i \\ 1 \end{bmatrix}. \tag{8}$$

This problem has up to 4 pairs of solutions, where for each pair, there is one solution lying in front of the center of perspectivity and one behind it [4].

Equation (8) differs from (5) in that the former expresses a homogeneous transformation, while the latter describes a homogeneous transformation followed by a reflection. Effectively, our scenario is equivalent to a P3P in which an "imaginary" camera $\{\mathbf{C}^*\}$ with a left-handed reference frame lies behind the mirror and observes the true points (not the reflections). To bring (5) into a form similar to (8), we convert the imaginary camera to a right-handed system by pre-multiplying with a reflection across the y-axis (although any axis can be chosen):

$$\begin{aligned} \begin{bmatrix} {}^{\check{C}}\mathbf{p}' \\ 1 \end{bmatrix} &= \begin{bmatrix} \left(\mathbf{I}_3 - 2\mathbf{e}_2\mathbf{e}_2^{\mathrm{T}}\right) & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \left(\mathbf{I}_3 - 2\,{}^C\mathbf{n}{}^C\mathbf{n}^{\mathrm{T}}\right) & 2d\,{}^C\mathbf{n} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^C_B\mathbf{R} & {}^C\mathbf{p}_B \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{p} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} {}^{\check{C}}_B\mathbf{R} & {}^{\check{C}}\mathbf{p}_B \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{p} \\ 1 \end{bmatrix} \end{aligned} \tag{9}$$

where $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$, and $\{{}^{\check{C}}_B\mathbf{R}, {}^{\check{C}}\mathbf{p}_B\}$ is the transformation between $\{B\}$ and the right-handed frame $\{\check{C}\}$ of the "imaginary" camera behind the mirror. The origin of $\{\check{C}\}$ coincides with that of $\{C^*\}$, their x- and z-axes are common, and their y-axes lie in opposite directions. Note that this additional reflection can be implemented easily, by simply negating the sign of the y-coordinates of the image measurements.

Applying any P3P solution method to the modified problem in (9), we obtain up to 4 solutions, in general, for the unknown transformation $\{{}^{\check{C}}_B\mathbf{R}, {}^{\check{C}}\mathbf{p}_B\}$. We then reflect each of the solutions back, to obtain:

$$\begin{bmatrix} {}^C\mathbf{p}' \\ 1 \end{bmatrix} = \begin{bmatrix} \left(\mathbf{I}_3 - 2\mathbf{e}_2\mathbf{e}_2^{\mathrm{T}}\right) & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^{\check{C}}_B\mathbf{R} & {}^{\check{C}}\mathbf{p}_B \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} {}^B\mathbf{p} \\ 1 \end{bmatrix} \tag{10}$$

where the pair $\{\mathbf{A}, \mathbf{b}\}$, describes a reflection and a homogeneous transformation. Equating (5) and (10), we observe that:

$$\mathbf{A} = \left(\mathbf{I}_3 - 2\,{}^C\mathbf{n}{}^C\mathbf{n}^{\mathrm{T}}\right){}^C_B\mathbf{R} \tag{11}$$

$$\mathbf{b} = \left(\mathbf{I}_3 - 2\,{}^C\mathbf{n}{}^C\mathbf{n}^{\mathrm{T}}\right){}^C\mathbf{p}_B + 2d\,{}^C\mathbf{n}. \tag{12}$$

[2] The indices in this paper are: i for points, j for images, and k for solutions.

To summarize, in order to exploit the similarity of our problem to the P3P, we execute the following steps: First, the y-coordinates of the image measurements are negated. Then, the measurements are processed by a P3P algorithm to obtain up to 4 solutions $\{{}^{\check{C}}_{B}\mathbf{R}, {}^{\check{C}}\mathbf{p}_B\}$. Subsequently we employ (10), to obtain up to 4 solutions for $\mathbf{A}$ and $\mathbf{b}$. In the next section, we describe our approach for recovering the unknowns, $\{{}^{C}_{B}\mathbf{R}, {}^{C}\mathbf{p}_B, {}^{C}\mathbf{n}, d\}$, from $\mathbf{A}$ and $\mathbf{b}$ using (11) and (12).

3.3 Solution from 3 Points in 3 Images

We first examine the number of measurements required for a unique solution. When less than 3 points are observed, regardless of the number of images, there is not enough information to determine the transformation, since 3 non-collinear points are required to define the base frame of reference[3]. From 1 image with 3 points, there are not enough constraints to determine the unknowns [cf. (5) and (6)]. From 2 images with 3 points observed in each, the number of constraints equals the number of unknowns; however, in this case rotations of the 2 mirror planes about the axis of their intersection are unobservable, and thus 2 images are not sufficient [6].

From 3 images with 3 points in each, there are 18 scalar measurements [cf. (6)] and 15 unknowns; 6 from $\{{}^{C}_{B}\mathbf{R}, {}^{C}\mathbf{p}_B\}$, and 3 for each mirror configuration $\{\mathbf{n}_j, d_j\}$, $j = 1 \ldots 3$ [cf. (5)]. This is an overdetermined system, which is nonlinear in the unknown variables. In what follows, we show how to obtain a solution for this system.

Using P3P as an intermediate step, and momentarily ignoring multiple solutions, we obtain constraints of the form (11), (12) for each of the 3 images: $\{\mathbf{A}_j, \mathbf{b}_j\}$, $j = 1 \ldots 3$. For each pair of images, $j, j' \in \{1 \ldots 3\}$, we define the unit vector $\mathbf{m}_{jj'}$, as the perpendicular direction to $\mathbf{n}_j$ and $\mathbf{n}_{j'}$ (i.e., $\mathbf{n}_j^{\mathrm{T}}\mathbf{m}_{jj'} = \mathbf{n}_{j'}^{\mathrm{T}}\mathbf{m}_{jj'} = 0$). Alternatively stated, $\mathbf{m}_{jj'} = \alpha\mathbf{n}_j \times \mathbf{n}_{j'}$, where α is a scaling constant to ensure unit length. Using (11), we obtain:

$$\mathbf{A}_j\mathbf{A}_{j'}^{\mathrm{T}}\mathbf{m}_{jj'} = \left(\mathbf{I}_3 - 2\,{}^{C}\mathbf{n}_j\,{}^{C}\mathbf{n}_j^{\mathrm{T}}\right)\left(\mathbf{I}_3 - 2\,{}^{C}\mathbf{n}_{j'}\,{}^{C}\mathbf{n}_{j'}^{\mathrm{T}}\right)\mathbf{m}_{jj'} = \mathbf{m}_{jj'}. \tag{13}$$

Thus, by computing the eigenvector corresponding to the unit eigenvalue of $\mathbf{A}_j\mathbf{A}_{j'}^{\mathrm{T}}$, we determine $\mathbf{m}_{jj'}$ up to sign (it can be shown that $\mathbf{A}_j\mathbf{A}_{j'}^{\mathrm{T}}$ is a special orthogonal matrix with 2 complex conjugate eigenvalues, and 1 eigenvalue equal to 1). Employing the properties of the cross product, we obtain:[4]

$$\mathbf{n}_1 = \frac{\mathbf{m}_{13} \times \mathbf{m}_{12}}{||\mathbf{m}_{13} \times \mathbf{m}_{12}||}, \quad \mathbf{n}_2 = \frac{\mathbf{m}_{21} \times \mathbf{m}_{23}}{||\mathbf{m}_{21} \times \mathbf{m}_{23}||}, \quad \mathbf{n}_3 = \frac{\mathbf{m}_{13} \times \mathbf{m}_{23}}{||\mathbf{m}_{13} \times \mathbf{m}_{23}||}. \tag{14}$$

Once we have determined the unit vectors corresponding to the 3 mirror planes, the rotation matrix, ${}^{C}_{B}\mathbf{R}$, can be computed independently from 3 sets of equations:

[3] In the case of 2 points, or 3 or more collinear points, rotations about the line that the points lie on are not observable.

[4] For the remainder of the paper, we drop the superscript 'C' from $\mathbf{n}_j,\ j = 1 \ldots 3$.

$$ {}^{C}_{B}\mathbf{R}_j = \left(\mathbf{I} - 2\mathbf{n}_j\mathbf{n}_j^{\mathrm{T}}\right)\mathbf{A}_j, \quad j = 1\ldots 3. \tag{15} $$

In order to utilize all the available information, and to reduce numerical errors, we seek to compute an "average" ${}^{C}_{B}\mathbf{R}$ from these 3 sets of equations. However, employing the arithmetic mean is inappropriate since the property of orthonormality is not maintained. We address this issue with the procedure described in Appendix 1.

Once the rotation, ${}^{C}_{B}\mathbf{R}$, and the mirror normal vectors, $\mathbf{n}_j$, $j = 1\ldots 3$, are determined, the remaining unknowns $\{{}^{C}\mathbf{p}_B,\ d_1,\ d_2,\ d_3\}$ appear linearly in the constraint equations [cf. (12)]

$$ \begin{bmatrix} \left(\mathbf{I} - 2\mathbf{n}_1\mathbf{n}_1^{\mathrm{T}}\right) & 2\mathbf{n}_1 & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} \\ \left(\mathbf{I} - 2\mathbf{n}_2\mathbf{n}_2^{\mathrm{T}}\right) & \mathbf{0}_{3\times 1} & 2\mathbf{n}_2 & \mathbf{0}_{3\times 1} \\ \left(\mathbf{I} - 2\mathbf{n}_3\mathbf{n}_3^{\mathrm{T}}\right) & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} & 2\mathbf{n}_3 \end{bmatrix} \begin{bmatrix} {}^{C}\mathbf{p}_B \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix} \Leftrightarrow \mathbf{D}\mathbf{x} = \mathbf{c} \tag{16} $$

where $\mathbf{D}$ is a 9×6 known matrix, $\mathbf{c}$ is a 9×1 known vector, and $\mathbf{x}$ is the 6×1 vector of unknowns. The least-squares solution for $\mathbf{x}$ in this linear system is $\mathbf{x} = \mathbf{D}^{\dagger}\mathbf{c}$, where $\mathbf{D}^{\dagger}$ denotes the Moore-Penrose generalized inverse of $\mathbf{D}$. From (14), (15), (23), and (16) the mirror configurations, as well as the camera-to-base transformation are computed.

Up to this point, we assumed that the P3P solution was unique, however, there may be up to 4 solutions per image. Recall that 3 images are required to compute the camera-to-base transformation analytically, hence, there are up to 64 solutions for $\{{}^{C}_{B}\mathbf{R}, {}^{C}\mathbf{p}_B, d_1, d_2, d_3, \mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3\}$, arising from the $4\times 4\times 4$ possible combinations of P3P solutions. When the measurements are noiseless, we have observed in simulations that only one of these solutions yields a zero-reprojection error (i.e., satisfies all the constraints exactly). This is because the problem at hand is over-constrained (18 constraints for 15 unknowns), and we expect to have a unique solution. In the presence of pixel noise, none of the solutions will satisfy the measurements exactly, thus, we choose the one with the minimum reprojection error.

Moreover, when $N_c > 3$ images are available, there are $N_s = \binom{N_c}{3}$ analytically computed transformations. However, some of these may be inaccurate as a result of degenerate sets of measurements (e.g., when 3 images are all taken from similar viewing angles). In order to identify the correct solution, we employ spectral clustering to determine the largest set of similar solutions [16]. Specifically, we adopt the unit-quaternion representation of rotation [19], ${}^{C}\bar{q}_B$, which corresponds to ${}^{C}_{B}\mathbf{R}$, and denote each solution as $\{{}^{C}\bar{q}_B^{(k)}, {}^{C}\mathbf{p}_B^{(k)}\}$ for $k = 1\ldots N_s$. To perform spectral clustering, we define an affinity matrix, $\mathbf{L}$, in which each element is the Mahalanobis distance between a pair of solutions, indexed by k and k':

$$ \mathbf{L}_{kk'} = \begin{bmatrix} \delta\boldsymbol{\theta}_{kk'}^{\mathrm{T}} & \delta\mathbf{p}_{kk'}^{\mathrm{T}} \end{bmatrix} \left[\left(\mathbf{H}_k^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{H}_k\right)^{-1} + \left(\mathbf{H}_{k'}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{H}_{k'}\right)^{-1} \right]^{-1} \begin{bmatrix} \delta\boldsymbol{\theta}_{kk'} \\ \delta\mathbf{p}_{kk'} \end{bmatrix} \tag{17} $$

where $\delta\boldsymbol{\theta}_{kk'}$ is the quaternion error-angle vector between ${}^{C}\bar{q}_B^{(k)}$ and ${}^{C}\bar{q}_B^{(k')}$ [19], and $\delta\mathbf{p}_{kk'} = {}^{C}\mathbf{p}_B^{(k)} - {}^{C}\mathbf{p}_B^{(k')}$ is the difference between the translation vectors. The matrices

$\mathbf{H}_k$ and $\mathbf{H}_{k'}$ are the measurement Jacobians with respect to the transformation [6], and $\mathbf{Q} = \sigma_\eta^2 \mathbf{I}_2$ is the covariance of the pixel noise. We compute the transformation, $\{{}^C_B\mathbf{R}, {}^C\mathbf{p}_B\}$, from the largest spectral cluster. The rotation, ${}^C_B\mathbf{R}$, is determined from (23) using all the quaternions in the cluster (cf. Appendix 1), and the translation, ${}^C\mathbf{p}_B$, is computed as the arithmetic mean of the translations in the cluster.

3.4 Refining the Solution

Due to the presence of pixel noise, and the fact that noise was not accounted for in the analytical solution, the result of the procedure presented in Sections 3.2-3.3 may be coarse (cf. Section 4). Hence, we employ an MLE to refine our analytically computed estimate. We now present an overview of the MLE for determining the unknown transformation between the camera and base frame [7]. Let the vector of all unknown parameters be denoted by $\mathbf{x}$. This vector comprises the unknown transformation, as well as the parameters $\{{}^C\mathbf{n}_j, d_j\}$, $j = 1 \ldots N_c$, that describe each mirror configuration:

$$\mathbf{x} = \left[{}^C\mathbf{p}_B^{\mathrm{T}} \; {}^C\bar{q}_B^{\mathrm{T}} \; {}^C\mathbf{n}_1^{\mathrm{T}} \; d_1 \; \ldots \; {}^C\mathbf{n}_{N_c}^{\mathrm{T}} \; d_{N_c} \right]^{\mathrm{T}}. \tag{18}$$

Assuming Gaussian pixel noise, the likelihood of the measurements is given by:

$$\begin{aligned} L(\mathscr{Z};\mathbf{x}) &= \prod_{i=1}^{N_p}\prod_{j=1}^{N_c} p(\mathbf{z}_{ij};\mathbf{x}) = \prod_{i=1}^{N_p}\prod_{j=1}^{N_c} \frac{1}{2\pi\sigma_\eta^2} \exp\left[-\frac{\left(\mathbf{z}_{ij} - \mathbf{h}\left({}^{C_j}\mathbf{p}_i'\right)\right)^{\mathrm{T}} \left(\mathbf{z}_{ij} - \mathbf{h}\left({}^{C_j}\mathbf{p}_i'\right)\right)}{2\sigma_\eta^2} \right] \\ &= \prod_{i=1}^{N_p}\prod_{j=1}^{N_c} \frac{1}{2\pi\sigma_\eta^2} \exp\left[-\frac{\left(\mathbf{z}_{ij} - \mathbf{h}_{ij}(\mathbf{x})\right)^{\mathrm{T}} \left(\mathbf{z}_{ij} - \mathbf{h}_{ij}(\mathbf{x})\right)}{2\sigma_\eta^2} \right] \end{aligned}$$

where the dependence on $\mathbf{x}$ is explicitly shown [cf. (5), (6)], and N_p is the total number of points observed in each of the N_c images. Maximizing the likelihood is equivalent to minimizing its negative logarithm, or minimizing the cost function:

$$c(\mathbf{x}) = \sum_{i=1}^{N_p}\sum_{j=1}^{N_c} \left(\mathbf{z}_{ij} - \mathbf{h}_{ij}(\mathbf{x})\right)^{\mathrm{T}} \left(\mathbf{z}_{ij} - \mathbf{h}_{ij}(\mathbf{x})\right). \tag{19}$$

We solve this nonlinear least-squares problem with Gauss-Newton iterative minimization to estimate $\mathbf{x}$. At each iteration, indexed by ℓ, the estimate is changed by

$$\delta\mathbf{x}^{(\ell)} = \left(\sum_{i,j} \mathbf{J}_{ij}^{(\ell)\mathrm{T}} \mathbf{J}_{ij}^{(\ell)} \right)^{-1} \left(\sum_{i,j} \mathbf{J}_{ij}^{(\ell)\mathrm{T}} \left(\mathbf{z}_{ij} - \mathbf{h}_{ij}(\mathbf{x}^{(\ell)}) \right) \right)$$

where $\mathbf{J}_{ij}^{(\ell)}$ is the Jacobian of $\mathbf{h}_{ij}$ with respect to $\mathbf{x}$, evaluated at the current iterate, $\mathbf{x}^{(\ell)}$. The analytically computed solution from Sections 3.2-3.3 is utilized as the

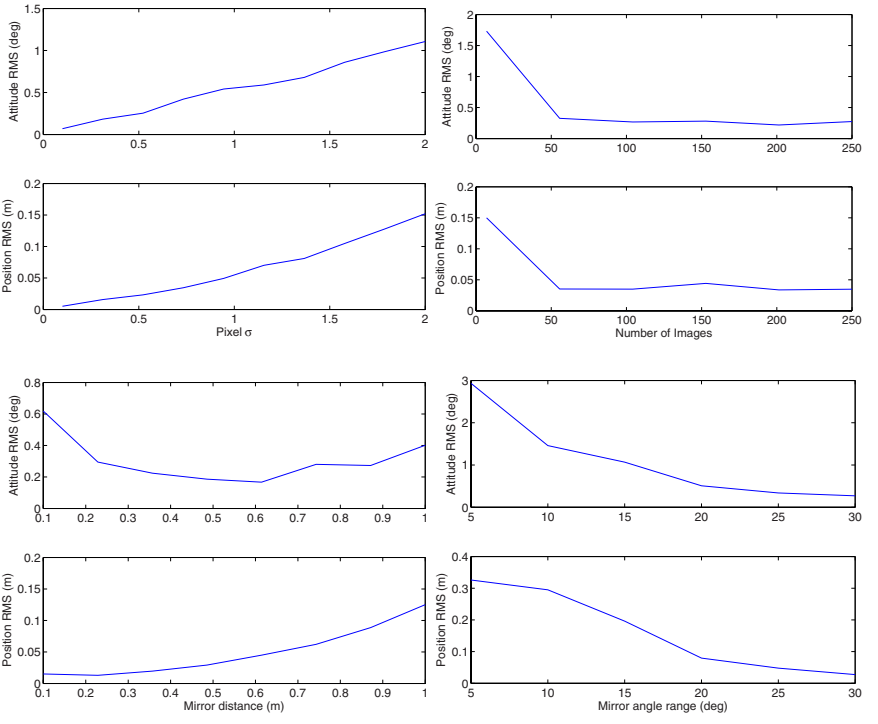


Fig. 2. Average RMS error over 10 trials for attitude and position plotted versus: (a) pixel noise, (b) number of images, (c) mirror distance, and (d) range of mirror rotation.

initial iterate, $\mathbf{x}^{(0)}$. Since the MLE is not the main contribution of this work, we limit our discussion here, but refer the reader to [7] for more details.

4 Simulations

In this section, we study the accuracy of the analytically computed camera-to-base transformation (cf. Sections 3.2-3.3). In particular, we investigate how the accuracy is affected by the following parameters: (i) pixel noise, (ii) number of images, (iii) distance from camera to mirror, and (iv) range of the mirror's angular motion. We consider a "standard" case, in which 3 points placed at the corners of a right triangle with sides measuring $20 \times 20 \times 20\sqrt{2}$ cm are observed in 200 images, while a mirror placed at a distance of 0.5 m is rotated by 30^{o} in two directions. We vary each of the aforementioned parameters individually to examine its effect on the solution accuracy. In Fig. 2, we plot the average RMS error for the position and attitude, computed over 10 trials. Some key observations are:

- Increasing the camera's pixel noise decreases the accuracy of the computed solution. When the camera measurements become substantially noisy, e.g., $\sigma = 2$ pixels, the average RMS error is 1^{o} in attitude and 15 cm in position.
- Increasing the number of images results in higher accuracy. However, the improvement follows the "law of diminishing returns," i.e., when a large number of images is already available, the impact of recording more observations is smaller.
- Changing the distance from the mirror to the camera has a significant effect on the position accuracy. When the mirror is at a distance of 1 m, the average RMS error for position is approximately 13 cm. The magnitude of this error suggests that the mirror distance should be kept small. Additionally, it highlights the need to refine our analytically computed transformation with an MLE. As we show in [7], the accuracy of the MLE is approximately 5 times better in attitude, and 10 times better in position compared to the analytical solution.
- Increasing the range of the mirror's angular motion results in improved accuracy. The effect is significant and every effort should be made to move the mirror in the widest range of motion allowed by the camera's field of view.

As a final remark, we note that using the analytical solution as an initial guess for the MLE enables the latter to converge to the correct minimum 100% of the time. On average, fewer iterations were required (approx. 7) when compared to using a naïve initial guess (approx. 18). This shows that a precise analytical solution improves the speed and robustness of the overall estimation process.

5 Experiments

The method described in the preceding sections was employed for computing the transformation between a camera and a base frame attached on the robot-body. For this purpose, 3 fiducial points were placed in known positions on the robot as shown in Fig. 3b. The origin of $\{B\}$ coincides with the top-left fiducial point; both $\{B\}$ and $\{C\}$ are right-handed systems with the axes of $\{B\}$ approximately aligned with those of $\{C\}$. These points were tracked using the KLT algorithm [18] in 1000 images, recorded by a Firewire camera with resolution of 1024×768 pixels.

A planar mirror was maneuvered in different spatial configurations (rotating about two axes), and in distances varying between 30 and 50 cm from the camera, in order to generate a wide range of views. All the measurements were processed to compute the transformation analytically: ${}^{C}\mathbf{p}_{B} = \begin{bmatrix} -14.13 & -10.25 & -13.89 \end{bmatrix}^{\text{T}}$ cm, and ${}^{C}\bar{q}_{B} = \begin{bmatrix} -0.0401 & -0.0017 & -0.0145 & 0.9991 \end{bmatrix}^{\text{T}}$. This initial solution was refined using the MLE described in Section 3.4, to obtain a better estimate for the transformation between the two frames of interest. The Gauss-Newton iterative minimization converged after 8 iterations, to the following solution for the transformation: ${}^{C}\mathbf{p}_{B} = \begin{bmatrix} -14.80 & -15.96 & -14.95 \end{bmatrix}^{\text{T}}$ cm, and ${}^{C}\bar{q}_{B} = \begin{bmatrix} 0.0045 & 0.0774 & 0.0389 & 0.9962 \end{bmatrix}^{\text{T}}$. The corresponding 3σ uncertainty bounds are $\begin{bmatrix} 1.1 & 1.6 & 5.0 \end{bmatrix}$ mm for the position, and $\begin{bmatrix} 0.2419 & 0.2313 & 0.0665 \end{bmatrix}$ degrees for the orientation estimates. We point out that the estimates agree with our best guess from manual measurement. We believe that the

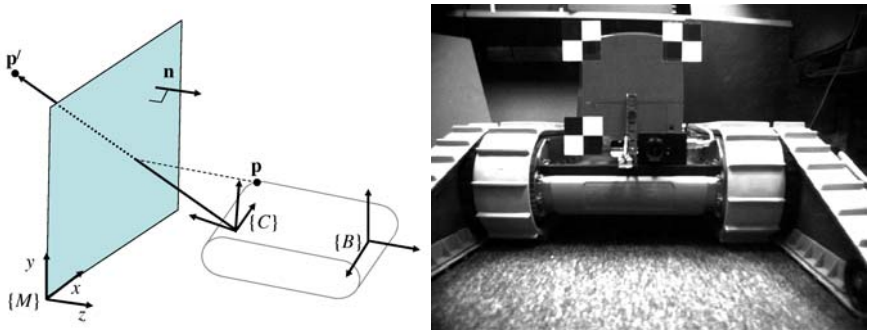


Fig. 3. (a) Observation of a point on the robot reflected in the mirror, and (b) an image with 3 fiducial points, captured during experimentation.

attained accuracy (given by the 3σ bounds from the MLE) is sufficiently high for most practical applications.

6 Conclusions and Future Work

In this paper, we propose a method for computing the 6 d.o.f. transformation between a camera and a base frame of reference. A mirror is maneuvered in front of the camera, to provide observations of known points from different viewing angles and distances. These measurements are utilized to analytically compute the camera-to-base transformation, and the solution is refined using a maximum-likelihood estimator, which produces estimates for the camera-to-base transformation, as well as for the mirror configuration in each image. The approach was validated both in simulation and experimentally. One of the key advantages of the proposed method is its ease of use; it only requires a mirror, and it provides a solution with as little as 3 points viewed in 3 images. When more information is available, it can be incorporated to produce a more accurate estimate of the transformation.

In our future work, we will investigate the feasibility of mirror-based robot-body 3D reconstruction which we briefly discuss in Appendix 2. Furthermore, we plan to extend this method to the case where the coordinates of the points in the base frame are not known *a priori*, but are estimated along with the camera-to-base transformation and the mirror configurations.

Acknowledgements. This work was supported by the University of Minnesota (DTC), and the National Science Foundation (EIA-0324864, IIS-0643680, IIS-0811946). Anastasios Mourikis was partially supported by the UMN Doctoral Dissertation Fellowship, and by the University of California, Riverside. Joel Hesch was supported by the NIH Neuro-physical-computational Sciences Fellowship.

Appendix 1

In this section, we describe the procedure employed for computing an "average rotation," given N_q rotation estimates $\bar{q}_j,\ j = 1 \ldots N_q$. We adopt the quaternion notation from [19] and denote the quaternion of rotation arising from the jth set of equations as $\bar{q}_j$, which corresponds to ${}^{C}_{B}\mathbf{R}_j$ [cf. (15)]. Assuming that $\bar{q}$ is the optimal estimate, and employing the small error-angle approximation, we write the following expression for the error in each $\bar{q}_j$:

$$\bar{q}_j \otimes \bar{q}^{-1} \simeq \begin{bmatrix} \hat{\mathbf{k}}_j \frac{\delta\theta_j}{2} \\ 1 \end{bmatrix}, \quad j = 1 \ldots N_q \tag{20}$$

where $\otimes$ denotes quaternion multiplication, $\hat{\mathbf{k}}_j$ is the unit-vector axis of rotation, and $\delta\theta_j$ is the error angle between the two quaternions. Rewriting this last expression as a matrix-vector multiplication [19], yields

$$\mathscr{L}(\bar{q}_j)\,\bar{q}^{-1} = \begin{bmatrix} \hat{\mathbf{k}}_j \frac{\delta\theta_j}{2} \\ 1 \end{bmatrix}, \quad j = 1 \ldots N_q \tag{21}$$

where $\mathscr{L}(\bar{q}_j)$, is the left-side quaternion multiplication matrix parameterized by $\bar{q}_j$. Projecting this relation, to keep only the error components, we obtain:

$$\mathbf{P}\mathscr{L}(\bar{q}_j)\,\bar{q}^{-1} = \hat{\mathbf{k}}_j \frac{\delta\theta_j}{2}, \quad j = 1 \ldots N_q \tag{22}$$

where $\mathbf{P} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 1} \end{bmatrix}$. Stacking these relations, we have

$$\begin{bmatrix} \mathbf{P}\mathscr{L}(\bar{q}_1) \\ \vdots \\ \mathbf{P}\mathscr{L}(\bar{q}_{N_q}) \end{bmatrix} \bar{q}^{-1} = \frac{1}{2} \begin{bmatrix} \hat{\mathbf{k}}_1 \frac{\delta\theta_1}{2} \\ \vdots \\ \hat{\mathbf{k}}_{N_q} \frac{\delta\theta_{N_q}}{2} \end{bmatrix}. \tag{23}$$

Our goal is to find the $\bar{q}^{-1}$ that minimizes the norm of the right-hand side. This occurs when $\bar{q}^{-1} = \mathbf{v}(\sigma_{min})$, i.e., we select $\bar{q}^{-1}$ to be the right singular vector corresponding to the minimum singular value of the $3N_q \times 4$ matrix multiplying $\bar{q}^{-1}$ in (23). After finding $\bar{q}^{-1}$ by SVD, we compute the optimal estimate for the rotational matrix ${}^{C}_{B}\mathbf{R} = \mathbf{R}(\bar{q})$, which is the rotational matrix parameterized by $\bar{q}$.

Appendix 2

We turn our attention to mirror-based robot-body 3D reconstruction using mirror reflections. We assume that in addition to the 3 points which are known in the robot-body frame, $\{B\}$, we observe another point, $\mathbf{p}_u$, which is *unknown* in $\{B\}$. From one image, we have [cf. (4)]:

$$\beta\, {}^{C}\mathbf{p}'_{u_0} = \left(\mathbf{I}_3 - 2\,{}^{C}\mathbf{n}\,{}^{C}\mathbf{n}^{\mathrm{T}}\right) {}^{C}_{B}\mathbf{R}\,{}^{B}\mathbf{p}_u + \left(\mathbf{I}_3 - 2\,{}^{C}\mathbf{n}\,{}^{C}\mathbf{n}^{\mathrm{T}}\right) {}^{C}\mathbf{p}_B + 2d\,{}^{C}\mathbf{n} \tag{24}$$

where β is an unknown scale factor and ${}^{C}\mathbf{p}'_{u_0}$ is the unit vector along the direction of ${}^{C}\mathbf{p}'_{u}$. Pre-multiplying both sides by the reflection matrix yields

$$\beta\, \left(\mathbf{I}_3 - 2\,{}^{C}\mathbf{n}\,{}^{C}\mathbf{n}^{\mathrm{T}}\right) {}^{C}\mathbf{p}'_{u_0} = {}^{C}_{B}\mathbf{R}\,{}^{B}\mathbf{p}_u + {}^{C}\mathbf{p}_B - 2d\,{}^{C}\mathbf{n}. \tag{25}$$

We assume that the transformation from $\{B\}$ to $\{C\}$, as well as the mirror configuration have been determined using the method outlined in this paper. Hence, the quantities $\{{}^{C}_{B}\mathbf{R}, {}^{C}\mathbf{p}_B, {}^{C}\mathbf{n},\ d\}$ are known and ${}^{C}\mathbf{p}'_{u_0}$ is measured, while the quantities $\{\beta, {}^{B}\mathbf{p}_u\}$ are unknown. From a single image, there are 3 constraints [cf. (25)] and 4 unknowns; hence, we can constrain ${}^{B}\mathbf{p}_u$ to lie on a line parameterized by β. If the point is observed in 2 consecutive images, then we will have 6 constraints and 5 unknowns, 3 corresponding to the unknown point's coordinates and 2 to the unknown scale factors. In this case, we expect that ${}^{B}\mathbf{p}_u$ can be determined uniquely.

This problem is analogous to triangulation of a point from two image views ([5], ch. 12). It is solvable when the origin of the camera frame is different for the two views. This corresponds to the quantity $d\,{}^{C}\mathbf{n}$ changing. Thus, it suffices to either change the distance to the mirror, or the mirror's orientation with respect to the camera. We expect that the location of every unknown point on the robot-body, which is visible in the mirror reflections, can be determined in the body frame of reference, given that it can be reliably tracked in at least 2 images taken from different views.

References

1. Brun, X., Goulette, F.: Modeling and calibration of coupled fish-eye CCD camera and laser range scanner for outdoor environment reconstruction. In: Proc. of the Int. Conf. on 3D Digital Imaging and Modeling, Montréal, Canada, August 21-23, pp. 320–327 (2007)
2. Daniilidis, K.: Hand-eye calibration using dual quaternions. Int. Journal of Robotics Research 18(3), 286–298 (1999)
3. Gluckman, J., Nayar, S.K.: Planar catadioptric stereo: Geometry and calibration. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Ft. Collins, CO, June 23-25, pp. 22–28 (1999)
4. Haralick, R.M., Lee, C.-N., Ottenberg, K., Nölle, M.: Review and analysis of solutions of the three point perspective pose estimation problem. Int. Journal of Computer Vision 13(3), 331–356 (1994)
5. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
6. Hesch, J.A., Mourikis, A.I., Roumeliotis, S.I.: Camera to robot-body calibration using planar mirror reflections. Technical Report 2008-001, University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab (July 2008)
7. Hesch, J.A., Mourikis, A.I., Roumeliotis, S.I.: Determining the camera to robot-body transformation from planar mirror reflections. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Nice, France, September 22-26, pp. 3865–3871 (2008)

8. Inaba, M., Hara, T., Inoue, H.: A stereo viewer based on a single camera with view-control mechanisms. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Yokohama, Japan, July 26-30, pp. 1857–1865 (1993)
9. Jang, G., Kim, S., Kweon, I.: Single camera catadioptic stereo system. In: Proc. of the Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, Beijing, China, October 21 (2005)
10. Jang, K.H., Lee, D.H., Jung, S.K.: A moving planar mirror based approach for cultural reconstruction. Computer Animation and Virtual Worlds 15(3-4), 415–423 (2004)
11. Kanbara, M., Ukita, N., Kidode, M., Yokoya, N.: 3D scene reconstruction from reflection images in a spherical mirror. In: Proc. of the Int. Conf. on Pattern Recognition, Hong Kong, China, August 20-24, pp. 874–897 (2006)
12. Kumar, R.K., Ilie, A., Frahm, J.-M., Pollefeys, M.: Simple calibration of non-overlapping cameras with a mirror. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Anchorage, AK, June 24-26 (2008)
13. Martinelli, A., Siegwart, R.: Observability properties and optimal trajectories for on-line odometry self-calibration. In: Proc. of the IEEE Conf. on Decision and Control, San Diego, CA, December 13-15, pp. 3065–3070 (2006)
14. Mirzaei, F.M., Roumeliotis, S.I.: A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. IEEE Trans. on Robotics 24(5), 1143–1156 (2008)
15. Nayar, S.K.: Sphereo: Determining depth using two specular spheres and a single camera. In: Proc. of the SPIE Conf. on Optics, Illumination, and Image Sensing for Machine Vision, November 1988, vol. 1005, pp. 245–254 (1988)
16. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems, British Columbia, Canada, December 3-8, vol. 2, pp. 849–856 (2002)
17. Ramsgaard, B.K., Balslev, I., Arnspang, J.: Mirror-based trinocular systems in robot-vision. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Barcelona, Spain, September 3-7, pp. 499–502 (2000)
18. Shi, J., Tomasi, C.: Good features to track. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Washington, DC, June 27-July 2, pp. 593–600 (1994)
19. Trawny, N., Roumeliotis, S.I.: Indirect Kalman filter for 3D attitude estimation. Technical Report 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab (March 2005)
20. Tsai, R.Y., Lenz, R.K.: A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. IEEE Trans. on Robotics and Automation 5(3), 345–358 (1989)
21. Wasielewski, S., Strauss, O.: Calibration of a multi-sensor system laser rangefinder/camera. In: Proc. of the Intelligent Vehicles Symposium, Detroit, MI, September 25-26, pp. 472–477 (1995)
22. Würz-Wessel, A., Stein, F.K.: Calibration of a free-form surface mirror in a stereo vision system. In: Proc. of the IEEE Intelligent Vehicle Symposium, Versailles, France, June 17-21, vol. 2, pp. 471–476 (2002)

On the Analysis of the Depth Error on the Road Plane for Monocular Vision-Based Robot Navigation*

Dezhen Song, Hyunnam Lee, and Jingang Yi

Abstract. A mobile robot equipped with a single camera can take images at different locations to obtain the 3D information of the environment for navigation. The depth information perceived by the robot is critical for obstacle avoidance. Given a calibrated camera, the accuracy of depth computation largely depends on locations where images have been taken. For any given image pair, the depth error in regions close to the camera baseline can be excessively large or even infinite due to the degeneracy introduced by the triangulation in depth computation. Unfortunately, this region often overlaps with the robot's moving direction, which could lead to collisions. To deal with the issue, we analyze depth computation and propose a predictive depth error model as a function of motion parameters. We name the region where the depth error is above a given threshold as an untrusted area. Note that the robot needs to know how its motion affect depth error distribution beforehand, we propose a closed-form model predicting how the untrusted area is distributed on the road plane for given robot/camera positions. The analytical results have been successfully verified in the experiments using a mobile robot.

1 Introduction

Vision-based navigation is preferable because cameras can be very small, passive, and energy-efficient. Using a single camera to assist a mobile robot is the most

Dezhen Song
CSE Dept., Texas A&M University
e-mail: dzsong@cse.tamu.edu

Hyunnam Lee
Samsung Techwin Robot Business
e-mail: hyunnamlee@gmail.com

Jingang Yi
MAE Dept., Rutgers University
e-mail: jgyi@jove.rutgers.edu

* This work was supported in part by the National Science Foundation under IIS-0534848 and IIS-0643298, and in part by Microsoft Corporation.

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 301–315.
springerlink.com

simplistic configuration and is often adopted in small robots. However, images from cameras contain rich information of the environment, and understanding the imaging data is nontrivial. Extracting geometry information from images is critical for obstacle avoidance. Stereo vision approaches are often employed.

For the monocular system, the stereo information can be constructed using structure from motion (SFM) approach[1]. This method constructs depth information using images taken at different perspectives. Since the robot motion changes camera perspectives, the baseline distance is not limited by the width of the robot and it is desirable for small robots. However, the SFM approach has its own limitation. The depth of obstacles located at the baseline cannot be obtained because the camera centers and obstacle locations are collinear. Unfortunately, if the robot moves along a straight line, its forward direction is always the baseline direction.

Understanding depth error distribution on the road plane is critical for applications such as robot navigation. We model how depth error is distributed on the road plane and partition the road plane using a given error threshold. The predictive closed-form model is a function of robot motion settings and can be used to predict how the region beyond the given error threshold changes on the road plane. Hence the model has the potential to benefit a variety of applications including 1) guiding the robot for mixed initiative motion planning for better sensing and navigation, 2) guiding the selection of visual landmarks for vision-based simultaneous localization and mapping (SLAM), and 3) improving the visual tracking performance for mobile robots.

The proposed predictive depth error distribution model has been tested in physical experiments. The experiments use a mobile robot and artificial obstacles to validate the predictive depth error model. The experimental results have confirmed our analysis.

2 Related Work

Our research is related to monocular vision systems for robots, structure from motion (SFM) [1], and active vision[2, 3, 4].

Due to its simple configuration, a monocular vision system is widely used in mobile robots with space and power constraints. The research work in this category can be classified into two types including SLAM and vision-based navigation. SLAM [5, 10, 7, 8] focuses on the mapping and localization aspects and is often used in structured indoor environments where there are no global positioning system (GPS) signals to assist robots in navigation. SLAM focuses on identifying and managing landmark/feature points from the scene for map building and localization. Obstacle avoidance is not the concern of SLAM.

Our work focuses on monocular vision-based navigation for obstacle detection and avoidance. Due to the inherent difficulty in understanding the environment using monocular vision, many researchers focus on applying machine learning techniques to assist navigation [9, 6, 11, 12]. However, those methods are appearance-based

and only utilize color and texture information. Lack of geometry information limits their ability in obstacle detection.

Our work is a geometry-based approach that uses SFM to obtain the information of the environment. SFM can simultaneously estimate both the 3D scene and camera motion information[1]. Since the camera motion information is usually available from on-board sensors such as an inertial measurement unit (IMU) or wheel encoders, the dimensionality of the SFM problem can be reduced to the only estimation of the 3D scene, namely the triangulation computation. The depth error is determined by the image correspondence error and the camera perspectives. To obtain the 3D information, it is necessary to find the corresponding points between the overlapping images. However, due to the fact that images are discrete representations of the environment and the inherent difficulty in image matching, it is unavoidable that matching errors are introduced into the corresponding points[13, 14]. There are many newly developed techniques that can be used to reduce correspondence errors. Such techniques include low-rank approximations [15, 16, 17], power factorization [18], closure constraints [19], and covariance-weighted data [20]. In addition, new features, such as planar parallax [21, 22, 23, 24] and the probability of correspondence points [25], can be used instead of correspondence points to reduce the correspondence error.

Our work accepts the fact that image correspondence cannot be eliminated completely. We instead study how the depth error is affected by the image correspondence error. Although the variance of the image correspondence error are the same across the image plane[13, 14], the variance of depth error is not uniformly distributed across the image coverage[26]. Therefore, robot navigation and camera motion planning should take the depth error distribution information into account. This observation inspires our development.

3 Problem Description

3.1 Coordinate Systems

Our algorithm runs every τ_0 time. In each period, the robot has a trajectory $T(\tau)$, $\tau \in [0, \tau_0]$. The period length τ_0 is a preset parameter depending on the speed of the robot and the computation time necessary for stereo reconstruction. The most common approach to assist robot navigation is to take a frame $\underline{F}$ at $\tau = 0$ and another frame $\overline{F}$ at $\tau = \tau_0$ for the two-view stereo reconstruction. As a convention, we use underline and overline with variables to indicate their correspondence to $\underline{F}$ and $\overline{F}$, respectively. To clarify the problem, we introduce the following right hand coordinate systems as illustrated in Fig. 1.

- World coordinate system (WCS): A fixed 3D Cartesian coordinate system. Its y-axis is the vertical axis, and its x-z plane is the road plane. Trajectory $T(\tau)$ is located in the x-z plane with $T(\tau_0)$ located at the origin of the WCS. Hence, $T(\tau) = [x_w(\tau), z_w(\tau)]^T, 0 \leq \tau \leq \tau_0$ as illustrated in Fig. 1.

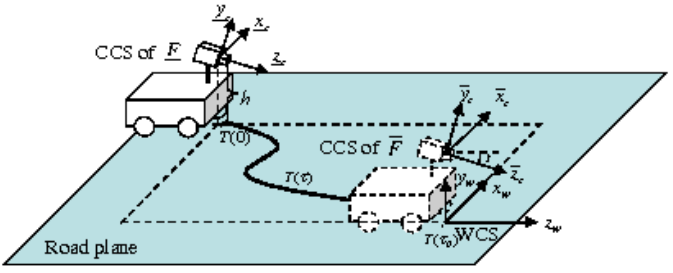


Fig. 1. Definition of coordinate systems and their relationship. The WCS is a fixed coordinate system while a CCS is attached to the moving camera.

- Camera coordinate system (CCS): A 3D Cartesian coordinate system that is attached to a camera mounted on a robot with its origin at the camera optical center. Its z-axis coincides with the optical axis and points to the forward direction of the robot. Its x-axis and y-axis are parallel to the horizontal and vertical directions of the CCD sensor plane, respectively.
- Image coordinate system (ICS): A 2D image coordinate system with the u-axis and v-axis parallel with the horizontal and vertical directions of an image, respectively. Its origin is located at its principal point. Coordinates u and v are discretized pixel readings. When we mention frames such as F, $\underline{F}$ and $\overline{F}$, they are defined in the ICS.

Frames such as $\underline{F}$ and $\overline{F}$ have their corresponding CCSs and ICSs. We use the notation $CCS(F)$ to represent the corresponding CCS for frame F. As illustrated in Fig. 1, the origin of $CCS(\overline{F})$ projects to $T(\tau_0)$ on the road plane, which is the origin of the WCS. The vertical distance between the origins of the $CCS(\overline{F})$ and the WCS is the camera height h. The origin of $CCS(\underline{F})$ projects to $T(0)$ on the road plane.

3.2 Assumptions

- We assume that obstacles in the environment are either static or slow-moving. Therefore, the SFM algorithm can be applied to compute the depth information.
- We assume both intrinsic and extrinsic camera parameters are known either from pre-calibration or camera angular potentiometers and robot motion sensors. The camera has square pixels and zero skew factors, which is valid for most cameras.
- The robot takes frames periodically for the stereo reconstruction. During each period, we assume that the road surface can be approximated by a plane, which is the x-z plane of the WCS as illustrated in Fig. 1.
- We assume that the pixel correspondence error across different frames is uniformly distributed in the ICS. We believe that the pixel correspondence errors do not have an infinite tail distribution in reality and the uniform distribution is a conservative description of the property.

- We assume all CCSs are iso-oriented with the $CCS(\overline{F})$, which is determined by the navigation direction at time τ_0. Although the robot may have different positions and orientations when taking images, we can project the images into the CCSs that are iso-oriented with $CCS(\overline{F})$ using a perspective re-projection because we know accurate camera parameters.

3.3 Problem Context

For frames such as $\underline{F}$ and $\overline{F}$, we need to define their corresponding robot locations and camera parameters. As illustrated in Fig. 1, the camera is mounted at a height of h. Hence the camera position is uniquely defined by its coordinates (x_w, h, z_w) in the WCS. In order to have a good coverage of the road, the camera usually tilts towards the ground as illustrated in Fig. 1. The tilt angle is defined as t.

The previous period provides an obstacle-free road region R_f. The robot needs to stay in R_f and reach $T(\tau_0)$ at the end of the current period.

A camera frame usually covers a wide range, from adjacent regions to an infinite horizon. For navigational purposes, the robot is not interested in regions that are too far away. As illustrated in Fig. 1, the z-axis of the WCS points to the robot's forward direction at time $\tau = \tau_0$ when frame $\overline{F}$ is taken. z_M is defined as the maximal distance that the robot cares about in the next iteration of the algorithm. The region of interest R_i is a subset of camera coverage,

$$R_i = \{(x_w, z_w) | 0 \leq z_w \leq z_M, (x_w, z_w) \in \Pi(\overline{F})\}, \tag{1}$$

where x_w and z_w are defined in the WCS and function $\Pi(\overline{F})$ is the coverage of $\overline{F}$ in the x-z plane of the WCS. Our research problem is to understand how the depth error is associated with objects in R_i. To study how the depth error is distributed on the road plane, we introduce the *untrusted area* below.

3.4 Untrusted Area and Problem Formulation

The computed depth information is not accurate due to the image correspondence error. According to our assumptions, for a given pixel in $\underline{F}$, the corresponding pixel in $\overline{F}$ can be found with an error that is uniformly and independently distributed. Hence, the depth error is also a random variable. Define $e = z_w - \hat{z}_w$ as the depth error, which z_w is the true depth of the corresponding object in the WCS and $\hat{z}_w$ is the depth computed from the stereo reconstruction process. e has a range $|e| \leq |e_\Delta|$. The depth error range e_Δ will be formally defined later. We adopt $|e_\Delta|$ as the metric to characterize the quality of the depth information. $e_t > 0$ is a pre-defined threshold for $|e_\Delta|$. To facilitate robot navigation, we want to ensure that $|e_\Delta| \leq e_t$.

Although the image correspondence error is uniformly and independently distributed in the ICS, the influence of the image correspondence error on the depth is non-uniform due to a nonlinear stereo reconstruction process. For the two camera frames $\underline{F}$ and $\overline{F}$ taken from two different camera perspectives, we can construct the

depth map for the overlapping regions of the two frames $\Pi(\underline{F} \cap \overline{F})$. We define the untrusted area $A_u(\underline{F}, \overline{F})$ that is constructed by the image pair $(\underline{F}, \overline{F})$ in the WCS as

$$A_u(\underline{F}, \overline{F}) = \{(x_w, z_w) | (x_w, z_w) \in \Pi(\underline{F} \cap \overline{F}), |e_\Delta(x_w, z_w)| > e_t\}, \tag{2}$$

because we know that the depth information in A_u is untrustworthy due to the excessive $|e_\Delta|$. Our problem is,

Definition 3.1. For a given threshold $|e_\Delta|$, a pair of overlapping frames $(\underline{F}, \overline{F})$, and the corresponding camera parameters, compute $A_u(\underline{F}, \overline{F})$.

The error threshold $|e_\Delta|$ is not necessarily a constant. For example, we define $e_t = \rho z_w$ where ρ is the relative error threshold and $0 < \rho < 1$. The choice of $|e_\Delta|$ and ρ depends on how conservative the motion planning is. A smaller value results in larger A_u and the robot has to travel longer distance to avoid A_u. In our experiments, $\rho = 20\%$ works well for navigation purpose.

4 Analysis of Depth Error

4.1 Computing Depth from Two Views

In stereo vision, 3D information is computed through triangulation under the perspective projection based on the extracted correspondence points from each pair of images[27]. Define $\underline{c}$ and $\overline{c}$ as camera centers for frames $\underline{F}$ and $\overline{F}$, respectively. Define $\underline{P}$ and $\overline{P}$ as the camera projection matrices for $\underline{F}$ and $\overline{F}$, respectively. Since the CCSs of $\underline{F}$ and $\overline{F}$ are iso-oriented and only differ from the WCS by a tilt value t in orientation, the orientation of the WCS with respect to the CCSs can be expressed by a rotation matrix

$$R_X(-t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(t) & s(t) \\ 0 & -s(t) & c(t) \end{bmatrix}.$$

Note that we use $s(\cdot)$ and $c(\cdot)$ to denote $\sin(\cdot)$ and $\cos(\cdot)$, respectively. If CCSs are not iso-oriented, it is not difficult to extend the rotation matrix using Euler angle sets. The origin of the WCS with respect to the CCSs of $\underline{F}$ and $\overline{F}$ are defined as $\underline{W}$ and $\overline{W}$, respectively. Since $T(0) = [x_w(0), z_w(0)]^T$, $T(\tau_0) = [0,0]^T$, and the camera height is h, the camera center positions with respect to the WCS are $\underline{c} = [x_w(0), h, z_w(0)]^T$ and $\overline{c} = [0, h, 0]^T$, respectively. Then we have,

$$\underline{W} = -R_X(-t)\underline{c}, \text{ and } \overline{W} = -R_X(-t)\overline{c}.$$

Therefore,

$$\underline{P} = K[R_X(-t)|\underline{W}], \ \overline{P} = K[R_X(-t)|\overline{W}], K = diag(f, f, 1),$$

where f is the focal length of the camera divided by the side length of a pixel. Let $\underline{q} = [\underline{u}\ \underline{v}\ 1]^T$ and $\overline{q} = [\overline{u}\ \overline{v}\ 1]^T$ be a pair of corresponding points in $\underline{F}$ and $\overline{F}$,

respectively. Define $Q = [x_w, y_w, z_w]^T$ as their corresponding point in WCS. Let $\underline{Q}_c = [\underline{x}_c\ \underline{y}_c\ \underline{z}_c]^T$ and $\overline{Q}_c = [\overline{x}_c\ \overline{y}_c\ \overline{z}_c]^T$ be Q's position in the CCSs of $\underline{F}$ and $\overline{F}$, respectively. Also, we know that $\underline{Q}_c$ and $\overline{Q}_c$ can be expressed as,

$$\underline{Q}_c = R_X(-t)Q + \underline{W}, \text{ and } \overline{Q}_c = R_X(-t)Q + \overline{W}. \tag{3}$$

The following holds according to the pin-hole camera model,

$$\underline{q} = \frac{1}{\underline{z}_c}\underline{P}\begin{bmatrix} Q \\ 1 \end{bmatrix} = \frac{1}{\underline{z}_c}K\underline{Q}_c, \text{ and } \overline{q} = \frac{1}{\overline{z}_c}\overline{P}\begin{bmatrix} Q \\ 1 \end{bmatrix} = \frac{1}{\overline{z}_c}K\overline{Q}_c. \tag{4}$$

From (3), we know $\underline{Q}_c = \overline{Q}_c + (\underline{W} - \overline{W})$, namely,

$$\underline{x}_c = \overline{x}_c - x_w(0),\ \ \underline{y}_c = \overline{y}_c - z_w(0)s(t), \text{ and } \underline{z}_c = \overline{z}_c - z_w(0)c(t). \tag{5}$$

From (4) and (5), we obtain,

$$\underline{q} = \frac{1}{\overline{z}_c - z_w(0)c(t)}\left(\overline{z}_c\overline{q} + K(\underline{W} - \overline{W})\right). \tag{6}$$

Since K, $\underline{W}$, $\overline{W}$, $\underline{q}$, and $\overline{q}$ are known, (6) consists of a system of equations with $\overline{z}_c$ as an unknown quantity. There is one unknown variable and a total of two equations (e.g. the first two equations in (6)). This is an overly-determined equation system. A typical approach would be to apply a least-square (LS) method [27]. Using the solution from LS method would result in a high-order polynomial when analyzing the depth error. Solving the high-order polynomial is computationally inefficient. Another method is to simply discard one equation and solve it directly. This method has a speed advantage and its solution quality is slightly worse than that of the LS method. The advantage is that the format of solution can be expressed in simpler format that allows us to derive the depth error distribution. Actually, a worse solution can actually provide a more conservative error bound than that of the LS method. Employing the method and solving the first equation in (6), we have $\overline{z}_c = \frac{x_w(0)f - \underline{u}z_w(0)c(t)}{\overline{u} - \underline{u}}$. From (3), we know $z_w = \overline{z}_c\left(\frac{\overline{v}}{f}s(t) + c(t)\right)$. Hence,

$$z_w = \frac{x_w(0)f - \underline{u}z_w(0)}{\overline{u} - \underline{u}}\left(\frac{\overline{v}}{f}s(t) + c(t)\right). \tag{7}$$

Depth z_w describes the distance from the robot to an obstacle along the z-axis of the WCS. Its error directly affects the robot's collision avoidance performance.

4.2 Estimating the Depth Error Range

For the given pair of corresponding points $(\underline{q}, \overline{q})$ from $(\underline{F}, \overline{F})$ with camera centers ($\underline{c}$, $\overline{c}$), the geometric interpretation of the above triangulation process is the following. If we back project a ray from $\underline{c}$ through $\underline{q}$, it intersects with the ray generated by

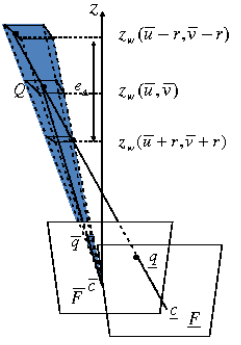


Fig. 2. An illustration of depth error caused by the image correspondence error in $\overline{F}$. The intersection zone between the ray from $\underline{q}$, and the pyramid from $\overline{q}$ is the error range. If the error range projects onto the z axis, it is always bound between $z_w(\overline{u}+r,\overline{v}+r)$ and $z_w(\overline{u}-r,\overline{v}-r)$.

back-projecting from $\overline{c}$ through $\overline{q}$, provided that both $\underline{q}$ and $\overline{q}$ are accurate. The intersection point in the 3D space is Q; see Fig. 2.

However, for a given point $\underline{q}$, finding the accurate $\overline{q}$ is unlikely due to noises and pixelization errors. According to our assumptions, the corresponding errors in $\overline{u}$ and $\overline{v}$ are independently distributed according to $U(-r,r)$, where r is usually 0.5-2 pixels in length. This means that $\overline{q}$ is distributed in a small square on $\overline{F}$. When we back project the square, it forms a pyramid in 3D space as illustrated in Fig. 2. When the pyramid meets the ray that is back-projected from $\underline{q}$, it has a range of intersections instead of a single point. The estimated depth z_w is a function of random variables $(\overline{u},\overline{v})$ and can be expressed as $z_w(\overline{u},\overline{v})$. It is apparent that z_w is a random variable that could take any value in this intersection zone.

To compute the intersection zone, we need to compute the intersection points between the ray from $\underline{c}$ through $\underline{q}$ and all four side planes of the pyramid. However, the solution cannot be expressed in a closed-form for further analysis. Instead, we employ the upper and the lower bounds of the length of the intersection zone as illustrated in Fig. 2. Then the bound of the maximum length of the intersection zone is defined as $|e_\Delta|$, where

$$e_\Delta = z_w(\overline{u}+r,\overline{v}+r) - z_w(\overline{u}-r,\overline{v}-r). \tag{8}$$

We skip the process of deriving the bound due to the page limit. The intuition is that any line segment bounded inside the pyramid truncated between plane $z_w(\overline{u}+r,\overline{v}+r)$ and $z_w(\overline{u}-r,\overline{v}-r)$ is shorter than e_Δ. Similarly, another choice is $z_w(\overline{u}-r,\overline{v}+r)-z_w(\overline{u}+r,\overline{v}-r)$. Since both the analysis method and results are similar, we use (8) in the rest of the paper.

$|e_\Delta|$ describes the range of the depth error and is employed as the metric to measure the quality of the stereo reconstruction. To simplify the notation in computing e_Δ, we define the following intermediate variables for (7).

$$\lambda = \beta\overline{v} + c(t),\ \ \zeta_d = \overline{u} - \underline{u},\ \ \beta = \frac{s(t)}{f},\ \zeta_n = x_w(0)f - \underline{u}z_w(0). \tag{9}$$

Then $z_w = \lambda\frac{\zeta_n}{\zeta_d}$ according to (7) and (9). Substituting them into (8), we have,

$$e_\Delta = (\lambda + r\beta)\frac{\zeta_n}{\zeta_d + r} - (\lambda - r\beta)\frac{\zeta_n}{\zeta_d - r} = \zeta_n\frac{2r(\beta\zeta_d - \lambda)}{\zeta_d^2 - r^2}. \tag{10}$$

Eq. (10) illustrates e_Δ in the ICS. For robot navigation purposes, we are interested in e_Δ in the x-z plane of the WCS. Hence $\underline{u}$, $\overline{u}$ and $\overline{v}$ in (10) should be transformed into functions of x_w and z_w. Recall that $s(\cdot)$ and $c(\cdot)$ denote $\sin(\cdot)$ and $\cos(\cdot)$, respectively. From (4), (7), and (9), we know $\overline{u} = \frac{x_w f}{\overline{z}_c} = f\lambda\frac{x_w}{z_w}$, and $y_w = \left(\frac{\overline{v}}{f}c(t) - s(t)\right)\overline{z}_c + h$. Since we are interested in obstacles on the x-z plane, $y_w = 0$, we have $\overline{v} = \frac{f(z_w s(t) - hc(t))}{z_w c(t) + hs(t)}$. Similarly, from (4), (7), and (9), we know $\underline{u} = \alpha_x x_w + \alpha_0$, where $\alpha_x = \frac{f}{\overline{z}_c - z_w(0)c(t)} = \frac{f\lambda}{z_w - z_w(0)c(t)\lambda}$, and $\alpha_0 = -x_w(0)\alpha_x$. Plugging into (9), we obtain the intermediate variables λ, ζ_n, and ζ_d, in terms of x_w and z_w.

$$\lambda = \frac{z_w}{z_w c(t) + hs(t)},\ \zeta_n = n_x x_w + n_0,\ \text{and}\ \zeta_d = \frac{n_x\lambda}{z_w}x_w + \frac{n_0\lambda}{z_w},$$

where $n_x = -z_w(0)c(t)\alpha_x$ and $n_0 = x_w(0)z_w\alpha_x/\lambda$. Plugging them into (10), we obtain e_Δ as a function of x_w and z_w,

$$e_\Delta = \frac{2r\beta\lambda z_w(n_x x_w + n_0)^2 - 2r\lambda z_w^2(n_x x_w + n_0)}{\lambda^2(n_x x_w + n_0)^2 - r^2 z_w^2}. \tag{11}$$

For an obstacle located at $(x_w, 0, z_w)$, Eq. (11) allows us to estimate e_Δ. It is clear that the depth error range varies dramatically in different regions, and thus should be considered in robot navigation to avoid obstacles.

4.3 *Predicting Untrusted Area*

For a given frame pair with the corresponding robot locations, we can partition R_i using a preset depth error threshold $e_t > 0$. We are now ready to predict A_u by computing its boundary using Eq. (11).

4.3.1 Partition R_i According to the Sign of e_Δ

To find the regions corresponding to $|e_\Delta| < e_t$, there are two possible cases to consider: $e_\Delta < 0$ and $e_\Delta > 0$. We can rewrite (11) as,

$$e_\Delta = \frac{2r\lambda z_w(x_w - \mu_{n1})(x_w - \mu_{n2})}{(x_w - \mu_{d1})(x_w - \mu_{d2})}, \tag{12}$$

where

$$\mu_{n1} = \frac{x_w(0)}{z_w(0)\lambda c(t)} z_w, \quad \mu_{n2} = \frac{x_w(0)}{z_w(0)\lambda c(t)} z_w - \frac{z_w(z_w - z_w(0)\lambda c(t))}{f z_w(0)\lambda \beta c(t)},$$
$$\mu_{d1} = \frac{x_w(0)}{z_w(0)\lambda c(t)} z_w + \frac{r z_w(z_w - z_w(0)\lambda c(t))}{f z_w(0)\lambda^2 c(t)},$$
$$\mu_{d2} = \frac{x_w(0)}{z_w(0)\lambda c(t)} z_w - \frac{r z_w(z_w - z_w(0)\lambda c(t))}{f z_w(0)\lambda^2 c(t)}.$$

Recall that t is the camera tilt angle and a typical camera setup has $0 \leq t \leq 30^\circ$. A regular camera would have a focal length of 5-100 mm and pixel side length of 5-10 μm. Therefore, $f \geq 100$. Since $\beta = s(t)/f$,

$$0 < \beta \leq \sin(30^\circ)/100 = 0.005. \tag{13}$$

Also we know that

$$\lambda = \beta \overline{v} + c(t) = s(t)\frac{\overline{v}}{f} + c(t) > \beta \tag{14}$$

because $|\frac{\overline{v}}{f}| < 1$ for any camera with a vertical field of view less than 90°. Combining this information, we have $0 < \beta < r/\lambda$ and $\beta < \lambda$. For obstacles in R_i, $z_w > 0$ according to the definition of the WCS. Also $z_w(0) < 0$ as illustrated in Fig. 1. Hence, we have

$$\frac{z_w(z_w - z_w(0)\lambda c(t))}{f z_w(0)\lambda c(t)} < 0. \tag{15}$$

Combining the inequalities above, we can derive the following relationship:

$$\mu_{d1} < \mu_{n1} < \mu_{d2} < \mu_{n2}. \tag{16}$$

Combining (16) with (12), we have,

$$e_\Delta > 0 \text{ if } \mu_{n1} < x_w < \mu_{d2} \text{ or } x_w < \mu_{d1}, \tag{17}$$
$$e_\Delta < 0 \text{ if } \mu_{d2} < x_w < \mu_{n2} \text{ or } \mu_{d1} < x_w < \mu_{n1}. \tag{18}$$

We ignore the region $x_w > \mu_{n2}$ in $e_\Delta > 0$ as this region is always outside of the camera's coverage.

We are now ready to compute A_u for the two cases defined in (17) and (18).

4.3.2 Computing A_u for $e_\Delta > 0$

This is the case illustrated in Fig. 2(a). Recall that the untrusted area satisfies $e_\Delta > e_t$. It is worth mentioning that the error threshold e_t is usually not a fixed number but a function of z_w. Recall that $e_t = \rho z_w$ where ρ is the relative error threshold. There are two cases: Case (i): $x_w < \mu_{d1}$ and Case (ii): $\mu_{n1} < x_w < \mu_{d2}$.

Case (i): when $x_w < \mu_{d1}$, the denominator of e_Δ in (12) is positive. Plug (12) into $e_\Delta > e_t$, and we have

$$(e_t\lambda^2 - 2r\beta\lambda z_w)n_x^2 x_w^2 + (2(e_t\lambda^2 - 2r\beta\lambda z_w)n_x n_0 + 2r\lambda n_x z_w^2)x_w + (e_t\lambda^2 - 2r\beta\lambda z_w)n_0^2 - e_t r^2 z_w^2 + 2r\lambda n_0 z_w^2 < 0. \tag{19}$$

The solution to the quadratic inequality (19) is

$$\frac{-\kappa_1 - \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2} < x_w < \frac{-\kappa_1 + \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2}, \tag{20}$$

where

$$\kappa_2 = (e_t\lambda^2 - 2r\beta\lambda z_w)n_x^2, \quad \kappa_1 = 2(e_t\lambda^2 - 2r\beta\lambda z_w)n_x n_0 + 2r\lambda n_x z_w^2,$$
$$\kappa_0 = (e_t\lambda^2 - 2r\beta\lambda z_w)n_0^2 - e_t r^2 z_w^2 + 2r\lambda n_0 z_w^2.$$

The untrusted area is the region that satisfies (20) and $x_w < \mu_{d1}$. To compute the intersection, we need to understand the relationship between the solution in (20) and the coefficients in (12). Combining them, we know,

$$\mu_{d1} - \frac{-\kappa_1 - \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2} = \frac{rz_w(z_w - z_w(0)\lambda c(t))}{fz_w(0)\lambda^2 c(t)}\left(1 - \frac{\lambda + \sqrt{\lambda^2 + \rho^2\lambda^2 - 2r\beta\lambda\rho}}{\rho\lambda - 2r\beta}\right).$$

Notice that $0 < r \leq 2$, $0 < \rho < 1$, β is very small according to (13), and $\lambda > 0$ according to (14). Therefore, $2r\beta$ and $2r\beta\lambda\rho$ are close to zero. Hence, we approximate $\left(1 - \frac{\lambda + \sqrt{\lambda^2 + \rho^2\lambda^2 - 2r\beta\lambda\rho}}{\rho\lambda - 2r\beta}\right) \approx 1 - \frac{2}{\rho} < 0$. Combining this equation with (15), we know,

$$\mu_{d1} > \frac{-\kappa_1 - \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2}. \tag{21}$$

Similarly, we can obtain

$$\mu_{d1} < \frac{-\kappa_1 + \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2}. \tag{22}$$

According to (20), (21), (22), and $x_w < \mu_{d1}$, the untrusted area for this case is given by,

$$\frac{-\kappa_1 - \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2} < x_w < \mu_{d1}. \tag{23}$$

Case (ii): when $\mu_{n1} < x_w < \mu_{d2}$, from (16), we know that the denominator of (12) is negative. Hence, $\kappa_2 x_w^2 + \kappa_1 x_w + \kappa_0 > 0$. Similar to the analysis in Case (i), we obtain,

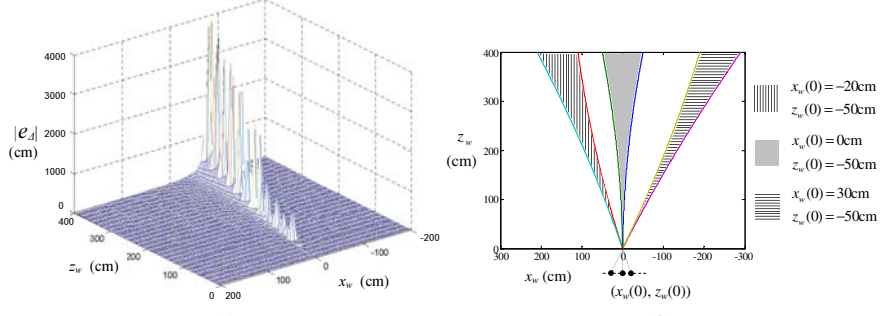
$$\frac{-\kappa_1 + \sqrt{\kappa_1^2 - 4\kappa_2\kappa_0}}{2\kappa_2} < x_w < \mu_{d2}. \tag{24}$$

Similarly, we can compute A_u for $e_\Delta < 0$. Combing the results, we can obtain the boundaries of A_u. Represented as a function of x_w, we define the lower boundary and the upper boundary of A_u as x_w^- and x_w^+, respectively. Hence we have the two boundaries

$$x_w^{\mp}(z_w, x_w(0), z_w(0)) = \frac{x_w(0) z_w}{z_w(0)\lambda c(t)} + \frac{r z_w (z_w - z_w(0)\lambda c(t))}{f z_w(0)\lambda c(t)(\pm e_t\lambda^2 - 2r\beta\lambda z_w)} (z_w\lambda + \sqrt{\lambda^2 z_w^2 + e_t^2\lambda^2 \mp 2re_t\beta\lambda z_w}). \tag{25}$$

5 Experiments

We have verified our analysis for the depth error estimation using a three-wheeled mobile robot. The robot has two front driving wheels and one rear castor. The robot is 30 cm long, 30 cm wide, 33 cm tall and can travel at a speed of 25 cm/s with a 25 lbs payload. It is also equipped with two wheel encoders and a digital compass. The camera mounted on the robot is a Canon VCC4 pan-tilt-zoom camera with a 47.5° horizontal field of view. The camera mounting height $h = 44$ cm. The intrinsic camera parameters are estimated using the Matlab calibration toolbox[28]. During the experiment, we set $z_M = 4$ m and $t = 15°$ according to our robot and camera configurations. We conducted the experiments in the H. R. Bright Bldg. at Texas A&M University. The obstacles used in the experiments are books and blocks with a size of 20 cm × 14.5 cm × 10 cm.



(a) (b)

Fig. 3. (a) An illustration of $|e_\Delta|$. Robot positions are set to be $x_w(0) = 10$ cm, $z_w(0) = -50$ cm. (b) A_us with different robot positions $(x_w(0), z_w(0))$, which are the black dots in the figure. We set the threshold $e_t = 0.2 z_w$.

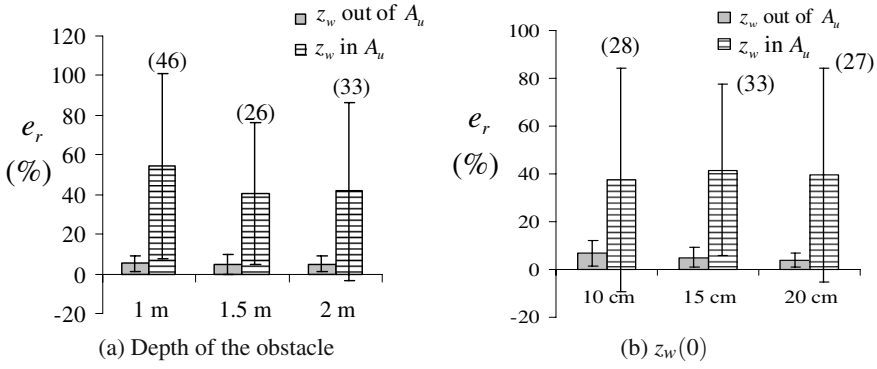


Fig. 4. The effectiveness of depth error reduction. The height of the bar is the mean value of e_r and the vertical interval represents the variance of e_r. The number in the parenthesis is the total number of trials.

Fig. 3a illustrates how $|e_\Delta|$ is distributed on the road plane $y_w = 0$ according to our analysis. The 3D mesh is just an approximation of actual $|e_\Delta|$ distribution because it is generated by a finite set of testing locations. The illustration avoids the points on the baseline because the corresponding error range is infinite. It is apparent that the depth error is excessive in the area that is close to the camera baseline.

The second test is to show to how the different camera perspectives affect the location of A_u. Fig. 3b gives three examples of A_u for different camera perspectives $(x_w(0), z_w(0))$. It is clear that the selection of perspective can determine the location of A_u.

We also compared the depth error for objects inside and outside A_u in actual robot navigation. To facilitate the comparison, we defined the relative depth error in percentage $e_r = \frac{|e|}{z_w} \times 100$, where z_w is the measured depth that is used as a ground truth. We compare e_r for objects inside and outside the A_u for two scenarios: (a) the different depth of objects and (b) different robot positions as illustrated in Fig. 4. In (a), in each trial, the testing objects are randomly placed with a fixed depth. In (b), we change the relative position between two camera perspectives to verify the depth error with respect to A_u. Obstacles are randomly placed in each trial. The accurate total number of trials for each setup is shown above the bars in the figures. In both (a) and (b), we first compute the obstacle depth using stereo vision and then compare it with the measured ground truth by computing e_r. Note that the mean and the variance of e_r are significantly reduced if the robot stays outside A_u.

6 Conclusion and Future Work

We analyzed the depth error range distribution across the camera coverage for a mobile robot equipped with a single camera. For SFM-based stereo vision for navigation, we showed that the depth error can be excessively large and hence cause

collisions in robot navigation. We defined and modeled the untrusted area where the depth error range is beyond a preset threshold. Physical experiment results confirmed our analysis. In the future, we will apply the analysis into a new robot motion planning algorithm that will purposefully generate trajectories to avoid the untrusted area. The introduction of the untrusted area will help us to add more camera perspectives for the SFM. The introduction of the predictive model of the untrusted area opens a door to add depth-error aware planning into a variety of applications involving the monocular vision system. It is possible to use the untrusted area to guide the visual landmark selection for SLAM. Similarly, the untrusted area can be used to improve visual tracking performance when the robot plans to follow a moving target.

Acknowledgement

We thank R. Volz and R. Gutierrez-Osuna for their insightful discussions. Thanks for Y. Xu, C. Kim, H. Wang, N. Qin, B. Fine, and T. Southard for their inputs and contributions to the Netbot Laboratory, Texas A&M University.

References

1. Chaumette, F., Boukir, S., Bouthemy, P., Juvin, D.: Structure from controlled motion. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(5), 492–504 (1996)
2. Bajcsy, R.: Active perception. Proceedings of the IEEE 76(8), 996–1005 (1988)
3. Zavidovique, B.: First steps of robotic perception: The turning point of the 1990s. Proceedings of the IEEE 90(7), 1094–1112 (2002)
4. Tarabanis, K., Allen, P., Tsai, R.: A survey of sensor planning in computer vision. IEEE Transactions on Robotics and Automation 11(1), 86–104 (1995)
5. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Monocular vision based slam for mobile robots. In: The 18th International Conference on Pattern Recognition, August 2006, pp. 1027–1031 (2006)
6. Chen, Z., Rodrigo, R., Samarabandu, J.: Implementation of an update scheme for monocular visual slam. In: International Conference on Information and Automation, December 2006, pp. 212–217 (2006)
7. Mortard, E., Raducanu, B., Cadenat, V., Vitria, J.: Incremental on-line topological map learning for a visual homing application. In: IEEE International Conference on Robotics and Automation, April 2007, pp. 2049–2054 (2007)
8. Lemaire, T., Lacroix, S.: Monocular-vision based slam using line segments. In: IEEE International Conference on Robotics and Automation, April 2007, pp. 2791–2796 (2007)
9. Royer, E., Bom, J., Michel Dhome, B.T., Lhuillier, M., Marmoiton, F.: Outdoor autonomous navigation using monocular vision. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, August 2005, pp. 1253–1258 (2005)
10. Chen, Z., Birchfield, S.T.: Qualitative vision-based mobile robot navigation. In: IEEE International Conference on Robotics and Automation, Olrando, FL, May 2006, pp. 2686–2692 (2006)
11. Michels, J., Saxena, A., Ng, A.: High speed obstacle avoidance using monocular vision and reinforcement learning. In: 22nd International Conference on Machine Learning, August 2005, pp. 593–600 (2005)

12. Song, D., Lee, H., Yi, J., Levandowski, A.: Vision-based motion planning for an autonomous motorcycle on ill-structured roads. Autonomous Robots 23(3), 197–212 (2007)
13. Azarbayejani, A., Pentland, A.: Recursive estimation of motion, structure, and focal length. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(6), 562–575 (1995)
14. Jebara, T., Azarbayejani, A., Pentland, A.: 3D structure from 2D motion. IEEE Signal Processing Magazine 16(3), 66–84 (1999)
15. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International Journal of Computer Vision 9(2), 137–154 (1992)
16. Martinec, D., Pajdla, T.: 3D recontruction by fitting low-rank matrices with missing data. In: IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, June 2005, pp. 198–205 (2005)
17. Brandt, S.: Closed-form solutions for affine reconstruction under missing data. In: 7th European Conference on Computer Vision, Copenhagen, Denmark, May 2002, pp. 109–114 (2002)
18. Hartley, R., Schaffalizky, F.: Powerfactorization: 3D reconstruction with missing or uncertain data. In: Australia-Japan Advanced Workshop on Computer Vision (September 2003)
19. Guilbert, N., Bartoli, A.: Batch recovery of multiple views with missing data using direct sparse solvers. In: British Machine Vision Conference, Norwich, UK (September 2003)
20. Anandan, P., Irani, M.: Factorization with uncertainty. International Journal of Computer Vision 49(3), 101–116 (2002)
21. Triggs, B.: Plane+parallax, tensors and factorization. In: 6th European Conference on Computer Vision, Dublin, Ireland, June 2000, pp. 522–538 (2000)
22. Irani, M., Anandan, P., Cohen, M.: Direct recovery of planar-parallax from multiple frames. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(11), 1528–1534 (2002)
23. Rother, C., Carlsson, S.: Linear multi view reconstruction and camera recovery using a reference plane. International Journal of Computer Vision 49(3), 117–141 (2002)
24. Bartoli, A., Sturm, P.: Contrained structure and motion from multiple uncalibrated views of a piecewise planar scene. International Journal of Computer Vision 52(1), 45–64 (2003)
25. Dellaert, F., Seitz, S.M., Thorpe, C.E., Thrun, S.: Structure from motion without correspondence. In: IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head, SC, June 2000, pp. 557–564 (2000)
26. Chowdhury, A.K.R., Chellappa, R.: Statistical bias in 3-D reconstruction from a monocular video. IEEE Transactions on Image Processing 14(8), 1057–1062 (2005)
27. Hartley, R., Zisserman, A.: Multiple View Geometry in computer vision. Cambridge University Press, Cambridge (2003)
28. Bouguet, J.-Y.: Camera calibration toolbox for matlab (2007), `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`

Sensor Beams, Obstacles, and Possible Paths

Benjamin Tovar, Fred Cohen, and Steven M. LaValle

Abstract. This paper introduces a problem in which an agent (robot, human, or animal) travels among obstacles and binary detection beams. The task is to determine the possible agent path based only on the binary sensor data. This is a basic filtering problem encountered in many settings, which may arise from physical sensor beams or virtual beams that are derived from other sensing modalities. Methods are given for three alternative representations: 1) the possible sequences of regions visited, 2) path descriptions up to homotopy class, and 3) numbers of times winding around obstacles. The solutions are adapted to the minimal sensing setting; therefore, precise estimation, distances, and coordinates are replaced by topological expressions. Applications include sensor-based forensics, assisted living, security, and environmental monitoring.

1 Introduction

Imagine installing a bunch of cheap, infrared eye beams throughout a complicated warehouse, office, or shopping center; see Figure 1. Just like the safety beam on a motorized garage door, a single bit of information is provided: Is the beam currently obstructed? Now suppose that there are one or more moving bodies, which could be people, robots, animals, and so on. If the beams are distinguishable and we know the order in which beams were crossed, what can we infer about the paths taken by the moving bodies? This may be considered as a filtering problem, but with minimal, combinatorial information, in contrast to popular Kalman filters and particle filters.

This paper proposes the study of inference problems that arise from bodies crossing beams among obstacles. It turns out that the subject is much more general than the particular scenario just described. In addition to binary detection beams or

Benjamin Tovar and Steven M. LaValle
Department of Computer Science, University of Illinois. Urbana, IL 61801 USA
e-mail: {btovar,lavalle}@uiuc.edu

Fred Cohen
Department of Mathematics, University of Rochester. Rochester, NY 14627 USA
e-mail: fcohen@rochester.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 317–332.
springerlink.com

regions placed around an environment, the mathematical model arises in other contexts. For example, if a robot carries a camera and certain image features critically change, then the event may be equivalent to crossing a "virtual" beam in the environment (see Section 3).

Our questions are inspired by many problems that society currently faces. There is widespread interest in developing *assisted living* systems that use sensors to monitor the movements of people in their homes or hospitals. How much can be accomplished with simple detection beams, which are affordable, robust, and respect privacy? Alternatively, imagine the field of *sensor-based forensics*, in which police investigators or lawyers would like to corroborate or refute a testimony about how people moved at a crime scene. A simple verification test might based on the sequence of beam crossings might establish that someone is lying. Other problems include tracking wildlife movement for conservation purposes, landmark-based navigation with outdoor vehicles, sensor-assisted safe child care, and security.

Suppose there is one moving body, called an *agent*, and we have the information that a sequence of beams was crossed. We focus on three kinds of questions: 1) Supposing regions are delineated by the arrangement of beams, what possible sequences of regions did the agent visit? 2) What path did the agent take up to homotopy? 3) How many times did the agent wind around each obstacle? These questions form the basis of Sections 4 to 6. Multiple agents are briefly considered in Section 7. The last two questions are familiar problems in topology and group theory, and are motivated by homotopy and homology, respectively. In particular, the topic is close to *word problems* in group theory, in which it must be determined whether two words (e.g., $abac^{-1}b$ and cab^{-1}) are the same group element. In the general group-theoretic setting, such questions go back to 1910 with Dehn's fundamental problems (see [9]).

The most closely related works are algorithms to decide whether two paths in a punctured plane are homotopic [1, 3]. These algorithms are based on extending vertical lines from each of the punctures. The vertical lines serve two purposes:

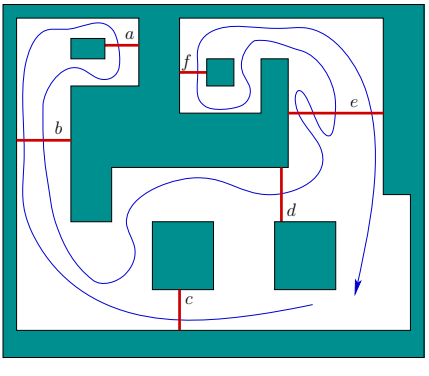


Fig. 1. What can be determined about the path using only the word *cbabdeeefe*, which indicates the sequence of sensor beams crossed?

First, any given path is represented with the sequence of vertical rays it intersects. Second, they connect the different fibers of a covering space of the punctured plane. In this context, two paths are homotopic if and only if they have the same endpoints when they are lifted to the universal covering space. The novelty in our work is that we start with *sensor* words and must first convert them into path descriptions. This represents an *inverse problem* that is constrained by the geometry and topology of the sensors and obstacles.

Minimal sensing requirements have been studied to solve a particular task [4]. This typically involves a characterization and simplification of the information space associated with the task [7], which considers the whole histories of commands given to the actuators and sensing observations. Following a minimal sensing perspective in the context of sensor networks, works such as [10] detect and count targets using binary proximity sensors. The binary proximity sensors can be considered as *overlapping beams* that go off when an agent is in range. Tracking targets is done with a particle filter, in which each particle is a candidate trajectory of a target. In [12] the location of moving agents is inferred from combinatorial changes in sensing observations. Such combinatorial changes may correspond to visual events, which can be abstracted in our work as sensor beams. An example of this is presented in this paper in Section 3, in which the combinatorial changes correspond to crossing of landmarks [11]. The careful consideration of visual events is the basis for solutions to problems such as localization [2, 6] and visibility-based pursuit-evasion [5, 8].

2 Problem Formulation

Let $W \subseteq \mathbb{R}^2$ be the closure of a contractible open set. A common case is $W = \mathbb{R}^2$. Let $\mathscr{O}$ be a set of n pairwise-disjoint *obstacles*, which are each the closure of a contractible open set. Let X be the *free space*, which is the open subset of W that has all $o \in \mathscr{O}$ removed. Let $\mathscr{B}$ be a set of m *beams*, each of which is an open linear subset of X. If W is bounded, then every beam is a line segment with both endpoints on the boundary of X. (Note that beams may connect an obstacle boundary to itself, another obstacle's boundary, or the boundary of W; also, a beam may connect the boundary of W to itself.) If W is unbounded, then some beams may be open rays that emanate from the boundary of an obstacle or even lines that are contained in the interior of W.

Regions: The collection of obstacles and beams induces a decomposition of X into connected *cells*. If the beams in $\mathscr{B}$ are pairwise disjoint, then each $B \in \mathscr{B}$ is a 1-cell and the 2-cells are maximal regions bounded by 1-cells and portions of the boundary of X. If beams intersect, then the 1-cells are maximal segments between any beam intersection points or boundary elements of X; the 2-cells follow accordingly. Every 2-cell will be called a *region*.

Agent path: Suppose that an *agent* moves along a *state trajectory* (or path) $\tilde{x} : [0,1] \rightarrow X$, in which $[0,1]$ is imagined as a time interval. (Alternatively, $[0,t_f]$ could

be allowed for any $t_f > 0$; however, this flexibility is unnecessary because speed and time scaling are irrelevant to our questions.)

Sensor model: If the agent crosses a beam, what exactly is observed? Assume that the set of possible $\tilde{x}$ is restricted so that: 1) every beam crossing is transverse (the agent cannot "touch" a beam without crossing it and the agent cannot move along a beam) and 2) the agent never crosses an intersection point between two or more beams (if any such intersections exist).

Let L be a finite set of *labels*. Suppose each beam is assigned a unique label by some bijection $\alpha : \mathscr{B} \to L$. The sensor model depicted in Figure 1 can be obtained by a sensor mapping $h : X \to Y$, in which $Y = L \cup \{\varepsilon\}$ is the *observation set*. If $\tilde{x}(t) \in B$ for some $B \in \mathscr{B}$, then $h(\tilde{x}(t)) = \alpha(B)$; otherwise, $h(\tilde{x}(t)) = \varepsilon$, which is a special symbol to denote "no beam". This is referred to as the *undirected beam* model because it indicates that the beam was crossed, but we do not know the direction.

To obtain a *directed beam* model, let $D = \{-1, 1\}$ be a set of *directions*. In this case, the observation set is $Y = (L \times D) \cup \{\varepsilon\}$, and the sensor mapping yields the orientation of each beam crossing (note that in addition to $\tilde{x}(t)$, the sensor mapping must now know what side of the beam the agent was on at time t^-).

So far, the beams have been *fully distinguishable* because α is a bijection. It is possible to make $|L| < m$ (the number of beams) and obtain some *indistinguishable* beams, in which case $\alpha : \mathscr{B} \to L$ is not bijective.

If a collection of beams is disjoint, distinguishable, and directed, the case will be referred to as *ddd-beams*, which is the best situation.

Sensor words: What observations are accumulated after $\tilde{x}$ is traversed? We assume that all ε observations are ignored, resulting in a sequence $\tilde{y}$, called the *sensor word*, of the remaining observations ($\tilde{y}$ is a kind of *observation history* [7]). For the example in Figure 1, suppose $L = \{a, b, c, d, e, f\}$. In the case of a undirected beams, the sensor word is $cbabdeeefe$. If the beams were directed so that left-to-right and bottom-to-top are the "forward" direction, then the sensor word could be encoded as $c^{-1}ba^{-1}b^{-1}dee^{-1}efe^{-1}$. For each $l \in L$, l denotes the forward direction and l^{-1} denotes the backward direction.

Inference: Let $\tilde{Y}$ be the set of all possible sensor words and let $\tilde{X}$ be the set of all possible state trajectories. Let $\phi : \tilde{X} \to \tilde{Y}$ denote the mapping that produces the sensor word $\tilde{y} = \phi(\tilde{x})$.

Suppose that $\tilde{y}$ has been obtained with no additional information. What can be inferred about $\tilde{x}$? Let $\phi^{-1}(\tilde{y})$ denote the *preimage* of $\tilde{y}$:

$$\phi^{-1}(\tilde{y}) = \{\tilde{x} \in \tilde{X} \mid \tilde{y} = \phi(\tilde{x})\}. \tag{1}$$

The inference problem amounts to: Under what conditions can we compute a useful description of the preimages?

The following are three ways to partially characterize these preimages, forming the basis of Sections 4, 5, and 6, respectively:

1. Using $\tilde{y}$, compute the set of possible sequences of regions visited by $\tilde{x}$. This characterizes $\phi^{-1}(\tilde{y})$ in a stage-by-stage manner.
2. If there are n obstacles, the *fundamental group* $\pi_1(X)$ is the free group F_n on n letters. Using some combinations of initial conditions, and assuming fixed starting and stopping paths, compute a representation of the possible paths as an element of F_n. This clearly throws away some information by considering only the homotopy equivalence class of paths. Hence, it is an over-approximation of $\phi^{-1}(\tilde{y})$.
3. Compute the signed number of windings around each obstacle, in which the order that windings are made is dropped. This throws away even more information than the homotopy equivalence class; therefore, it yields an even larger over-approximation of $\phi^{-1}(\tilde{y})$.

Using these representations, we can answer questions such as: Do two sensor words correspond to homotopic paths? Do they correspond to homologous paths? Is it possible for them to visit the same regions in the same order?

Combinatorial filters: Whenever possible, we try to design a *filter*, which computes statistics incrementally as new data are obtained. The most common example is the Kalman filter, which computes the next mean and covariance based on the new sensor reading and the previous mean and covariance. In this paper, we design *combinatorial filters*, which are minimalist non-probabilistic analogs to Bayesian filters. For a sensor word $\tilde{y}_k$ of length k, let $\kappa(\tilde{y}_k)$ denote a statistic, which could, for example, be the set of possible current regions. A combinatorial filter efficiently computes $\kappa(\tilde{y}_{k+1})$ using only $\kappa(\tilde{y}_k)$ and y_{k+1}, in which y_{k+1} is the last (most recent) letter in $\tilde{y}_{k+1}$. This implies that $\tilde{y}_k$ does not need to be stored in memory; only $\kappa(\tilde{y}_k)$ is needed.

3 Concrete Scenarios

This section motivates the general formulation of Section 2 to illustrate the wide range of settings to which it applies.

Physical sensor beams: The first set of examples corresponds to actually engineering a physical environment with the placement of cheap beam sensors among with nonconvex obstacles. Recall Figure 1. In this case, virtually any model from Section 2 can be realized in practice. In the remainder of this section, we obtain beams *virtually* via other sensing modalities.

Crossings of landmarks: Imagine that a robot moves in a large field, in which several landmarks (e.g., radio towers) are visible using an omni-directional camera. This can be modeled by $W = \mathbb{R}^2$ and $\mathcal{O}$ as a set of point obstacles. Suppose that the landmarks are fully distinguishable and some simple vision software indicates when a pair of landmarks are "on top of each other" in the image. In other words, the robot and two landmarks are collinear, with one of the two landmarks in the middle. The result is mathematically equivalent to placing $n(n-1)$ beams as shown in Figure 2.a, in which rays extend outward along lines passing through each pair of landmarks.

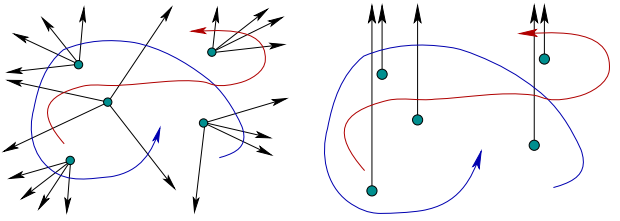


Fig. 2. a) Virtual beams based on pairwise crossings in an image, b) virtual beams based on passing directly north of an obstacle.

Several interesting variations are possible based on precisely what is detected in the image. If the only information is that o_i and o_j crossed each other in the image, then all beams are undirected and the two beams associated with o_i and o_j are indistinguishable. If we know whether o_i passes in front of or behind o_j, then the beams become fully distinguishable. If we know whether o_i passes to the left or right of o_j in the image, then the beams even become directed.

Passing over a windshield mark: Keeping the landmark-based example, consider changing the sensing so that instead of detecting pairwise landmark crossings, the robot simply knows when some landmark is directly south. This could be achieved by using a compass to align the vehicle and noting when a landmark crosses a fixed spot on the image plane or windshield. Figure 2b shows virtual beams that are obtained in this way. Directed and undirected beam models are possible, based on whether the sensor indicates the left-right direction that the landmark moves as it crosses the fixed spot. An important property of this model is that the beams do not intersect (assuming the points are not collinear). Section 5 utilizes this property to reconstruct the path up to homotopy equivalence.

4 Region Filters

In this section, we present a simple method to keep track of the possible regions in which the agent might be after obtaining the sensor word. Suppose that n obstacles $\mathscr{O}$ and m beams $\mathscr{B}$ are given, which leads to the cell decomposition of X, discussed in Section 2. The beams may intersect and may or may not be directed.

Let R_0 denote the set of possible regions that initially contain the agent, before any sensor data is observed. Let R_k denote the set of possible regions after a sensor word $\tilde{y}_k$, of length k, has been obtained. The task is to design a filter that computes R_{k+1}, given R_k and y_{k+1}, which is the most recent observation in $\tilde{y}_{k+1}$.

Let G denote a directed (multi)graph that possibly contains self-loops. Each vertex of G is a region, and a directed edge is made from region r_1 to region r_2 if either: 1) they share an interval of an undirected beam along their boundary, or 2) they share an interval of a directed beam that is directed from r_1 to r_2. The edge is labeled with the beam label. A self-loop in G is made if it is possible to cross a

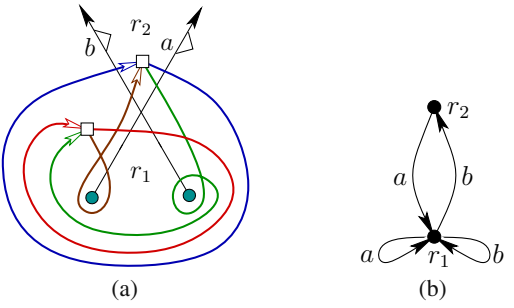


Fig. 3. a) Two intersecting directed beams and four region sequences (and possible paths) that can be inferred from the sensor word ab; b) the corresponding graph G.

beam and remain in the same region, which is illustrated in Figure 3.a; a or b may be crossed while remaining in r_1 the whole time. Also, note that if the initial region is unknown and $\tilde{y} = ab$ is the sensor word, there are four possible interpretations in terms of the possible regions traversed. The corresponding graph G is shown in Figure 3.b.

Simulation as a Nondeterministic Machine

The region filter is implemented on G in a way similar to the simulated operation of a nondeterministic finite automaton. The method keeps track possible states by *marking* the corresponding vertices of G. Initially, mark every vertex in R_0. The filter proceeds inductively. At stage k, the marked vertices are precisely those corresponding R_k. Suppose that y_{k+1} is observed, which extends the sensor word by one observation. For each marked vertex, look for any outgoing edge labeled with y_{k+1}. In each case, the destination vertex is marked. If $y_{k+1} = l^{-1}$ for some $l \in L$ and an ingoing edge is labeled with l, then the edge's source vertex is marked. Any vertex that was marked at stage k but did not get marked in stage $k+1$ becomes *cleared*. Note that the total number of marked vertices may increase because from a single vertex there may be multiple edges that match y_{k+1}. Also, this approach works for the case of partially distinguishable beams because the match is based on the observation y_{k+1}, rather than the particular beam. The set of marked vertices yields R_{k+1}.

Suppose that after computing R_k, we would like to know the possible *sequence* of regions traversed by the agent. The graph G can be used for this computation as well by taking each region in R_k and working backwards using the sensor word to construct possible regions at earlier stages. Note that once R_k is given, the set of possible regions R_i, which was computed at some earlier stage $i < k$, might contain regions that are known at stage k to have been impossible. In other words, information gained at later stages can refine our belief about what might have occurred several stages earlier.

Algorithm complexity

The region filter based on marking vertices in G runs in time $O(|V| + |E|)$ for each update from k to $k+1$, in which $|V|$ and $|E|$ are the numbers of vertices and edges in G, respectively. Note that the number of computations grows with the number of marked vertices, which reflects the amount of uncertainty about the current region. In some cases the number of marked vertices cannot increase, as in the case of ddd-beams. Using exponential space (undesirable), each update can be performed in constant time, similar to the classical NFA to DFA transformation.

5 Reconstruction Up to Homotopy

In this section, the task is to use the sensor word $\tilde{y}$ to reconstruct a description of possible paths $\tilde{x} \in \tilde{X}$ up an equivalence class of homotopic paths. Assume that all paths start and stop at some fixed *basepoint* $x_0 \in X$ which lies in the interior of some region; this assumption is lifted at the end of the section. Paths can be described up to homotopy using the fundamental group $\pi(X)$, which is known to be F_n for a planar region with n holes (caused by the obstacles). Each element of F_n is an equivalence class of homotopic loop paths with endpoints fixed at x_0. The set F_n is called the *free group on n letters*, and its elements can be considered as the set of all finite strings that can be formed using any $l \in L$ and their inverse forms l^{-1}. Since F_n is a group, it also contains an identity element ε. Furthermore, the relations $\varepsilon l = l\varepsilon = l$ and $l\, l^{-1} = \varepsilon$ must be applied to shorten strings by applying cancellations and deleting identity elements. The result is referred to in group theory as a *reduced word*.

An important issue with F_n is its choice of basis. Recall from linear algebra that there are many ways to define and transform bases for a vector space. A similar but more complicated situation exists for F_n. For any fixed ordered basis (called letters above), $a_1, \ldots, a_n$, every other possible ordered basis is given by $g(a_1), \ldots, g(a_n)$. The elements g run over *every* element in the automorphism group of F_n, which is denoted by $Aut(F_n)$. In other words, there is a natural one-to-one correspondence between every ordered basis of F_n and $Aut(F_n)$. This structure is currently under active investigation in pure mathematics. In this paper, a basis of F_n will be chosen in the most straightforward way and other bases will be directly transformed to it. The full structure of $Aut(F_n)$ will thus be avoided.

5.1 Perfect Beams

Let a beam be called *outer* if it is either an infinite ray (possible only if X is unbounded) or it is a finite segment that connects an obstacle to the boundary of W. For a set of n obstacles, let a *perfect* collection of beams mean that there are exactly $m = n$ ddd-beams (recall that this means *disjoint, distinguishable, and directed beams*), with exactly one outer beam attached to each obstacle. For convenience, further assume that all beams in a perfect collection are oriented so that a

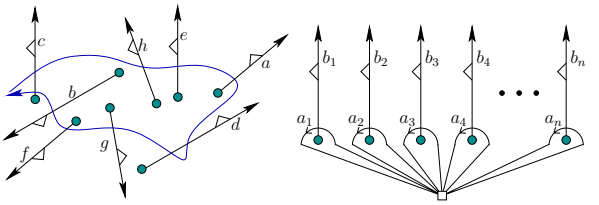


Fig. 4. a) A perfect collection of beams; b) a construction that converts the sensor word $\tilde{y}$ into an element of the fundamental group F_n.

counterclockwise traversal corresponds to the "forward" direction, as shown in Figure 4.a.

Proposition 5.1. *For a perfect collection of beams, the sensor word $\tilde{y}$ maps directly to an element of F_n by simply renaming each letter.*

Proof. While preserving homotopy equivalences, the obstacles and basepoint can be moved into a canonical form as shown in Figure 4.b. This corresponds to choosing a particular basis of F_n in which each generator a_i is exactly a counterclockwise loop around one obstacle. Each b_i corresponds to a beam and letter in L. Using the given sensor word $\tilde{y}$, a word $f(\tilde{y}) \in F_n$ is formed by mapping each a_i in $\tilde{y}$ to b_i in $f(\tilde{y})$. Each beam crossing then corresponds directly to a generator of F_n under the chosen basis. This converts every $\tilde{y}$ into an element of F_n that represents the loop path that was traversed using basepoint x_0. □

Note that reductions of the form $\varepsilon l = l\varepsilon = l$ and $l\, l^{-1} = \varepsilon$ may be performed in $\tilde{y}$ or $f(\tilde{y})$, either before or after the mapping f is applied.

5.2 Sufficient ddd-Beams

What if the collection of beams is not perfect? For some arrangements of obstacles, it might not even be possible to design a perfect collection (unless beams are allowed to be nonlinear). Suppose that a collection $\mathscr{B}$ of $m \geq n$ ddd-beams is given. It is called *sufficient* if all of the resulting regions are simply connected; see Figure 5.a. Note that any sufficient collection must contain at least one outer beam. Also, any perfect collection is also sufficient.

Suppose that a sensor word $\tilde{y}$ is obtained for a sufficient collection $\mathscr{B}$ of beams. The first step in describing the path as an element of F_n is to disregard redundant beams. To achieve this, let $\mathscr{B}' \subseteq \mathscr{B}$ be a minimal subset of $\mathscr{B}$ that is still sufficient. Such a collection is called *minimally sufficient* and can be computed using spanning tree algorithms such as depth-first or breadth-first search. An example is shown in Figure 5.b.

Proposition 5.2. *For a minimally sufficient collection $\mathscr{B}$ of ddd-beams, the sensor word $\tilde{y}$ maps directly to an element of F_n by simply renaming each letter; however, a different basis is obtained for F_n in comparison to Proposition 5.1.*

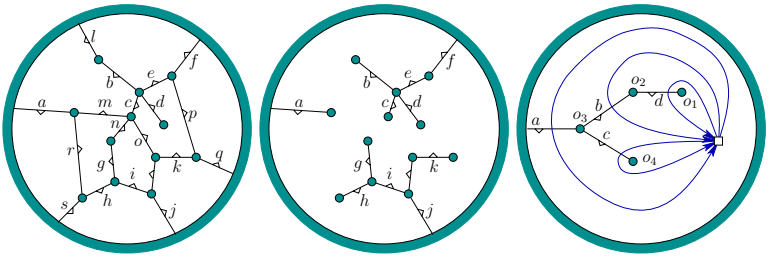


Fig. 5. a) A sufficient collection of ddd-beams; b) a minimally sufficient collection of ddd-beams forms trees that each contain one outer beam. This example is obtained by removing beams from the the figure on the left; c) Forming a basis using a sufficient tree of beams.

Proof: A basis for the fundamental group is defined as follows. For each tree of beams in $\mathscr{B}$, a collection of loop paths can be formed as shown in Figure 5.c. Each loop must cross transversely the interior of exactly one beam and enclose a unique nonempty set of obstacles. Such loops always exist and can be constructed inductively by first enclosing the leaves of the tree and then progressing through parents until a loop is obtained that traverses the outer beam. Since there is only one region, it is possible to inductively construct such a collection of loops for every tree of beams in $\mathscr{B}$. The total collection of loops forms a basis of F_n, which can be related to the basis in Proposition 5.1 via classical Tietze transformations. The mapping f from $\tilde{y}$ to $f(\tilde{y}) \in F_n$ is once again obtained by mapping each letter in $\tilde{y}$ to its corresponding unique loop that traverses the beam. □

Using Proposition 5.2, a simple algorithm is obtained. Suppose that any sufficient collection $\mathscr{B}$ of ddd-beams is given and a sensor word $\tilde{y}$ is obtained. A spanning tree $\mathscr{B}' \subseteq \mathscr{B}$ of beams is computed, which is minimally sufficient. Let $L' \subseteq L$ denote the corresponding set of beam labels. To compute the element of F_n, the first step is to delete from $\tilde{y}$ any letters in $L \setminus L'$. This yields a reduced word $\tilde{y}'$ for which each letter can be mapped directly to a loop using Proposition 5.2 to obtain a representation of the corresponding path in F_n. Once again, reductions based on the identity and inverses in F_n can be performed before or after the mapping is applied.

5.3 The General Case

Consider a collection $\mathscr{B}$ of beams in which some may intersect, some may be undirected, and some may even be indistinguishable. The collection is nevertheless assumed to be *sufficient*, which means that all of the corresponding regions are simply connected. Rather than worry about making a minimal subset of $\mathscr{B}$, the method for the general case works by inventing a collection of *imaginary* beams that happens to be minimally sufficient. Since the ambiguity may be high enough to yield a set of possible paths, the region filter from Section 4 is used.

Before any sensor words are processed, the following preprocessing steps are performed based on W, $\mathscr{O}$, $\mathscr{B}$, L, and α:

1. Compute the arrangement of regions and multigraph G from Section 4.
2. For each vertex in G choose a *sample point* in its corresponding region.
3. For each directed edge e in G, compute a piecewise-linear *sample path* that: i) starts at the sample point of the source vertex of e, ii) ends at the sample point of the destination vertex of e, and iii) crosses the beam associated with e in a manner consistent with its label. The sample path cannot intersect any other beams.
4. Construct any minimally sufficient collection $\mathscr{B}_I$ of *imaginary* ddd-beams. A convenient choice is to make all imaginary beams vertical.
5. For all computed sample paths from Step 3, compute their intersections with the imaginary beams of Step 4 and record the order in which they occur.

Now suppose that a sensor word $\tilde{y}$ is given. The region filter of Section 4 is used to determine the set of possible region sequences. For each region sequence, a path $\tilde{x}'$ is obtained from the corresponding sample points and paths in G. An imaginary sensor word $\tilde{y}'$ is obtained by the sequence of beams in $\mathscr{B}_I$ that are crossed by $\tilde{x}'$. The mapping from sensor words and elements in F_n is provided by the following Corollary to Proposition 5.2.

Corollary 5.1. *For minimally sufficient collection $\mathscr{B}_I$ of imaginary ddd-beams, the imaginary sensor word $\tilde{y}'$ maps directly to an element of F_n, by simply renaming each letter.*

Proof: Use $\tilde{y}' = \tilde{y}$ for the sensor word, and $\mathscr{B} = \mathscr{B}_I$ as the set of beams. □

As usual, reductions can be applied to $\tilde{y}'$ or its image in F_n. Once elements of F_n are computed and reduced for each possible region sequence, duplicates are removed to obtain the complete set of possible homotopically distinct paths based on the sensor word $\tilde{y}$. Refer to Figure 6 for a computed example. Note that $\mathscr{B}_I$ essentially allows the user to define whatever basis of F_n is desired to express the result, however a subtle technical point should be taken into account. Considet the region r that contains the basepoint. The imaginary beams in B_I that intersect r subdivide it into "subregions". If the sample point of r and the basepoint lie in different subregions, then the word obtained corresponds to paths starting and ending at the sample point, not at the basepoint. All of the words obtained will be shifted in the same manner. This is because the words corresponding to the paths starting at the sample point and the basepoint are conjugate, which gives an automorphism of F_n.

Now remove the assumption that only loop paths are executed using a basepoint x_0. If all paths start at some x_0 and terminate at some x_1, then a fixed path segment that connects x_1 back to x_0 can be chosen. This path may intersect some beams, which is false information; however, actual possible paths executed by the agent can at least be compared up to homotopy. If either of the endpoints is not fixed, then all paths become trivially homotopic by continuously shrinking each path to the other basepoint. One possibility is to assume that there are several possible fixed points based on the starting and final regions produced in each sequence from the region filter. In this way, possible paths can at least be compared if their starting and terminating regions match.

6 Path Winding Numbers

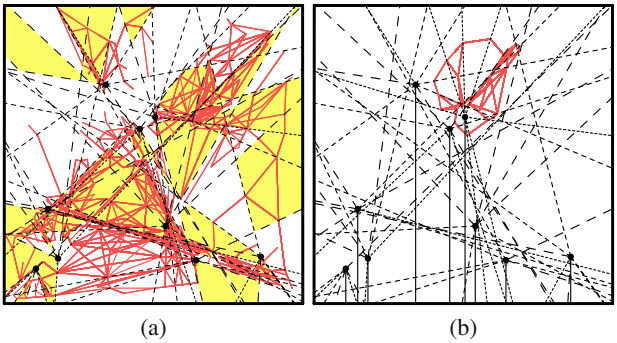
Rather than characterizing paths up to homotopy, this section considers the number of times the path "winds" around each obstacle. Suppose that the agent travels along a loop path. There is an integer *winding number* $v_i \in \mathbb{Z}$ for each $o_i \in \mathcal{O}$, in which m_i is defined as the number of times the path wraps counterclockwise around o_i after deleting all other obstacles and pulling the path tight around o_i using homotopy. If there are n obstacles, then a vector of n winding numbers is obtained. Two paths are called *homologous* if and only if their vector of winding numbers are identical.

6.1 Perfect Beams

In the case of perfect beams, the winding numbers are obtained by directly "abelianizing" the sensor word $\tilde{y}$. Due to Proposition 5.1, the sensor word maps directly to the free group element $f(\tilde{y}) \in F_n$. The winding numbers are then obtained by applying the commutativity relation to $f(\tilde{y})$ (or conveniently, directly to $\tilde{y}$) and sorting the terms. For example, $\tilde{y} = aba^{-1}bbab^{-1}b^{-1}ab^{-1}b^{-1}b^{-1}$ is abelianized to:

$$\begin{aligned} \tilde{y} &= aba^{-1}bab^{-1}b^{-1}ab^{-1}b^{-1}b^{-1} \\ &= aaaa^{-1}bbb^{-1}b^{-1}b^{-1}b^{-1}b^{-1} \\ &= a^2b^{-3}. \end{aligned} \tag{2}$$

The first step sorts the terms, and the second step performs cancellations. This result is expressed as a monomial in which l^k is a sequence of k l's and l^{-k} is a sequence



(a) (b)

Fig. 6. Computed examples, with 337 faces, and 70 beams. In the examples, the three different kinds of dashed line segments represent three different classes of undirected beams. That is, the beams are disjoint, but not directed and only partially distinguishable. The vertical solid segments in (b) are imaginary beams. The black discs are the obstacles, and candidate paths are shown in gray (red). Figure (a) shows the possible candidate paths for a given sensor word, assuming the agent could start in any region. Figure (b) shows the sample paths used to computed the possible homotopy classes for a particular sensor word. Both examples were computed in about two seconds on a commercial PC.

of k l^{-1}'s for each $l \in L$. The winding numbers are simply the exponents; for (2) we obtain $w = (2, -3)$. Note that the winding numbers can computed in time $O(|\tilde{y}|)$ without actually sorting by simply maintaining n counters, one for each letter in $l \in L$. Scan across $\tilde{y}$ and increment or decrement each counter, based on whether l or l^{-1}, is encountered, respectively. Note that this makes a constant-time combinatorial filter, as defined in Section 2, by computing the winding numbers w_{k+1} at step $k+1$ from the winding number vector w_k and the last observation y_{k+1}, which is the last letter of $\tilde{y}_{k+1}$.

6.2 Sufficient ddd-Beams

Once a minimally sufficient collection is determined, the computation for this case then proceeds in the same way as for perfect beams. This yields winding information, but it needs to be transformed to obtain the correct result. Recall the basis from Figure 5.c and suppose that for a path, the abelianized word obtained is $a^5b^{-3}c^4d$. Each exponent may contain information about multiple winding numbers. For example, a^5 implies that the agent wrapped 5 times around o_3, but it also wrapped 5 times around o_1, o_2, and o_4. Likewise, b^{-3} wraps -3 times around o_1 and o_2. A counter is made for each obstacle and each computed exponent raises or lowers some counters. After being performed for each exponent, the result is obtained. For the example $a^5b^{-3}c^4d$ based on Figure 5.c, the winding numbers are $(3, 2, 5, 9)$. Note that if the positive direction of a beam is in the clockwise direction, then the computed winding number needs to be multipled by -1.

6.3 The General Case

Now suppose that a sufficient collection of general beams has been given, which is the model used in Section 5.3. A straightforward approach is to first run the algorithm of Section 5.3. After the sufficient collection of imaginary beams has been placed and the free group elements have been computed, they can be abelianized to obtain the exponents in the method just described. This yields a set of vectors of winding numbers. This approach, however, computes more information than is needed to simply obtain the winding numbers. Suppose that a sufficient collection of beams is given that is not necessarily disjoint, but all beams are directed and distinguishable. Since the winding number essentially ignores all other beams, an approach can be developed by picking a minimally sufficient collection of beams that is not necessarily disjoint. The beam intersections do not interfere with the calculation of winding numbers. For a given sensor word, any letters that do not appear in the minimally sufficient collection can simply be deleted. The method then proceeds as in the case of ddd-beams. In the most general setting of a sufficient collection of beams, the region filter of Section 4 can be applied to yield possible region sequences. For each computed region sequence, the particular beam and its direction crossed can be inferred. Based on this information, the method described for distinguishable, directed beams can be applied.

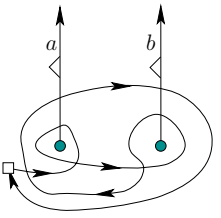


Fig. 7. A simple commutator example that yields sensor word $aba^{-1}b^{-1}$ and winding numbers $(0,0)$, but corresponds to a non-trivial path.

6.4 Higher-Order Winding Numbers

Winding numbers give some crude data concerning free groups and an agent path. These give a measure of how many times an agent circles a given obstacle without counting how the agent weaves in between obstacles. However, winding numbers are insensitive to paths such as a commutator around obstacles, as shown in Figure 7. There are "higher order winding numbers" which keep track of how a agent does in fact interweave through different obstacles. These "higher order winding numbers" in the case above arise in two classical ways reflecting the interplay between geometry and a free group. These are the Lie algebras which arise from either (i) the descending central series of a free group, or (ii) principal congruence subgroups of level p^r in the group of $SL(2,\mathbb{Z})$. These Lie algebras provide measures of complexity in addition to "higher order winding numbers" and it remains an open problem to develop computation methods that characterize them for a given sensor word.

7 Multiple Agents

The formulation in Section 2 can be naturally extended by allowing more than one agent to move in W. In this case, suppose that the sensor beams cannot distinguish between agents. They simply indicate the beam label whenever crossed. For simplicity, assume that agents never cross beams simultaneously. The task is to reconstruct as much information as possible about what path they might have taken. We could proceed as in Sections 4 - 6 and determine region sequences, path homotopy class, and winding numbers. The high level of ambiguity, however, may require further simplifications.

Figure 8.a shows a simple example of this, in which there is one obstacle, two agents, and three undirected beams. Question: If the agents start together in a room, are they together some room after some sensor word was observed? Consider designing the simplest algorithm that answers this question. Figure 8.b shows a surprisingly simple four-state automaton that answers the question for any sensor word. The T state means they are together in some room. Each D_x state means they are in different rooms, with beam x separating them. With only two bits of memory, arbitrarily long sensor words can be digested to produce the answer to the question. Many open questions remain, especially for substantially more complicated

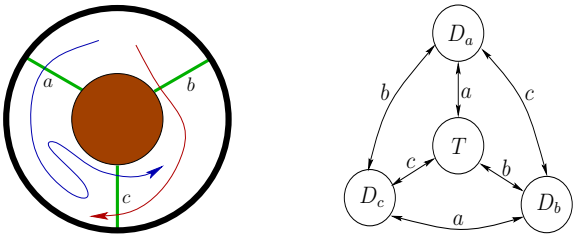


Fig. 8. a) A three-region problem with two agents; b) a tiny automaton (combinatorial filter) that determines whether the agents are together in a room.

environments, such as the one in Figure 2.a with several agents. For which questions can small automata be designed? For a given question, what is the complexity in terms of numbers of agents, obstacles, and beams? What other ways exist for reconstructing and describing and possible paths taken by multiple agents?

8 Conclusions and Open Questions

In this paper we identified a basic inference problem based on agents moving among obstacles and detection beams. Recall from Section 3 that the beams may directly model physical sensors or they may arise virtually from a variety of other sensing models. Therefore, the region filter, homotopic reconstruction, and winding-number computations provide basic information that arises in numerous settings such as robotics, security, forensics, environmental monitoring, and assisted living.

The results presented here represent a first step in understanding this broad class of problems. Many open issues remain for future research, several of which are suggested here: 1) It is assumed that the geometric arrangement of obstacles and beams is known. What happens when this is uncertain? For example, we might not even know which beams intersect. The sensor words can be used to make simultaneous inferences about the agent path and the beam arrangement. 2) Without the assumption of transverse beam crossings and crossings are intersection points, significantly more ambiguity arises. How do these affect the computations? 3) What are the limits of path reconstruction when there are two or more agents? How efficient can filters be made for such problems when there are many obstacles and beams? 4) What other specific path statistics can be computed efficiently from beam data? Can Lie algebra constructions be applied to efficiently compute higher-order winding numbers (based on commutators) for the paths? Can the sensor data be used to compare paths as elements of the braid group? 5) Since the methods so far provide only inference, how can their output be used to design motion plans? In other words, how can the output be used as a filter that provides feedback for controlling how the agents move to achieve some task?

Acknowledgements. The authors are supported in part by the DARPA SToMP program (DSO HR0011-07-1-0002).

References

1. Cabello, S., Liu, Y., Mantler, A., Snoeyink, J.: Testing homotopy for paths in the plane. In: Proceedings of the eighteenth annual symposium on Computational geometry, pp. 160–169. ACM, New York (2002)
2. Dudek, G., Romanik, K., Whitesides, S.: Global localization: Localizing a robot with minimal travel. SIAM Journal on Computing 27(2), 583–604 (1998)
3. Efrat, A., Kobourov, S., Lubiw, A.: Computing homotopic shortest paths efficiently. Computational Geometry 35(3), 162–172 (2006)
4. Goldberg, K.Y.: Orienting polygonal parts without sensors. Algorithmica 10, 201–225 (1993)
5. Guibas, L.J., Motwani, R., Raghavan, P.: The robot localization problem. In: Goldberg, K., Halperin, D., Latombe, J.-C., Wilson, R. (eds.) Algorithmic oundations of Robotics, pp. 269–282. A.K. Peters, Wellesley (1995)
6. Guibas, L.J., Latombe, J.-C., LaValle, S.M., Lin, D., Motwani, R.: Visibility-based pursuitevasion in a polygonal environment. International Journal of Computational Geometry and Applications 9(5), 471–494 (1999)
7. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006), http://planning.cs.uiuc.edu/
8. Lee, J.-H., Shin, S.Y., Chwa, K.-Y.: Visibility-based pursuit-evasions in a polygonal room with a door. In: Proceedings ACM Symposium on Computational Geometry (1999)
9. Magnus, W., Karrass, A., Solitar, D.: Combinatorial Group Theory, 2nd edn., revised edn. Dover (2004)
10. Singh, J., Madhow, U., Kumar, R., Suri, S., Cagley, R.: Tracking multiple targets using binary proximity sensors. In: Proceedings of the 6th international conference on Information processing in sensor networks, pp. 529–538. ACM, New York (2007)
11. Tovar, B., Freda, L., LaValle, S.M.: Mapping and navigation from permutations of landmarks. In: AMS Contemporary Mathematics Proc., vol. 438, pp. 33–45 (2007)
12. Yu, J., LaValle, S.M.: Tracking hidden agents through shadow information spaces. In: Proceedings IEEE International Conference on Robotics and Automation (2008) (under review)

A Motion Planner for Maintaining Landmark Visibility with a Differential Drive Robot

Jean-Bernard Hayet, Claudia Esteves, and Rafael Murrieta-Cid

Abstract. This work studies the interaction of the nonholonomic and visibility constraints of a robot that has to maintain visibility of a static landmark. The robot is a differential drive system and has a sensor with limited field of view. We determine the necessary and sufficient conditions for the existence of a path for our system to be able to maintain landmark visibility in the presence of obstacles. We present a complete motion planner that solves this problem based on a recursive subdivision of a path computed for a holonomic robot with the same visibility constraints.

1 Introduction

Landmarks are of common use in robotics, either to localize the robot with respect to them [17] or to navigate in all kinds of environments [3], being used as goals or subgoals to reach or perceive during the motion. Landmarks can be defined in several manners: From single, characteristic image points with useful properties, up to a 3D object associated with a semantic label and having 3D position accuracy [7]. In all cases, this definition involves at some degree properties of saliency and invariance to viewpoint changes.

To use landmarks in the context of mobile robotics, the first basic requirement is to perceive them during the robot motion. It is to this end that our current research efforts are focused on. Although landmarks have been extensively used, this is to our knowledge *the first attempt* to show whether or not a path of a holonomic robot in the

Jean-Bernard Hayet and Rafael Murrieta-Cid
Centro de Investigación en Matemáticas, CIMAT
Guanajuato, México
e-mail: {jbhayet,murrieta}@cimat.mx

Claudia Esteves
Facultad de Matemáticas de la Universidad de Guanajuato
Guanajuato, México
e-mail: cesteves@cimat.mx

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 333–347.
springerlink.com

presence of obstacles that has to maintain visibility of one landmark with a limited sensor can be transformed into a feasible path for a differential drive robot (DDR). We believe that our research is very pertinent given that a lot of mobile robots are DDRs equipped with limited field of view sensors (e.g., lasers or cameras).

As it is well known in mobile robotics research, nonholonomic systems are characterized by constraint equations involving the time derivatives of the system configuration variables. If the state transition equation is integrable, the corresponding system is said holonomic; otherwise, it is said nonholonomic [10].

From the point of view of motion planning, the main implication of nonholomic constraints is that a collision-free path in the configuration space does not necessarily induce a feasible path for the system. Purely geometric techniques to find collision-free paths do not apply directly here.

1.1 Related Work

Motion planning with nonholonomic constraints has been a very active research field (a nice overview is given in [10]). The most important results in this field have been obtained by addressing the problem with tools from differential geometry and control theory. Laumond pioneered this research and produced the result that a free path for a holonomic robot moving among obstacles in a 2D workspace can always be transformed into a feasible path for a nonholonomic car-like robot by making car maneuvers [11]. Recently, a significant amount of work has been done on the problem of planning collision-free paths for nonholonomic systems, for instance, Isler et al. have used the results of the Dubins car to address pursuit-evasion problems [8].

The study of optimal paths for nonholonomic systems has also been an active research topic. Reeds and Shepp determined the shortest paths for a car-like robot that can move forward and backward [14]. In [16] a complete characterization of the shortest paths for a car-like robot is given. In [1], Balkcom and Mason determined the time-optimal trajectories for a DDR using Pontryagin's Maximum Principle (PMP) and geometric analysis. In [4], PMP is used to obtain the extremal trajectories to minimize the amount of wheel rotation for a DDR. In [13], the authors used the curves proposed by [2] in the context of visual servoing. Here, we use similar curves but the fact that our environments are populated with obstacles makes the problem substantially different.

1.2 Contributions

In this paper, we consider the problem of planning paths for a DDR, whose motion is further constrained by sensing considerations and by obstacles in the environment. These constraints generate both, motion and visibility obstructions. We extend our results from previous works [2]. We provide the necessary and sufficient conditions to compute feasible trajectories for the DDR with *limited sensing capabilities* to maintain landmark visibility in the *presence of obstacles*. Our contributions are:

1. We propose a complete planner to compute collision-free paths for a circular *holonomic* robot maintaining landmark visibility among obstacles.
2. We provide the controls for the execution of the optimal motion primitives.
3. We give the necessary and sufficient conditions for the feasibility of a path for the DDR in the presence of obstacles and with visibility constraints, provided that a collision-free path generated for the *holonomic* system with the same visibility constraints exists.
4. We implement a complete motion planner for the DDR maintaining landmark visibility, based on a recursive subdivision of the holonomic path.

2 Problem Settings and Approach Overview

2.1 The Differential Drive Robot

The DDR is described in Fig. 1. It is controlled through commands to its two wheels, i.e. through angular velocities w_l and w_r. We make the usual assignment of a body-attached $x'y'$ frame to the robot. The origin is at the midpoint between the two wheels, y'-axis parallel to the axle, and the x'-axis pointing forward, parallel to the robot heading. The angle θ is the angle formed by the world x-axis and the robot x'-axis. The robot can move forward and backward. The heading is defined as the direction in which the robot moves, so the heading angle with respect to the robot x-axis is zero (forward move) or π (backward move). The position of the robot w.r.t the origin will be defined either in Cartesian coordinates (x, y) or in polar coordinates $(r, \alpha) : r = \sqrt{x^2 + y^2}$, $\alpha = \arctan \frac{y}{x}$. Figure 1 sums up these conventions.

The robot is equipped with a pan-controllable sensor with limited field of view (e.g., a camera), that can move w.r.t. the robot basis. We will suppose that this sensor is placed on the robot so that the optical center always lies directly above the origin of the robot's local coordinate frame, i.e., the center of rotation of the sensor is the same as the one of the robot. Its pan angle ϕ is the angle from the robot x'-axis to its optical axis. The sensor is limited, both in angle and in range: $\phi \in [\phi_1, \phi_2]$ and the robot visibility region is made by the points p such that the Euclidean distance d from p to the robot satisfy $d_{min} \leq d \leq d_{max}$. Notice that the limitation of the sensor induces virtual obstacles in the configuration space even without physical obstacles. We first assume that the robot moves in the absence of physical obstacles, and then remove this assumption in Section 3.

2.2 Optimal Curves under Visibility Constraints for a DDR

In [2], it has been shown that the shortest distance paths, in the absence of obstacles for a DDR under *angular constraints only* are composed of three motion primitives: straight-line segments, in-site rotations without translation and logarithmic spirals, i.e. curves for which the camera pan angle is saturated. In [2], a characterization of the shortest paths for the system based on a partition of the plane into disjoint regions was also provided. This characterization (called a synthesis) attempted to

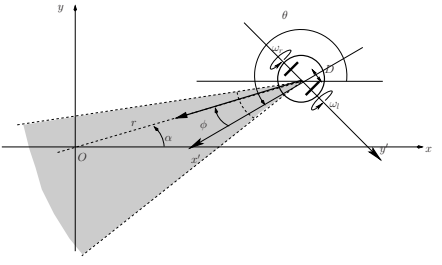


Fig. 1. DDR with visibility constraints. The robot visibility region is depicted in grey (filled region).

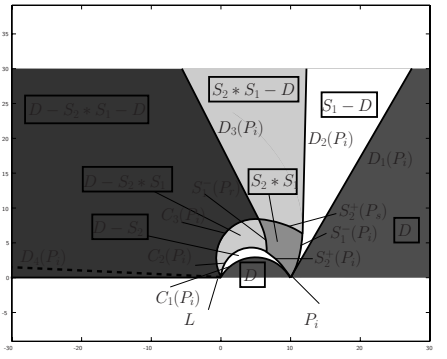


Fig. 2. Distribution of shortest paths, revised from [2] into [6].

obtain the globally optimal paths. Recently in [15], it has been shown that the synthesis presented in [2] was incomplete. Indeed, the work presented in [15] showed a concatenation of motion primitives in which the path is shorter than the one proposed in [2]. Motivated by the work in [15], we have revisited the problem and found the complete partition of the plane (see Fig. 2) and the corresponding globally optimal paths in the absence of obstacles [6]. In that work we showed that the globally optimal paths without obstacles are made of at most seven motion primitives, four (at most) of which produce translation (line-spiral*spiral-line)[1] and three (at most) correspond to in-site rotations. Seven types of trajectories are possible (D, $D-S$, $S-D$, $S-S$, $D-S*S$, $S*S-D$ and $D-S*S-D$). By lack of space, we cannot further develop on this issue in this paper, but the reader is referred to [6] (available on line) for details.

In this paper, we present a complete motion planner for the DDR maintaining landmark visibility in the presence of obstacles, based on a recursive subdivision of the holonomic path. The curves from [2, 6] replace the holonomic path.

2.3 Approach Overview

In [11] it has been shown that a free path for a holonomic robot moving among obstacles in a 2D workspace can always be transformed into a free path for a nonholonomic car-like robot. Three necessary and sufficient conditions guarantee the existence of the path for the car-like robot in the presence of obstacles, provided that a path for a holonomic robot exist.

1. The nonholonomic robot must be Small Time Local Controllable (STLC).
2. The existence of obstacles forces the use of some given metric in the plane to measure the robot clearance. Hence, the topology induced by the robot motion

[1] In the description of trajectories, "*" means a non-differentiable point, and "-" is a smooth transition point.

primitives metric and the one induced by the metric measuring the distance between the obstacles and the robot must be equivalent.
3. There must be $\varepsilon > 0$ clearance between the robot and the obstacles.

Here, we follow the same methodology presented in [11], that we applied to a DDR equipped with a sensor with a limited range and field of view.

The remaining of this work is organized as follows: In Section 3 we present a complete motion planner for a *holonomic* disk, which generates collision-free paths while maintaining landmark visibility. In Section 4 we show the admissible controls to generate our motion primitives with our state transition equation (system model) and we prove the STLC of our system. In Section 5 we use our motion primitives to determine lower and upper bounds of the paths metric and we show that the topology of the robot motion primitives metric and the metric used to measure the distance between the obstacles and the robot are equivalent. Section 6 presents a motion planner for the DDR able to maintain landmark visibility and simulation results. Finally in Section 7 we present the conclusion and future work.

3 Configuration Space Induced by Visibility Constraints

3.1 Configuration Space without Obstacles

As mentioned above, our robot must maintain visibility of a landmark. By visibility we mean that a clear line of sight, lying within the minimal and maximal bounds of the sensor rotation angle and range, can join the landmark and the sensor. The landmark is static and coincident with the origin O of the coordinate system. The visibility constraints imposed by the landmark can be written as

$$\theta = \alpha - \phi + (2k+1)\pi, k \in \mathbb{Z}, \tag{1}$$

$$\phi_1 \leq \phi \leq \phi_2, \tag{2}$$

$$d_{min} \leq r \leq d_{max}. \tag{3}$$

From these equations, we can describe precisely the robot admissible configuration space $\mathscr{C}_{adm}$. The robot can be seen as living in $SE(2)$, as from Eq. 1, ϕ is not really a degree of freedom. Moreover, Eq. 1 adds a constraint on x, y and θ, that can be rewritten

$$\phi_1 \leq -\theta + \arctan(\frac{y}{x}) + (2k+1)\pi \leq \phi_2 \text{ for some } k \in \mathbb{Z}. \tag{4}$$

This means that the visibility constraint both in range and angle can be translated into virtual obstacles in $SE(2)$. From Eq. 4, it is straightforward to deduce the admissible configuration space, which is $SE(2)$ minus these obstacles. Fig. 3 gives a representation of the virtual obstacle (there is actually only one obstacle) in $SE(2)$ for $\phi_2 = -\phi_1 = \frac{\pi}{2}$ (a) and $\phi_2 = -\phi_1 = \frac{\pi}{3}$ (b), as the hollow volume in $SE(2)$. It is worth noting that the free space resulting from this visibility obstacle is made of one single, helical-shaped component of $SE(2)$, which becomes smaller while the

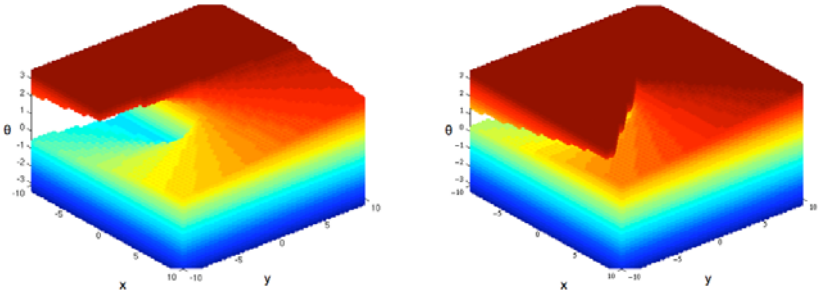


Fig. 3. Admissible configuration space $\mathscr{C}^{\phi}_{adm}$ in the case of a (x,y,θ) configuration space: visibility acts as a virtual obstacle in $SE(2)$. The obstacle is depicted for $(\phi_1,\phi_2)=(-\frac{\pi}{2},\frac{\pi}{2})$ (left) and $(\phi_1,\phi_2)=(-\frac{\pi}{3},\frac{\pi}{3})$ (right) .

authorized pan range is smaller. We call $\mathscr{C}^{\phi}_{adm}$ the admissible configuration space resulting from the angular constraints 2 and 3.

As far as the range constraints of the inequalities 3 are concerned, they introduce two other virtual cylindrical obstacles which reduce the admissible configuration space into $\mathscr{C}^{r}_{adm}$. Finally the combination of these constraints gives rise to the admissible configuration space :

$$\mathscr{C}_{adm} = \mathscr{C}^{\phi}_{adm} \cap \mathscr{C}^{r}_{adm}.$$

A simpler characterization can be made in the (x,y,ϕ) space, instead of the classical (x,y,θ). These two representations are equivalent, since ϕ and θ are related by Equation 1, and so are the constraints equations, but the admissible configuration space, as depicted on Fig. 4, left, is easier to handle, as the constraints over ϕ (inequalities 2) do not depend on x or y. As a result, in that case, $\mathscr{C}^{\phi}_{adm}$ is simply the space between the two planes $\phi=\phi_1$ and $\phi=\phi_2$, and $\mathscr{C}_{adm}$ is the intersection of this volume with $\mathscr{C}^{r}_{adm}$. The advantage of this representation is that it makes easier the task of determining a complete algorithm for the holonomic version of the DDR.

3.2 *Finding a Path for a Holonomic Robot with Visibility Constraints*

Let us suppose that our DDR is disk-shaped. We also suppose we are given a holonomic robot with the same circular shape. The holonomic robot evolves in a plane filled with obstacles and has to respect the visibility constraint.

The free space $\mathscr{C}_{free}$ is defined as the set of configurations inside $\mathscr{C}_{adm}$ which (1) are not in collision with the physical obstacles and (2) are not in the shadow areas created by these same obstacles. We can build it on top of $\mathscr{C}_{adm}$ as depicted on Fig. 4, right, by working in its projection on the xy plane. To begin with, all the physical obstacles, dilated by the circular robot, are subtracted from $\mathscr{C}_{adm}$. We get,

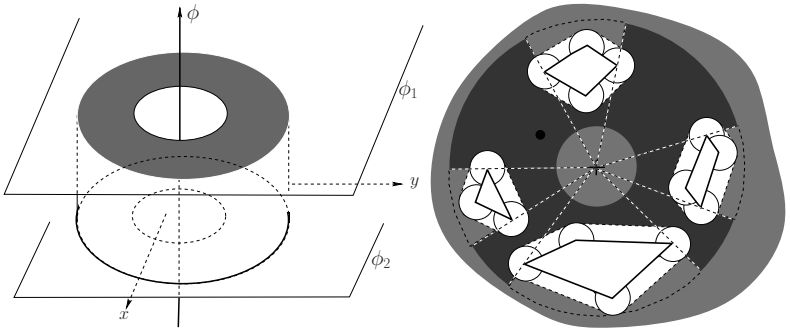


Fig. 4. Left, $\mathscr{C}_{adm}$ for a (x,y,ϕ) configuration space, delimited by two horizontal planes on ϕ and two vertical cylinders. Right, construction of $\mathscr{C}^g_{free}$ for the (x,y,ϕ) representation. By dilating physical obstacles in the xy plane to define collision obstacles (white), the circular robot can be reduced to a point (black). Shadows and visibility constraints define visibility obstacles (light gray).

in white, the *collision obstacles*. In a second step, we remove the obstacles shadows w.r.t. the origin. We get, in light gray, the *visibility (virtual) obstacles*. The resulting projection of $\mathscr{C}_{free}$ is the dark gray area, delimited by arcs of circles and straight line segments. Note that the obstacles, and in particular the visibility ones, do not depend on the values of ϕ, so that $\mathscr{C}_{obst}$ is made of cylinders in $SE(2)$, by translating the projection of Fig. 4, right along the ϕ axis.

Let $\mathscr{C}^g_{free}$ be the domain in the xy plane that generates $\mathscr{C}_{free}$. Several complete algorithms can generate a path for a 2D point in $\mathscr{C}^g_{free}$, e.g. by building a roadmap capturing the domain connectivity [9]. Among them :

- the Generalized Voronoi Graph (GVG) approach. Obstacles here are made of arcs of circles and line segments, hence the GVG is made of arcs of parabola (circle-line), of hyperbola (circle-circle) and line segments (line-line). This is the approach taken in the simulations of section 6.
- the Visibility Graph approach. It consists in generating a graph connecting all mutually visible points among vertices from the obstacles.

Any of these two approaches gives a complete algorithm for finding a path for a 2D point in $\mathscr{C}^g_{free}$ by connecting the desired start and end points to the generated graph [9]. By using this classical result, we can now state

Theorem 3.1. *The problem of planning a path in $\mathscr{C}_{free} \subset SE(2)$ for a holonomic, circular robot with visibility constraints on both range and angular displacement of its sensor is reductible to the problem of finding a path for a single point in $\mathscr{C}^g_{free} \subset \mathbb{R}^2$.*

Proof. Let $P_i = (x_i, y_i, \theta_i)^T$ and $P_f = (x_f, y_f, \theta_f)^T$ be two free initial and final configurations in $SE(2)$. By construction, the 2D points $(x_i, y_i)^T$ and $(x_f, y_f)^T$ belong to $\mathscr{C}^g_{free}$. Now suppose that we can find a path s^g connecting them in $\mathscr{C}^g_{free}$,

$$\begin{aligned} s^g : [0,1] &\rightarrow \mathscr{C}^g_{free} \\ s^g(0) &= (x_i, y_i)^T, \qquad s^g(1) = (x_f, y_f)^T \\ s^g(t) &= (x(t), y(t))^T. \end{aligned}$$

Then, if ϕ_i and ϕ_f are the sensor angle relative to the robot (given by Eq. 1) at initial and final configurations, let us define:

$$\begin{aligned} s_\theta : [0,1] &\rightarrow SO(2) \\ s_\theta(t) &= \arctan(\tfrac{y(t)}{x(t)}) - (1-t)\phi_i - t\phi_f + \pi. \end{aligned}$$

The function s_θ is continuous on $[0,1]$ since s^g is also continuous and $(x(t), y(t)) \neq (0,0)$. Then we can define the following path in $SE(2)$:

$$\begin{aligned} s : [0,1] &\rightarrow SE(2) \\ s(t) &= (x(t), y(t), s_\theta(t))^T. \end{aligned}$$

The path is continuous, it satisfies the initial and final constraints, and, by construction, as for all $t \in [0,1]$, $(1-t)\phi_i + t\phi_f \in [0, 2\pi)$, it also satisfies the visibility constraints at every point.

Conversely, if we are not able to find any free path in $\mathscr{C}^g_{free}$, we cannot have any free path in $SE(2)$: if there were, its projection on $\mathbb{R}^2$ would also be free, which contradicts our initial assumption. As a consequence, an algorithm that solves the planning problem in $\mathscr{C}^g_{free}$ also solves the problem in $\mathscr{C}_{free}$. □

4 System Controls and Small Time Local Controlability

We generate a state transition equation with two controls only. In this scheme, we suppose that the sensor is pointing to the landmark by adjusting its angle value according to equation 1, thus the sensor control is not considered in this motion analysis. It must be determined, whether this system is STLC or not. Far enough from the visibility obstacles, the robot can move forward and backward, hence it is a symmetric system. The construction of the state transition equation is as follows.

First, we get the derivatives $\dot{\phi}$ and $\dot{\alpha}$:

$$\dot{\phi} = \dot{\alpha} - \dot{\theta} \text{ and } \dot{\alpha} = \frac{\dot{y}x - \dot{x}y}{x^2 + y^2}. \tag{5}$$

The linear and angular velocities u_1 and u_2 can be expressed in function of the wheels controls w_l and w_r, as:

$$u_1 = w_r + w_l, \;\; u_2 = w_r - w_l. \tag{6}$$

Therefore, the state variables are:

$$\dot{\theta} = w_r - w_l, \;\; \dot{x} = \cos\theta(w_r + w_l) \text{ and } \dot{y} = \sin\theta(w_r + w_l). \tag{7}$$

A key observation is the following: ϕ is not a degree of freedom. It can be expressed as a function of x, y and θ. Hence, the robot configuration is totally defined by (x, y, θ). ϕ and $\dot{\phi}$ are adjusted so that the system maintain landmark visibility.

The derivative $\dot{\phi}$ can be expressed directly in function of the controls u_1, u_2 and the configuration variables (θ, x, y). This can be done by substituting in 5, the values of $\dot{\alpha}$ and $\dot{\theta}$ from Equations 5 and 7,

$$\dot{\phi} = \frac{(y\cos\theta - x\sin\theta)u_1}{x^2 + y^2} - u_2. \tag{8}$$

Therefore, the state transition equation takes the form:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{9}$$

which is exactly the same of the differential drive robot[11, 12].

We underline that, the *three only motion primitives* are straight lines, rotation in site and logarithmic spirals [2]. The vector field associated to the straight line is $\overrightarrow{X_1} = (\cos\theta, \sin\theta, 0)^T$, the one associated to the rotation in site is simply $\overrightarrow{X_2} = (0, 0, 1)^T$.

Now let us express the vector field associated to the spirals. The equations of these curves are [2]:

$$r = r_0 e^{(\alpha_0 - \alpha)/\tan\phi}, \tag{10}$$

where (r_0, α_0) is one point of the spiral and ϕ remain constant along it. From the previous equation, and by using (1) the equation $\phi = \alpha - \theta + \pi$ and (2) the relation $\alpha = \arctan\frac{y}{x}$, we can easily derive the corresponding vector field, after some algebraic developments

$$\overrightarrow{X_3} = \begin{pmatrix} -\frac{x^2+y^2}{y - x\tan\theta} \\ -\tan\theta\frac{x^2+y^2}{y - x\tan\theta} \\ 1 \end{pmatrix}. \tag{11}$$

This vector field is not defined for $y = x\tan\theta$, which corresponds to zones where the robot has to follow a straight line. In fact, in that case $\overrightarrow{X_3} = \overrightarrow{X_1}$.

A question that naturally arises is: What are the open-loop controls needed for the robot to follow the logarithmic, saturating sensor pan angle? These controls can be derived from the following. When the robot moves drawing sector of logarithmic spirals, the camera pan angle is saturated and hence the landmark is in the limit of the sensor field of view. Hence, the saturated sensor pan angle, and more generally any trajectory maintaining the sensor pan angle constant, satisfy $\dot{\phi} = 0$.

Now, by using Eq.8, we obtain a relation between u_1 and u_2:

$$(y\cos\theta - x\sin\theta)u_1 = (x^2 + y^2)u_2,$$

which can be easily re-written in its polar form

$$u_2 = \frac{1}{r} u_1 \sin(\alpha - \theta). \tag{12}$$

In terms of left and right wheels controls, we deduce from Eq.12, for $r > 0$,

$$\begin{cases} w_r = 1 \\ w_l = \frac{r - \sin(\alpha-\theta)}{r + \sin(\alpha-\theta)}. \end{cases}$$

Again in terms of u_1 and u_2, and by setting $u_1 = 1$,

$$\begin{cases} u_1 = 1 \\ u_2 = \frac{\sin(\alpha-\theta)}{r}. \end{cases}$$

Thus, make our system follow the optimal motion primitives, it is sufficient to consider three admissible pairs of controls (u_1, u_2) that allow to satisfy the visibility constraint and lead the robot to trace these primitives: Straight lines, rotation in site and logarithmic spirals. These controls are respectively

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 \\ \frac{\sin(\alpha-\theta)}{r} \end{pmatrix}. \tag{13}$$

As shown above, the third control produces a logarithmic spiral, and corresponds to a linear combination of the first two vector fields $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$,

$$\overrightarrow{X_3} = a_1 \overrightarrow{X_1} + a_2 \overrightarrow{X_2} \text{ where } a_i \in \mathbb{R}. \tag{14}$$

Hence, the state transition equation presented in 9 can be used to model our system and trace our motion primitives, and therefore our system is STLC by using Chow theorem [10, 5, 12].

The Lie bracket operation computed over vector files $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$ is $(\sin\theta, -\cos\theta, 0)^T$. It is immediate to see that this new vector field is linearly independent from $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$. Hence, this system is small time locally controllable everywhere in the open of the free space.

Note that the first two controls are constant and therefore bounded, and the third one is also bounded since we consider $r > d_{min} > 0$.

5 Analysis of the Metric Induced by Shortest Paths

Let us prove that the metric induced by the total lengths along the shortest paths defined by optimal primitives under visibility constraints is locally equivalent to the Euclidean metric in $\mathbb{R}^2$. Let $(x_i, y_i)^T$ and $(x_f, y_f)^T$ be a pair of initial and final points in the free space. As recorded in 2.2, there are seven kinds of shortest paths: line segments (on which the length is obviously equal to the Euclidean distance in $\mathbb{R}^2$), and four concatenations of one or two line segments with one or two logarithmic

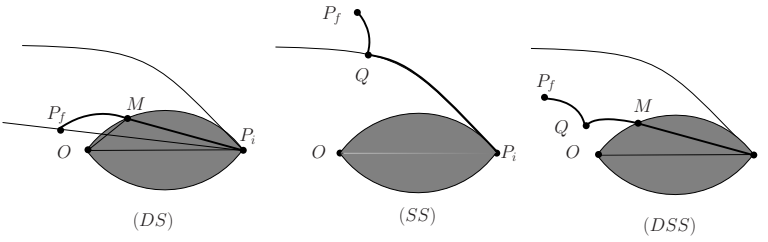


Fig. 5. Some of the shortest paths: concatenation of a line and a ϕ_2 spiral (DS, left), concatenation of a ϕ_2 and a ϕ_1 spiral (SS, center), and concatenation of line and two spirals (DSS, right).

spiral at saturated ϕ. Examples of S*S curves, S-D curves, D-S*S curves, and D-S*S-D curves are depicted in Fig. 5. An important consideration is that, as it is shown in [6], if the families of $3-$ and $4-$letter trajectories (e.g. D-S*S) give the optimal path for some configuration (P_i, P_f) then they are either shorter than the $D-S$ (or $S-D$) path, or shorter than the $S-S$ path (which are just instantiations of these families). As a consequence, we can simply focus on the lengths d_{DS} and d_{SS}.

First note that the length of an arc on a logarithmic spiral keeping ϕ constant, starting from a point $P_0 = (r_0, \alpha_0)^T$ and reaching a point $P_1 = (r_1, \alpha_1)^T$ is

$$l_\phi(P_0, P_1) = \frac{r_0}{\cos\phi} |1 - e^{\frac{\alpha_0 - \alpha_1}{\tan\phi}}|. \tag{15}$$

Also note that since $r \geq d_{min} > 0$, whenever P_1 is close enough to P_0,

$$l_\phi(P_0, P_1) \leq \frac{3r_0}{2|\sin\phi|}|\alpha_0 - \alpha_1| \leq \frac{9r_0}{4|\sin\phi|}|\sin(\alpha_0 - \alpha_1)| \leq \frac{9\|P_0P_1\|}{4|\sin\phi|}. \tag{16}$$

Case of a line segment and a spiral (DS). In that case (Fig. 5, left), the distance between P_i and P_f is given by

$$d_{DS}(P_i, P_f) = \|P_iM\| + l_{\phi_1}(M, P_f).$$

Now note that if P_f is close enough to P_i, by using the bound 16, we get

$$d_{DS}(P_i, P_f) \leq \|P_iM\| + \frac{9}{4|\sin\phi_2|}\|MP_f\| \leq (1 + \frac{9}{4|\sin\phi_2|})(\|P_iM\| + \|MP_f\|).$$

Now an analysis of points O, P_i, P_f and M shows that $\frac{\pi}{2} < \pi - \phi_2 < \angle P_fMP_i < \pi$ (with $\phi_2 > 0$) so that $-1 < \cos(P_fMP_i) < -\cos\phi_2$. From the cosine rule in the triangle P_iMP_f, we derive

$$\|P_iM\|^2 + \|MP_f\|^2 + 2\cos\phi_2\|P_iM\|\|MP_f\| \leq \|P_iP_f\|^2,$$

which induces $\|P_iM\| + \|MP_f\| \leq \frac{1}{\sqrt{\cos\phi_2}}\|P_iP_f\|$. Combining this result with the equation above, we have a relation of local equivalence between distances, for $\phi_2 < \frac{\pi}{2}$,

$$\|P_iP_f\| \leq d_{DS}(P_i,P_f) \leq \frac{1}{\sqrt{\cos\phi_2}}(1+\frac{9}{4|\sin\phi_2|})\|P_iP_f\|. \tag{17}$$

Case of two spirals (SS). By using the Equation 15 twice on the two spirals (Fig. 5, center), and with $t_1 = \tan\phi_1$, $t_2 = \tan\phi_2$,

$$\begin{aligned} d_{SS}(P_i,P_f) &= l_{\phi_2}(P_i,Q) + l_{\phi_1}(Q,P_f) \\ &= \frac{r_{P_i}}{\cos\phi_2}(1-e^{\frac{\alpha_{P_i}-\alpha_Q}{t_2}}) + \frac{r_{P_f}}{\cos\phi_1}(1-e^{\frac{\alpha_{P_f}-\alpha_Q}{t_1}}). \end{aligned}$$

The intersection point Q between the spirals can be easily shown to be

$$\alpha_Q = \frac{t_1t_2}{t_1-t_2}\log\frac{r_{P_i}}{r_{P_f}} + \frac{t_1}{t_1-t_2}\alpha_{P_i} - \frac{t_2}{t_1-t_2}\alpha_{P_f},$$

which can be plugged into the previous equation to give, after simplifications,

$$d_{SS}(P_i,P_f) = a_2r_{P_i} + a_1r_{P_f} - (a_1+a_2)e^{\frac{\alpha_{P_f}-\alpha_{P_i}}{t_1-t_2}} r_{P_i}^{\gamma} r_{P_f}^{1-\gamma},$$

where $a_l = \frac{1}{\cos\phi_l}$ for $l = 1,2$ and $\gamma = \frac{-t_2}{t_1-t_2}$.

Whenever P_f is sufficiently close to P_i (which we will suppose on the x-axis), by using Taylor expansion around P_i,

$$d_{SS}(P_i,P_f) \approx K_x|x_{P_i} - x_{P_f}| + K_y|y_{P_i} - y_{P_f}|,$$

where $K_x = \frac{\sin\phi_2+\sin\phi_1}{\sin(\phi_2+\phi_1)} > 0$ and $K_y = \frac{a_1+a_2}{t_2-t_1}$. It is then straightforward to get, for some other positive constant K'

$$\|P_iP_f\| \leq d_{SS}(P_i,P_f) \leq K'\|P_iP_f\|. \tag{18}$$

Note that in both cases involving spirals, the condition $r \geq d_{min} > 0$ is important to get a neighborhood size that is independent of point P_i. As a consequence, we can state that for any neighborhood of a point P_i in $\mathscr{C}^g_{free}$, there is a smaller neighborhood around P_i such that all the points in this neighborhood can be attained from P_i by the shortest paths of the DDR under visibility constraints that we get from the synthesis of [6]. We can now state the following theorem:

Theorem 5.1. *If a collision-free path for a holonomic robot that maintains visibility of a landmark exists, then, a feasible collision-free path for a DDR with the same visibility constraints also exists, provided that it moves only along the paths composed with the three motion primitives.*

Proof. This theorem is proven given the three properties already shown: (1) The system is STLC, then it can locally maneuver in a neighborhood of the open space. (2) Our motion primitives can be executed with a bounded control and (3) The metric of the primitives induce the same topology and are locally equivalent to the euclidean metric in $\mathbb{R}^2$, from this it follows that the holonomic paths can be always divided and replaced by paths composed of the three motion primitives. □

6 Motion Planner and Simulations

Here, the results from the previous section are used to propose a complete planner for a circular-shaped DDR navigating among obstacles and having to maintain a landmark in sight, whereas its sensor is under range and angular constraints. Inspired from the classical roadmap-based approach, we implemented a simple planning algorithm according to the following steps :

1. Build $\mathscr{C}^g_{obst} = \overline{\mathscr{C}^g_{free}}$ by taking the union of the dilated obstacles with the shadows induced by the landmark visibility;
2. Build the GVG on $\mathscr{C}^g_{free}$: as $\mathscr{C}^g_{free}$ is made of line segments and arcs of circle, the resulting Voronoi Diagram is made of line segments and arcs of parabola or hyperbola; the graph edges weights are a combination of the edge lengths and of the minimal clearance along this edge, so as to find a compromise between shortest and clearest paths;
3. Given a starting and a goal configurations, compute a path $\hat{s}$ for the holonomic system associated to the robot by connecting these locations to the GVG; if not possible, no non-holonomic path can be found as well;

Fig. 6. (Left) Shortest paths for a DDR under visibility constraints alone (angular and distance ranges, represented by the inner and outer circles). The robot shape, heading, and gaze are drawn here, and then omitted for the sake of readability. The Voronoi-based path is in dark, so is the final computed path. (Center) Obstacles (dark) and regions visible from the landmark (light gray). (Right) Construction of $\mathscr{C}^g_{free}$: dilated obstacles (dashed grey) are removed from the visible area and underlying GVG.

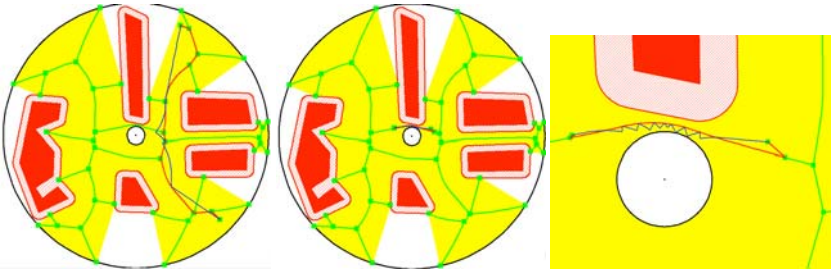


Fig. 7. (Left) Examples of a path computed by the recursive algorithm. (Center) and (Right) Behavior of the planner in narrow passages: as expected, a solution may imply a quantity of maneuvers to finally reach the goal. (Right) is a zoomed view of (Center).

4. Recursively try to connect the starting and ending points with the optimal primitives of section 2.2; whenever the sub-paths induced by these primitives are in collision, use as a sub-goal the point at middle-path in $\hat{s}$ and apply the recursive procedure to the two resulting pairs of points.

Figures 6 (center) and (right) illustrate the first two steps of the algorithm (up to the construction of the GVG), whereas in (left), in the free space, a curve composed of two spirals and in-site rotations is shown. Figure 7 (left) shows an example of path planning among obstacles, all made of concatenations of line segments, in-site rotations and logarithmic spirals. Finally, Figures 7 (center) and (right) illustrate the behavior of the algorithm in narrow passages, where a large number of maneuvers may have to be done to connect the starting and ending points.

Based on the metrics equivalence, we can ensure the convergence of the recursive algorithm, i.e., whenever a holonomic path exists, we obtain a path for the non-holonomic robot (DDR) after a *finite* number of iterations.

7 Conclusions and Future Work

In this work, we have proposed a complete motion planner to compute collision-free paths for a holonomic disk robot able to maintain landmark visibility in the presence of obstacles. We have shown that if a path exist for the holonomic robot then a feasible (collision free and maintaining landmark visibility) path composed by our three motion primitives shall always exist for the DDR. We have also provided the motion controls to execute these motion primitives. Finally, we have implemented a motion planner for the DDR based on a recursive sub-division of the holonomic path. In our planner the motion primitives replace the sections of the holonomic path. As future work, we would like to study the problem of determining a path as sequence of sub-goals defined by several landmarks. In the scheme at least one landmark should be visible at every element of the motion sequence.

References

1. Balkcom, D., Mason, M.: Time optimal trajectories for bounded velocity differential drive vehicles. Int. J. of Robotics Research 21(3), 199–217 (2002)
2. Bhattacharya, S., Murrieta-Cid, R., Hutchinson, S.: Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints. IEEE Trans. on Robotics 23(1), 47–59 (2007)
3. Briggs, A., Detweiler, C., Scharstein, D., Vandenberg-Rodes, A.: Expected shortest paths for landmark-based robot navigation. Int. J. of Robotics Research 8(12) (2004)
4. Chitsaz, H., LaValle, S., Balkcom, D., Mason, M.: Minimum wheel-rotation paths for differential-drive robots. In: IEEE Int. Conf. on Robotics and Automation (2006)
5. Choset, H., Lynch, K., Hutchinson, S., Cantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston (2005)
6. Hayet, J.B., Esteves, C., Murrieta-Cid, R.: Shortest paths for differential drive robots under visibility and sensor constraints. Technical Report I-09-02/24-02-2009, CIMAT (Submitted to IEEE-TRO, 2009), http://www.cimat.mx/~jbhayet/PUBLIS/Hayet-TR2009.pdf
7. Hayet, J.B., Lerasle, F., Devy, M.: A visual landmark framework for mobile robot navigation. Image and Vision Computing 8(25), 1341–1351 (2007)
8. Isler, V., Sun, D., Sastry, S.: Roadmap based pursuit-evasion and collision avoidance. Robot-Sci. Syst., 257–264 (2005)
9. Latombe, J.-C.: Robot motion planning. Kluwer, Dordrecht (1991)
10. Laumond, J.-P.: Robot motion planning and control. Springer, Heidelberg (1998)
11. Laumond, J.-P., Jacobs, P.E., Taïx, M., Murray, R.M.: A motion planner for nonholonomic mobile robots. IEEE Trans. on Robotics and Automation 10(5), 577–593 (1994)
12. LaValle, S.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
13. López-Nicolás, G., Bhattacharya, S., Guerrero, J., Sagüés, C., Hutchinson, S.: Switched homography-based visual control of differential drive vehicles with field-of-view constraints. In: Proc. of the IEEE Int. Conference on Robotics and Automation, pp. 4238–4244 (2007)
14. Reeds, J.A., Shepp, L.A.: Optimal paths for a car that goes both forwards and backwards. Pacific J. of Mathematics 145(2), 367–393 (1990)
15. Salaris, P., Belo, F., Fontanelli, D., Greco, L., Bicchi, A.: Optimal paths in a constrained image plane for purely image-based parking. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1673–1680 (2008)
16. Souères, P., Laumond, J.-P.: Shortest paths synthesis for a car-like robot. IEEE Trans. on Automatic Control 41(5), 672–688 (1996)
17. Thrun, S.: Bayesian landmark learning for mobile robot localization. Machine Learning 33(1) (1998)

Part VII
Geometric Computations

On Approximate Geodesic-Distance Queries amid Deforming Point Clouds*

Pankaj K. Agarwal, Alon Efrat, R. Sharathkumar, and Hai Yu

Abstract. We propose data structures for answering a geodesic-distance query between two query points in a two-dimensional or three-dimensional dynamic environment, in which obstacles are deforming continuously. Each obstacle in the environment is modeled as the convex hull of a continuously deforming point cloud. The key to our approach is to avoid maintaining the convex hull of each point cloud explicitly but still able to retain sufficient geometric information to estimate geodesic distances in the free space.

1 Introduction

The geodesic-distance query problem in an obstacle-present environment arises in motion planning, assembly planning, gaming industry, computational geometry, and related fields. The goal is to construct a suitable data structure based on the geometry of the environment, so that the length of a shortest obstacle-avoiding path between two query points can be reported quickly. This was the focal of motion planning.

Pankaj K. Agarwal and R. Sharathkumar
Department of Computer Science, Duke University
e-mail: {pankaj,sharath}@cs.duke.edu

Alon Efrat
Department of Computer Science, The University of Arizona
e-mail: alon@cs.arizona.edu

Hai Yu
Google Inc. New York, NY
e-mail: fishhai@google.com

* Work on this paper is supported by NSF under grants CNS-05-40347, CFF-06-35000, and DEB-04-25465, by ARO grants W911NF-04-1-0278 and W911NF-07-1-0376, by an NIH grant 1P50-GM-08183-01, by a DOE grant OEG-P200A070505, and by a grant from the U.S.–Israel Binational Science Foundation. Part of the work was done while the last author was at Duke University.

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 351–365.
springerlink.com

However, most of the existing work has focused on answering geodesic distance queries in a static environment or when each obstacle moves as a rigid body. There are several reasons to study geodesic-distance query problems when the queries are time dependent and obstacles are deforming continuously:

- With the availability of sensing and tracking technology, it is possible to monitor many geo-temporal phenomena in real time. Wild fires, areas contaminated by hazardous gas, regions under surveillance of enemy forces (in military applications), and regions with bad atmospheric conditions (in flights scheduling applications) can all be modeled as obstacles deforming in time, where one wishes to find a short path avoiding all the obstacles within some time frame.
- In an environment with a large numbers of moving obstacles one might wish to hierarchically cluster the obstacles and represent each cluster by its convex hull. Answering motion-planning queries can be done by either avoiding all obstacles within a cluster, if possible, or solving the problem with the convex hulls of the children sub-clusters. Obstacles might change their trajectory, start or stop being vertices of the convex hulls, and move from cluster to a sibling cluster, according to their location.
- The *asteroid avoidance problem* is to plan the path of a robot from source to destination while avoiding moving obstacles and not exceeding a given velocity. This important problem is known to be PSPACE-hard [17]. A natural heuristic to the problem is to dividing the free space and time domain into fine enough cells, so that within each region obstacles can be considered static with respect to the velocity of the robot, and the (portion of the) shortest path is constrained to the cell. One could use our algorithm as a tool to answer many geodesic distance queries in different time intervals or to recompute shortest paths after refining the time intervals.

Motivated by these applications, we focus on a model in which each obstacle is represented as the convex hull of a dynamic point cloud, e.g., it may correspond to a squad of enemy troops in motion, a scatter of spreading wild fire, or a cluster of asteroids. In the following, we introduce the model of motion to be followed throughout the paper and define our problem formally.

Model of motion. We use the *kinetic data structure* (KDS for short) framework proposed by Basch *et al.* [6] to handle dynamics of the environment. Let τ denote the time parameter. A moving point $p(\tau)$ in $\mathbb{R}^d$, for $d = 2, 3$, is a function $p : \mathbb{R} \to \mathbb{R}^d$. We call $p(\tau)$ *algebraic* if each individual coordinate of $p(\tau)$ (a real-valued function) is an algebraic function of τ. We use point clouds to model a deforming convex polytope $P(\tau)$ in $\mathbb{R}^d$. In this *point-cloud model*, let $S(\tau) = \{p_1(\tau), \cdots, p_n(\tau)\}$ be a finite set of moving points in $\mathbb{R}^d$, and a deforming convex polytope $P(\tau)$ is defined to be $\mathrm{conv}S(\tau)$ — the convex hull of $S(\tau)$. We emphasize that points in $S(\tau)$ are allowed to change their trajectories at any moment if needed.

In the KDS framework, we maintain a data structure on the fly, accompanied by a set of geometric predicates, called *certificates*, to serve as a proof of correctness for the maintained data structure. Since the current motion of objects are known, the KDS is able to predict the time (called *failure time*) at which each certificate

becomes invalid. All the failure times are scheduled in a global queue called *event queue*. The KDS does nothing until the time reaches the first failure time in the event queue. At that moment, an *event* occurs, and the KDS processes this event by updating the data structure to restore its correctness, as well as updating the certificates and the event queue accordingly. The KDS then moves on towards the next event.

As mentioned above, a change of trajectory or speed for an object is allowed, and it is assumed that the KDS is notified about the change when it occurs. A KDS is called *local* if each moving object is involved in a small number of certificates. A local KDS is able to quickly respond to the motion change of an object, by recomputing the failure times of all the certificates involving the object. All our KDS's are local and therefore accommodate motion changes efficiently.

Problem statement. Let $\mathcal{F}$ be a path-connected closed subset in $\mathbb{R}^d$, for $d = 2, 3$. For two points $s, t \in \mathcal{F}$, a *geodesic path* between s and t within $\mathcal{F}$, denoted by $\Pi_{\mathcal{F}}(s,t)$, is a path from s to t completely lying inside $\mathcal{F}$ and with minimum length. The *geodesic distance* between s and t, denoted by $d_{\mathcal{F}}(s,t)$, is the length of $\Pi_{\mathcal{F}}(s,t)$. In this paper, the subset $\mathcal{F}$ in question is the *free space* of a set $\mathbb{P}$ of k pairwise disjoint convex polytopes in $\mathbb{R}^d$, that is, $\mathcal{F} = \mathbb{R}^d \setminus \text{int} \bigcup_{P \in \mathbb{P}} P$. A path (line segment, point) is *free* (with respect to $\mathbb{P}$) if it lies in $\mathcal{F}$. So $\Pi_{\mathcal{F}}(s,t)$ is the shortest free path between s and t.

Let $\mathbb{P}(\tau)$ be a collection of pairwise disjoint deforming convex polytopes in $\mathbb{R}^2$ or $\mathbb{R}^3$, each defined by the point-cloud model, and let $\mathcal{F}(\tau)$ be the free space of $\mathbb{P}(\tau)$. Our problem is to maintain a certain data structure as τ varies, so that at any τ, the geodesic distance between any two query points $s, t \in \mathcal{F}(\tau)$ can be reported.

Many existing data structures for geodesic-distance queries in static environments (to be reviewed shortly) are quite complicated and unlikely to render efficient kinetic data structures for dynamic environments. Given this situation and taking into account practical considerations, our goal in this paper is to design a kinetic data structure that meets the following criteria:

(a) The number of events of the KDS is small, ideally nearly $O(n)$ in $\mathbb{R}^2$ and nearly $O(n^2)$ in $\mathbb{R}^3$. In particular, one cannot maintain each polytope in $\mathbb{P}(\tau)$ explicitly as the point cloud deforms because the number of events for maintaining $\mathbb{P}(\tau)$ alone would be $\Omega(n^2)$ in $\mathbb{R}^2$ [2] and $\Omega(n^3)$ in $\mathbb{R}^3$ in the worst case.
(b) The KDS provides a flexible tradeoff between the query time and the number of events.
(c) The KDS handles motion changes and transient obstacles (i.e., obstacles in $\mathbb{P}(\tau)$ may be added or deleted) efficiently.

As a compromise, we allow that the data structure only reports a (reasonable) approximation of the geodesic distance, and that the query time can depend on the number of obstacles (but not their total complexity).

Related work. Geodesic-distance queries in a static environment have been extensively studied. Chiang and Mitchell [8] proposed a polynomial sized data structure that answers geodesic-distance query amid polygonal obstacles in $\mathbb{R}^2$ in $O(\log n)$ time; n is the total number of obstacle vertices. They also proposed tradeoffs

between space and query time. Chen [7] observed that an algorithm of Clarkson [9] can be turned into a data structure of size $O(n^2)$ to support $(1+\varepsilon)$-approximate geodesic-distance queries in $O(\varepsilon^{-1}\log n)$ time, for a fixed parameter ε. He also designed a data structure of size $O(n\log n)$ to answer $(6+\varepsilon)$-approximate geodesic-distance queries in $O(\log n)$ time (see also [5]). Very few results are known on geodesic-distance queries in $\mathbb{R}^3$. Agarwal *et al.* [1] designed a data structure of size $O(n^6 m^{1+\delta})$, for $1 \le m \le n^2$ and for any $\delta > 0$, to store a convex polytope P with n vertices in $\mathbb{R}^3$ so that the geodesic-distance query for any two points on P can be answered in $O((\sqrt{n}/m^{1/4})\log n)$ time. Har-Peled [11] considered the same problem but allowed $(1+\varepsilon)$-approximations for $\varepsilon > 0$, and showed that a $(1+\varepsilon)$-approximation of the geodesic distance between two query points can be reported in $O(\varepsilon^{-3/2}\log n + \varepsilon^{-3})$ time. Recently, Agarwal *et al.* [3] have developed data structure for answering geodesic-distance queries in $\mathbb{R}^3$ from a fixed source point.

We are not aware of any existing work on geodesic-distance queries in the dynamic environment. For an account of extensive work on collision detection and motion planning in dynamic environments, we refer the readers to [14, 18] and the references therein.

Our results. We present simple kinetic data structures for answering approximate geodesic-distance queries amidst a collection $\mathbb{P}$ of k pairwise disjoint deforming convex polytopes in $\mathbb{R}^2$ and $\mathbb{R}^3$. Each polytope $P \in \mathbb{P}$ is defined as the convex hull of a set of moving points, each moving along a bounded-degree algebraic trajectory. Let n be the total number of such points.

In $\mathbb{R}^2$, for a prescribed parameter $\varepsilon > 0$, our kinetic data structure uses $O(n/\sqrt{\varepsilon})$ space and processes $O(\lambda_c(n)/\sqrt{\varepsilon})$ events in total, where $\lambda_c(n)$ is the maximum length of Davenport-Schinzel sequences of order c on n symbols and is nearly linear (throughout the paper, c denotes a constant integer related to the degree of the motion). Processing each event takes $O(\log^2 n)$ time. It can be used to report, in $O((k/\sqrt{\varepsilon})\log(k/\varepsilon))$ time, a $(1+\varepsilon)$-approximation to the geodesic distance between two arbitrary query points s,t in the free space. In $\mathbb{R}^3$, for a prescribed parameter $1 \le m \le n$, our data structure uses $O(n)$ space and processes $O(n\lambda_c(n/m))$ events in total, each of which can be handled in $O(\log^2 n)$ time. At any moment, given two query points s,t in the free space, the data structure is able to return, in $O(mk\log(n/m))$ time, an $O(k_{st})$-approximation to the geodesic distance between s and t, where k_{st} is the number of polytopes intersected by the line segment st and is expected to be small in practice. Our data structures are simple enough to allow for points defining the polytopes to be inserted or deleted or change their motion in polylogarithmic time per event.

2 Geodesic Distance Queries in $\mathbb{R}^2$

For notational convenience, we will omit the time parameter τ when no confusion arises. Let $\mathbb{P} = \{P_1, \cdots, P_k\}$ be a collection of k pairwise disjoint deforming convex polygons, where each P_i is defined as the convex hull of a set S_i of n_i moving points. Instead of explicitly maintaining each P_i, we maintain a certain "sketch" $\widetilde{P}_i$ of each

P_i, so that the geodesic distance between any two points s,t in the free space $\mathcal{F}$ is approximately preserved. More formally, a set $\widetilde{\mathbb{P}} = \{\widetilde{P}_1, \cdots, \widetilde{P}_k\}$ of convex polygons is called an *ε-sketch* of $\mathbb{P}$ if

(i) $\widetilde{P}_i \subseteq P_i$, for each $1 \leq i \leq k$;
(ii) for any two points $s,t \in \mathcal{F}$, $d_{\mathcal{F}}(s,t) \leq (1+\varepsilon) d_{\widetilde{\mathcal{F}}}(s,t)$, where $\mathcal{F} \subseteq \widetilde{\mathcal{F}}$ is the free space of $\widetilde{\mathbb{P}}$.

Since $\mathcal{F} \subseteq \widetilde{\mathcal{F}}$, $d_{\widetilde{\mathcal{F}}}(s,t)$ is well-defined for any pair $s,t \in \mathcal{F}$ and $d_{\mathcal{F}}(s,t) \geq d_{\widetilde{\mathcal{F}}}(s,t)$.

We construct an ε-sketch of $\mathbb{P}$ as follows. Set $r = \lceil 2\pi/\sqrt{2\varepsilon} \rceil$. For $0 \leq j < r$, let $u_j = (\cos(j\sqrt{2\varepsilon}), \sin(j\sqrt{2\varepsilon}))$. The set $\mathcal{N} = \{u_j \mid 0 \leq j < r\}$ forms a uniform set of directions in $\mathbb{R}^2$. For $1 \leq i \leq k$ and $0 \leq j < r$, let $p_i^j = \arg\max_{p \in S_i} \langle u_j, p \rangle$ denote the extremal point of S_i in the direction $u_j \in \mathcal{N}$. Let $\widetilde{S}_i = \{p_i^j \mid 0 \leq j < r\}$, $\widetilde{P}_i$ is the convex hull of $\widetilde{S}_i$ and $\widetilde{\mathbb{P}} = \{\widetilde{P}_1, \cdots, \widetilde{P}_k\}$.

Maintaining $\widetilde{\mathbb{P}}$ as $\mathbb{P}$ deforms over time is straightforward. For each S_i and each $u_j \in \mathcal{N}$, we use a kinetic tournament[1] to keep track of the extreme point p_i^j. Note that $p_i^0, p_i^1, \cdots, p_i^{r-1}$ are in convex position and naturally represent $\widetilde{P}_i$ in this order. (Although P_i deforms continuously, $\widetilde{P}_i$ may change discontinuously at certain time instances; a change of extremal point in some direction u_j may result in a discontinuous change in the shape of $\widetilde{P}_i$.) When a point p is inserted or deleted or changes its trajectory, we update $O(1/\sqrt{\varepsilon})$ kinetic tournaments involving that point. A point cloud can also be added to or removed from the environment in a straightforward manner.

Lemma 2.1. *$\widetilde{\mathbb{P}}$ is an ε-sketch of $\mathbb{P}$.*

Proof. Clearly, for $1 \leq i \leq k$, $\widetilde{S}_i \subseteq S_i$ and as such $\widetilde{P}_i \subseteq P_i$. Hence $\widetilde{\mathbb{P}}$ satisfies (i). We next prove $\widetilde{\mathbb{P}}$ also satisfies (ii).

Let s,t be two points in $\mathcal{F}$. Consider a geodesic path $\Pi = \Pi_{\widetilde{\mathcal{F}}}(s,t,)$. If $\Pi \subseteq \mathcal{F}$, then Π is also $\Pi_{\mathcal{F}}(s,t,)$ and there is nothing more to prove. So from now on we assume that Π intersects $\widetilde{\mathcal{F}} \setminus \mathcal{F}$.

By our construction, $\widetilde{\mathcal{F}} \setminus \mathcal{F}$ consists of a set of convex polygons whose interiors are pairwise disjoint (see Figure 1). For each such polygon O, one of its edges called the *base* of O is an edge of $\widetilde{P}_i$ for some i, and the other edges called the *dome* of O come from a subsequence of edges of P_i. Let $p_i^j p_i^{j+1}$ be the base of O. Since p_i^j (resp., p_i^{j+1}) is an extreme point of P_i in direction u_j (resp., u_{j+1}), any point on the dome of O is extreme only in some direction between u_j and u_{j+1} on $\mathbb{S}^1$. In other words, the outward unit normal of a point on the dome O lies in the interval $[u_j, u_{j+1}]$.

[1] A kinetic tournament [6] can be used to maintain the maximum (or minimum) of a set S of n moving points on the real line. The total number of events is $O(\lambda_c(n))$, each of which can be processed in $O(\log^2 n)$ time. Every point in S participates in $O(\log n)$ certificates, and hence a motion update can be performed in $O(\log^2 n)$ time. Similarly, a moving point can be inserted into or deleted from the kinetic tournament in $O(\log^2 n)$ time.

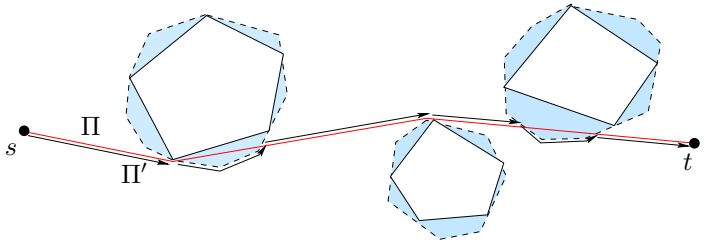


Fig. 1. Surgery on the path Π from s to t to make it free; shaded area represents $\widetilde{\mathcal{F}} \setminus \mathcal{F}$, which consists of a set of convex polygons.

Let O be one of these polygons intersected by Π. Since O is convex and Π cannot cross the base of O, $\Pi \cap O$ is a line segment between two points p,q on the dome of O. We modify Π to bypass O, by replacing the segment pq on Π with the path Π_{pq} between p and q along the dome of O. Clearly, $\Pi_{pq} \subseteq \mathcal{F}$. This surgery increases the length of Π, but by not much. More precisely, Let z be the intersection of the two tangents of O at p and q respectively. By the discussion in the preceding paragraph, we have $\angle pzq \geq \pi - \sqrt{2\varepsilon}$. Hence,

$$\begin{aligned} |\Pi_{pq}| &\leq |pz| + |qz| \leq |pq| / \sin(\angle pzq/2) \leq |pq| / \sin\bigl(\pi/2 - \sqrt{\varepsilon/2}\bigr) \\ &\leq |pq|/(1-\varepsilon/2) \leq (1+\varepsilon)|pq|. \end{aligned}$$

Here, the first inequality follows from a standard convexity argument, the second inequality follows from elementary trigonometry, and the last inequality assumes $\varepsilon \leq 1$.

We perform the above surgery on Π for each of the polygons in $\widetilde{\mathcal{F}} \setminus \mathcal{F}$ intersected by the original path Π. In the end, the new path Π' lies in $\mathcal{F}$. Therefore $d_{\mathcal{F}}(s,t) \leq |\Pi'| \leq (1+\varepsilon) d_{\widetilde{\mathcal{F}}}(s,t)$, and $\widetilde{\mathbb{P}}$ is indeed an ε-sketch of $\mathbb{P}$. □

The total number of vertices of $\widetilde{\mathbb{P}}$ is $O(k/\sqrt{\varepsilon})$. For query points $s,t \in \mathcal{F}$, we use the algorithm of Hershberger and Suri [13] to compute $d_{\widetilde{\mathcal{F}}}(s,t)$ in time $O\bigl(|\widetilde{\mathbb{P}}| \log |\widetilde{\mathbb{P}}|\bigr) = O((k/\sqrt{\varepsilon}) \log(k/\varepsilon))$ and return $(1+\varepsilon) d_{\widetilde{\mathcal{F}}}(s,t)$ which is a $(1+\varepsilon)$-approximation of $d_{\mathcal{F}}(s,t)$ by the preceding lemma.

Theorem 2.1. *Let $\mathbb{P} = \{P_1, \cdots, P_k\}$ be a collection of pairwise disjoint deforming convex polygons in $\mathbb{R}^2$, where each P_i is defined as the convex hull of a set of n_i moving points. Let $n = \sum_{i=1}^{k} n_i$. There is a kinetic data structure that reports, in $O((k/\sqrt{\varepsilon}) \log(k/\varepsilon))$ time, a $(1+\varepsilon)$-approximation to the geodesic distance between two arbitrary query points s,t in the free space. The data structure has $O(n/\sqrt{\varepsilon})$ size and processes $O(\lambda_c(n)/\sqrt{\varepsilon})$ events in total, each requiring $O(\log^2 n)$ time. A point (used for defining one of the convex hulls) can be inserted or deleted or change its motion in $O((1/\sqrt{\varepsilon}) \log^2 n)$ time.*

3 Geodesic-Distance Queries in $\mathbb{R}^3$

In this section we consider the geodesic-distance query problem amidst three-dimensional dynamic point clouds. We first describe the data structure for the case of one single point cloud; and then extend it to multiple point clouds.

3.1 Single Polytope

Consider a single polytope P defined as the convex hull of a set S of n moving points in $\mathbb{R}^3$. We start by considering the special case in which the query points s,t both lie on the surface of P (*boundary query*) and then extend to the case in which s,t are two arbitrary points in the free space $\mathcal{F}$ (*generic query*). In the former case, it is well known that $\Pi_{\mathcal{F}}(s,t)$ is also a path lying on the boundary of P, i.e., $d_{\mathcal{F}}(s,t) = d_{\partial P}(s,t)$.

Existing data structures for geodesic-distance query on the boundary of a convex polytope P [11] make use of the Dobkin-Kirkpatrick hierarchy [10] of P. However, as in the two-dimensional case, we do not want to explicitly maintain P explicitly, nor its Dobkin-Kirkpatrick hierarchy. The starting point of our algorithm is the work of Hershberger and Suri [12], in which a linear-time procedure is proposed for computing a constant-factor approximation for the geodesic-distance between two points $s,t \in \partial P$. Their procedure makes use of the two unit normals u_s and u_t at s and t respectively to guide the computation. Briefly, if the angle between u_s and u_t is small (say, less than $\pi/2$), then the Euclidean distance$\|st\|$ is a good approximation to their geodesic distance; otherwise, there is a point $p \in \partial P$ such that $\|sp\| + \|pt\|$ is a good approximation, and moreover u_p makes small angles with both u_s and u_t, where u_p is the normal to a plane containing p, avoiding the interior of P, and pointing away from P. In our case, since the polytope will not be explicitly maintained, we do not know the normals at s and t and therefore need a more careful design.

Data structure. For a unit vector $u \in \mathbb{S}^2$, we denote the plane $\langle p,u\rangle = 0$ ($p \in \mathbb{R}^3$) by h_u and the great circle $h_u \cap \mathbb{S}^2$ on $\mathbb{S}^2$ by g_u. For a set $X \subseteq \mathbb{R}^3$ and a unit vector $u \in \mathbb{S}^2$, we denote by $\Downarrow_u(X)$ the projection of X onto h_u. On the positive hemisphere $\mathbb{S}^2_+$ (i.e., the closed hemisphere of $\mathbb{S}^2$ lying on or above the xy-plane), we choose a $(\pi/8)$-net $\mathcal{N}$, that is, for any $u,v \in \mathcal{N}$, $\angle u,v \geq \pi/8$, and for any $u \in \mathbb{S}^2_+$, there is a $v \in \mathcal{N}$ so that $\angle u,v \leq \pi/8$ (here $\angle u,v$ denotes the angle between u and v). It can be shown that $|\mathcal{N}| = O(1)$. For each $u \in \mathcal{N}$, we fix arbitrarily a pair $\{u^x,u^y\}$ of orthogonal unit vectors in the plane h_u. We maintain the following information:

(I1) for each $u \in \mathcal{N}$, the convex hull $\mathcal{C}_u$ of $\Downarrow_u(S)$ in the plane h_u;
(I2) for each $\mathcal{C}_u$, an auxiliary data structure $\mathcal{D}_u$ so that given a point $p \in h_u$, it returns, in $O(\log n)$ time, the (at most) four edges of $\mathcal{C}_u$ that the rays from p in directions $\pm u^x, \pm u^y$ first hit.

The data structure can be readily maintained under motion. Each $\mathcal{C}_u$ can be maintained efficiently using a kinetic convex hull algorithm[2]. The data structure $\mathcal{D}_u$ is a balanced binary tree over the edges of $\mathcal{C}_u$. Since $|\mathcal{N}| = O(1)$, we obtain the following.

Lemma 3.1. *The data structure uses $O(n)$ space and can be maintained under motion in $O(\log^2 n)$ time per event, with a total of $O(n\lambda_c(n))$ events.*

Answering a boundary query. A face f of P is called *positive* with respect to a direction $u \in \mathbb{S}^2$ if $\langle u_f, u\rangle \geq 0$, where u_f is the outward unit normal of f, and *negative* if $\langle u_f, u\rangle \leq 0$. The positive faces form a connected component on ∂P called *positive component*, and similarly the negative faces form a *negative component*. The boundary between the positive and negative components consists of a sequence $\mathcal{E}_u$ of edges of P called *horizon edges*. It can be shown that, for any $u \in \mathbb{S}^2$, $\Downarrow_u(\mathcal{E}_u) = \mathcal{C}_u$, i.e., the projection of the horizon edges is the set of edges of the convex hull of $\Downarrow_u(S)$.

For $u \in \mathbb{S}^2$, let $\mathbb{I}_u = \{p \in \mathbb{R}^3 \mid \Downarrow_u(p) \text{ lies inside } \mathcal{C}_u\}$. The set $\mathbb{I}_u$ forms an infinite prism containing P, and $\mathbb{I}_u \cap \mathcal{F}$ consists of two connected components separated by the horizon edges $\mathcal{E}_u$. Given two points $s,t \in \mathbb{I}_u \cap \mathcal{F}$, we say that *$u$ separates s from t* if s,t lie in these two connected components respectively. As a convention, if either s or t lies on $\mathcal{E}_u$, then u also separates s and t. (When s,t are both on ∂P, this definition is equivalent to s,t lie in positive and negative components of ∂P respectively; the general definition presented here will be useful later for generic queries.)

The following procedure estimates the geodesic distance between two query points s and t lying on the boundary of P.

BOUNDARY_QUERY (s,t)
s,t: two points on the boundary of P

(1) $\mathcal{N}_{st} \leftarrow \{u \in \mathcal{N} \mid u \text{ separates } s,t\}$;
(2) **for** each direction $u \in \mathcal{N}_{st}$
$d[u] \leftarrow \|q \Downarrow_u(s)\|_1$,
where q is the point closest to $\Downarrow_u(s)$ on $\mathcal{C}_u$ under the L_1-norm;
(3) $D \leftarrow \|st\| + \sqrt{2} \cdot \max_{u \in \mathcal{N}_{st}} d[u]$;
(by convention, $\max_{u \in \mathcal{N}_{st}} d[u] = 0$ if $\mathcal{N}_{st} = \emptyset$)
(4) **return** D;

Since s,t are on the boundary of P, they lie inside $\mathbb{I}_u \cap \mathcal{F}$ for any $u \in \mathbb{S}^2$. So step (1) is well defined.

[2] A kinetic convex hull algorithm [4, 6] can be used to maintain the convex hull of a set S of n moving points in $\mathbb{R}^2$. There are $O(n\lambda_c(n)\log n)$ events in total, each of which can be processed in $O(\log^2 n)$ time. In addition, each point participates in $O(\log n)$ certificates and hence a motion update can be performed in $O(\log^2 n)$ time. A moving point can also be inserted or deleted in $O(\log^2 n)$ time [4].

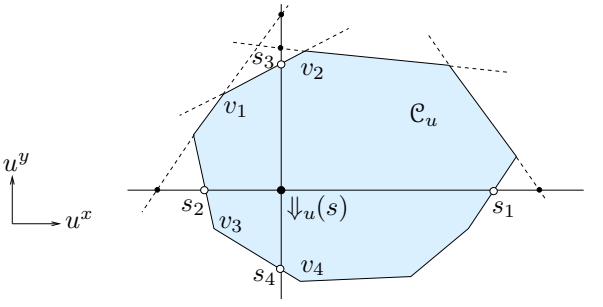


Fig. 2. Using (I2) to test whether u separates s,t, and to compute the closest point of $\Downarrow_u(s)$ on $\mathcal{C}_u$ under L_1-norm.

In step (1), we need to decide for a direction $u \in \mathcal{N}$ whether u separates s from t. This can be done as follows. Let v_1v_2 and v_3v_4 be the two edges of $\mathcal{C}_u$ that $\Downarrow_u(s)$ projects onto in directions $\pm u^y$ (see Figure 2). Let $p_i \in S$ be such that $\Downarrow_u(p_i) = v_i$, for $i = 1,2,3,4$. By examining whether s lies above or below the tetrahedron $p_1p_2p_3p_4$ along direction u, one can decide which component of $\mathbb{I}_u \cap \mathcal{F}$ contains s. (Recall that $s \in \mathcal{F}$, so it does not lie inside the tetrahedron.) A similar statement holds for t. Hence deciding whether u separates s,t can be done in $O(\log n)$ time using (I2).

Since $|\mathcal{N}| = O(1)$, step (1) takes $O(\log n)$ time and step (3) takes $O(1)$ time. The lemma below shows that step (2) also takes $O(\log n)$ time using (I2). Hence the total time for answering a query is $O(\log n)$.

Lemma 3.2. *Let p be any point lying inside $\mathcal{C}_u$. Let $s_1, \cdots, s_4$ be the four projections of p onto $\mathcal{C}_u$ in directions $\pm u^x, \pm u^y$. In the plane h_u equipped with the coordinate frame $\{u^x, u^y\}$, one of $s_1, \cdots, s_4$ is a point on $\mathcal{C}_u$ that is closest to p under L_1-norm.*

Proof. For any line $\ell \subset h_u$, let p_1 and p_2 be the projections of p onto ℓ in direction u^x (or $-u^x$) and u^y (or $-u^y$) respectively. It is easy to verify that if ℓ makes an angle $\leq \pi/4$ with u^x or $-u^x$, then p_2 is the point on ℓ closest to p (under L_1-norm), and otherwise p_1 is the closest. So either p_1 or p_2 realizes the closest point to p. Now, consider the projections of p in direction u^x to all the lines containing an edge of $\mathcal{C}_u$. The point s_1 is closest to p among all such projections (see Figure 2). Similarly, s_2 (resp., s_3, s_4) is the point closest to p among all projections of p onto these lines in direction $-u^x$ (resp., $u^y, -u^y$). Hence, the closest point on $\mathcal{C}_u$ to p must be one of $s_1, \cdots, s_4$. □

Next we show that BOUNDARY_QUERY indeed produces a constant-factor approximation to $d_{\partial P}(s,t)$. Observe that, if two points $s,t \in \partial P$ are separated along a direction $u \in \mathcal{N}$, then s lies on the positive component and t lies on the negative component, or vice versa. In particular, if there are outward unit normals u_s and u_t of P at s and t that lie on different sides of the great circle g_u on $\mathbb{S}^2$, then u separates s from t.

Lemma 3.3. $D \leq 3 \cdot d_{\partial P}(s,t)$.

Proof. If $\mathcal{N}_{st} = \emptyset$, then trivially $D = \|st\| \leq d_{\partial P}(s,t)$. So assume $\mathcal{N}_{st} \neq \emptyset$. Let u be any direction in $\mathcal{N}_{st}$. Since u separates s,t, one of s,t lies on the positive component and the other lies on the negative component. Hence by continuity, $\Pi_{\partial P}(s,t)$ intersects $\mathcal{E}_u$ at some point p. Note that $\Downarrow_u(p) \in \mathcal{C}_u$. Let q be defined as in (2) of BOUNDARY_QUERY. Then,

$$\begin{aligned} d_{\partial P}(s,t) &\geq \|sp\| \geq \| \Downarrow_u(s) \Downarrow_u(p)\| \geq \| \Downarrow_u(s) \Downarrow_u(p)\|_1/\sqrt{2} \\ &\geq (1/\sqrt{2}) \cdot \|q \Downarrow_u(s)\|_1 = (1/\sqrt{2}) \cdot d[u]. \end{aligned}$$

Since trivially $d_{\partial P}(s,t) \geq \|st\|$, we have

$$\|st\| + \sqrt{2} \cdot d[u] \leq 3 \cdot d_{\partial P}(s,t).$$

As this is true for any $u \in \mathcal{N}_{st}$, we have $D \leq 3 \cdot d_{\partial P}(s,t)$ as claimed. □

For a plane $h \subseteq \mathbb{R}^3$ avoiding P, we use h^+ to denote the halfspace bounded by h and containing P. For two non-parallel planes h_1, h_2 both avoiding P, the boundary of $h_1^+ \cap h_2^+$ is called a *wedge*, and the dihedral angle of h_1, h_2 in the quadrant $h_1^+ \cap h_2^+$ is the *angle* of this wedge. It is shown in [12] that, for any two points p,q on a wedge W of angle θ, $d_W(p,q) \leq \|pq\|/\sin(\theta/2)$.

Lemma 3.4. $D \geq (\sin(\pi/16)/\sqrt{2}) \cdot d_{\partial P}(s,t)$.

Proof. Let u_s, u_t be the outward unit normals of P at s and t respectively (if the normal is not unique, choose one arbitrarily). We first consider the case in which the angle between u_s and u_t (denoted by $\angle u_s, u_t$) is at most $\pi/2$. Let W be the wedge formed by the two tangent planes of P at s and t with normals u_s and u_t respectively. The angle of W is $\pi - \angle u_s, u_t \geq \pi/2$. Therefore,

$$d_{\partial P}(s,t) \leq d_W(s,t) \leq \|st\|/\sin(\pi/4) = \sqrt{2} \cdot \|st\|.$$

Hence $D \geq \|st\| \geq (1/\sqrt{2}) \cdot d_{\partial P}(s,t)$.

Next consider the case $\angle u_s, u_t \geq \pi/2$. Let γ be the middle point on the geodesic arc between u_s, u_t on $\mathbb{S}^2$, and $\eta \in \mathbb{S}^2_+$ be the direction orthogonal to the great circle of $\mathbb{S}^2$ passing through γ. Since $\mathcal{N}$ is a $(\pi/8)$-net of $\mathbb{S}^2_+$, there exists a direction $u \in \mathcal{N}$ for which $\angle u, \eta \leq \pi/8$. Using elementary spherical geometry, it can be shown that u_s and u_t lie on different sides of g_u and thus u separates s,t; furthermore, the angle between u_s and the plane h_u is at least $(1/2) \cdot \angle u_s, u_t - \pi/8 \geq \pi/8$, implying that $\angle u_s, v \leq \pi - \pi/8$ for all $v \in g_u$.

Let p be a point on $\mathcal{E}_u$ such that $\Downarrow_u(p) = q$ for q defined in (2) of BOUNDARY_QUERY. Since $p \in \mathcal{E}_u$, there is an outward unit normal u_p at p such that $u_p \in g_u$; in particular, $\angle u_s, u_p \leq \pi - \pi/8$ by the preceding discussion. Let W be the wedge formed by the two tangent planes of P at s and p with normals u_s and u_p respectively. The angle of W is $\pi - \angle u_s, u_p \geq \pi/8$. So,

$$d_{\partial P}(s,p) \leq d_W(s,p) \leq \|sp\|/\sin(\pi/16).$$

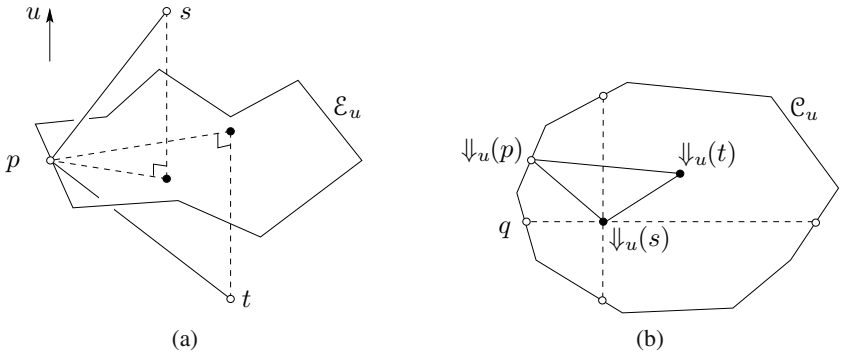


Fig. 3. Illustration for the proof of Lemma 3.4.

Similarly, $d_{\partial P}(p,t) \leq \|pt\|/\sin(\pi/16)$. Then we have

$$\begin{aligned}
d_{\partial P}(s,t) &\leq d_{\partial P}(s,p) + d_{\partial P}(p,t) \\
&\leq \big(\|sp\| + \|pt\|\big)/\sin(\pi/16) \\
&\leq \big(\| \Downarrow_u(s)\, \Downarrow_u(p)\| + \| \Downarrow_u(t)\, \Downarrow_u(p)\| + \langle s-t, u\rangle\big)/\sin(\pi/16) \\
&\leq \big(2\cdot\| \Downarrow_u(s)\, \Downarrow_u(p)\| + \| \Downarrow_u(s)\, \Downarrow_u(t)\| + \langle s-t, u\rangle\big)/\sin(\pi/16) \\
&\leq \big(2\cdot\| \Downarrow_u(s)\, \Downarrow_u(p)\| + \sqrt{2}\cdot\|st\|\big)/\sin(\pi/16) \\
&\leq \big(2\cdot\| \Downarrow_u(s)\, \Downarrow_u(p)\|_1 + \sqrt{2}\cdot\|st\|\big)/\sin(\pi/16) \\
&= \big(2\cdot d[u] + \sqrt{2}\cdot\|st\|\big)/\sin(\pi/16).
\end{aligned}$$

Hence, $D \geq \|st\| + \sqrt{2}\cdot d[u] \geq (\sin(\pi/16)/\sqrt{2})\cdot d_{\partial P}(s,t)$. □

By Lemmas 3.3 and 3.4, it immediately follows that:

Lemma 3.5. BOUNDARY_QUERY(s,t) *returns a constant-factor approximation to* $d_{\partial P}(s,t)$ *in* $O(\log n)$ *time.*

Answering a generic query. Now we generalize the above boundary query to generic queries in which s,t are two arbitrary points in $\mathcal{F}$. The data structure remains the same as in (I1) and (I2). We only have to slightly change line (1) of the query procedure BOUNDARY_QUERY to the following.

(1') $\mathcal{N}_{st} \leftarrow \{u \in \mathcal{N} \mid u$ separates s,t, or $\Downarrow_u(s)$ or $\Downarrow_u(t)$ lies outside $\mathcal{C}_u\}$;

Note that (I2) can be used to determine whether a point $p \in h_u$ lies inside $\mathcal{C}_u$ or not. So (1') takes $O(\log n)$ time. The resulting query procedure is GENERIC_QUERY.

Lemma 3.6. GENERIC_QUERY(s,t) *returns a constant-factor approximation to* $d_{\mathcal{F}}(s,t)$ *in* $O(\log n)$ *time.*

Proof. We first prove the following claim: $d_{\mathcal{F}}(s,t) = d_{\partial P'}(s,t)$, where $P' = \operatorname{conv} S \cup \{s,t\}$. When either s or t does not lie on the boundary of P, there are two cases. If

s and t are visible to each other, then clearly $\Pi_{\mathcal{F}}(s,t)$ is the line segment between s and t, which also appears as an edge on the boundary of P'. If s and t are not visible, then $\Pi_{\mathcal{F}}(s,t)$ consists of three components: a line segment from s to a point $s' \in \partial P$, a geodesic path from s' to a point $t' \in \partial P$ (which lies on the boundary of P), and a line segment from t' to t. Note that no points in the interior of the path $\Pi_{\partial P}(s',t')$ are visible to either s or t, as otherwise $\Pi_{\mathcal{F}}(s,t)$ can be shortcutted. Hence $\Pi_{\mathcal{F}}(s,t)$ must lie on the boundary of P', and the claim follows.

Therefore, to compute $d_{\mathcal{F}}(s,t)$, it suffices to compute $d_{\partial P'}(s,t)$. Imagine running BOUNDARY_QUERY (s,t) on P'. Lemma 3.5 then guarantees that it would return a constant-factor approximation. Let $\mathcal{N}'_{st}$ be the set produced at line (1) of its execution, and $\mathcal{N}_{st}$ be the set produced at line (1') of GENERIC_QUERY. Consider a direction $u \in \mathcal{N}$. If both $\Downarrow_u(s)$ and $\Downarrow_u(t)$ lie inside $\mathcal{C}_u$, then clearly $u \in \mathcal{N}_{st}$ if and only if $u \in \mathcal{N}'_{st}$. If either $\Downarrow_u(s)$ or $\Downarrow_u(t)$ lies outside $\mathcal{C}_u$, we have $u \in \mathcal{N}_{st}$; but at the same time, one of $\Downarrow_u(s), \Downarrow_u(t)$ must lie on the corresponding $\mathcal{C}'_u$ for P' and therefore $u \in \mathcal{N}'_{st}$. Hence, $\mathcal{N}'_{st}$ is the same as $\mathcal{N}_{st}$. The correctness of GENERIC_QUERY then follows. □

Remark. It is possible to modify GENERIC_QUERY so that it also reports a free path between s,t whose length is within a constant factor of $d_{\mathcal{F}}(s,t)$, although P is not explicitly maintained. We omit the details.

3.2 Tradeoffs

So far we have described a kinetic data structure for the approximate geodesic-distance query problem on the boundary of a single polytope, which uses $O(n)$ space, processes $O(n\lambda_c(n))$ events, each requiring $O(\log^2 n)$ processing time, and answers queries in $O(\log n)$ time. These performance bounds are favorable when there are many queries. However, when the number of queries is expected to be small, then a scheme to trade query efficiency for kinetic maintenance efficiency is desirable. We present such a scheme by using generalized linear programming [15].

Let m be an adjustable integer parameter between 1 and n. We divide S into m groups $S_1, \cdots, S_m$, each of size n/m (assume for the sake of simplicity that n is a multiple of m). For each $u \in \mathcal{N}$, instead of maintaining $\mathcal{C}_u$ directly as in (I1), we maintain the convex hull $\mathcal{C}_{u,i}$ of $\Downarrow_u(S_i)$ in the plane h_u, for each $i = 1, \cdots, m$, using the kinetic convex hull algorithm. Clearly, the total size of the data structure remains $O(n)$. The total number of events is $m \cdot O((n/m)\lambda_c(n/m)) = O(n\lambda_c(n/m))$, and the time to process each event remains $O(\log^2 n)$.

We next explain how to compute the projection of an arbitrary point $p \in h_u$ onto $\mathcal{C}_u$ in direction u^y, in $O(m\log(n/m))$ time using the $\mathcal{C}_{u,i}$'s. By handling each of the other directions $-u^y, \pm u^x$ similarly, we thereby fulfill (I2). The query times of both BOUNDARY_QUERY and GENERIC_QUERY are dominated by the query time provided here.

For simplicity, assume that h_u is xy-plane, p is the origin, and u^y is $+y$-axis. We first describe a simple but slower procedure that runs in $O(m^2 \log(n/m))$ time, which will become useful later. We only consider the *upper chain* $\mathcal{C}^+_{u,i}$ of each $\mathcal{C}_{u,i}$

in direction u^y, that is, the upper part of $\mathcal{C}_{u,i}$ lying between its two extreme vertices along $\pm x$-axis. We further divide each $\mathcal{C}^+_{u,i}$ into two sub-chains, the left chain $\mathcal{L}^+_{u,i}$ which lies to the left of $+y$-axis, and the right chain $\mathcal{R}^+_{u,i}$ which lies to the right of $+y$-axis. If there is an edge of $\mathcal{C}^+_{u,i}$ intersecting $+y$-axis, it does not belong to either subchains. For each pair of left chain $\mathcal{L}^+_{u,i}$ and right chain $\mathcal{R}^+_{u,j}$, since they are disjoint, we can use an algorithm of Overmars and van Leeuwen [16] to compute their common outer tangent line $\ell_{i,j}$ in $O(\log|\mathcal{L}^+_{u,i}|+\log|\mathcal{R}^+_{u,j}|)=O(\log(n/m))$ time. Let $p_{i,j}$ be the intersection of $\ell_{i,j}$ with $+y$-axis. We can compute all such $p_{i,j}$'s in time $O(m^2\log(n/m))$.

Let v_1v_2 be the edge of $\mathcal{C}_u$ that contains the sought projection of p, and ℓ be the line containing v_1v_2. Assume without loss of generality that v_1 lies to the left of $+y$-axis and v_2 to the right. Consider the left chain $\mathcal{L}^+_{u,a}$ that v_1 belongs to as a vertex, and the right chain $\mathcal{R}^+_{u,b}$ that v_2 belongs to as a vertex. Observe that the line ℓ is tangent to both $\mathcal{L}^+_{u,a}$ and $\mathcal{R}^+_{u,b}$ and thus is their common tangent. As such, the intersection $p_{a,b}$ of $\ell_{a,b}$ (i.e., ℓ) with $+y$-axis is exactly the sought projection of p. For a common tangent $\ell_{i,j}$ other than $\ell_{a,b}$, observe that $p_{i,j}$ belongs to $\operatorname{conv}\mathcal{L}^+_{u,i}\cup\mathcal{R}^+_{u,i}\subseteq\operatorname{conv}\Downarrow_u(S)$ and hence lies below $p_{a,b}$ on $+y$-axis. Thus, the highest point on $+y$-axis among all $p_{i,j}$'s is the projection of p onto $\mathcal{C}_u$ in direction $+y$.

Next we improve the time for finding the projection to $O(m\log(n/m))$, by formulating it as an LP-type problem. Consider the following optimization problem specified by pairs $(\mathcal{H},w)$, where $\mathcal{H}=\{\Downarrow_u(S_i)\mid i=1,\cdots,m\}$ and $w:2^{\mathcal{H}}\to\mathbb{R}$ is a function that maps each subset $\mathcal{G}\subseteq\mathcal{H}$ to the y-coordinate of the projection of p (the origin) onto $\operatorname{conv}\bigcup_{X\in\mathcal{G}}X$ in direction $+y$ ($w(\mathcal{G})=-\infty$ if the projection does not exist). For a subset $\mathcal{G}\subseteq\mathcal{H}$ with $w(\mathcal{G})>-\infty$, a *basis* of $\mathcal{G}$ is a minimal subset $\mathcal{B}$ of $\mathcal{G}$ with $w(\mathcal{B})=w(\mathcal{G})$. The goal is to compute a basis $\mathcal{B}$ of $\mathcal{H}$, from which the value of $w(\mathcal{H})$ then can be computed from $w(\mathcal{B})$. We verify the following properties of $(\mathcal{H},w)$:

Finite basis: every subset of $\mathcal{H}$ has a basis of size at most two. For a subset $\mathcal{G}\subseteq\mathcal{H}$ with $w(\mathcal{G})>-\infty$, Let v_1v_2 be the edge on $\operatorname{conv}\bigcup_{X\in\mathcal{G}}X$ that p projects onto in direction $+y$. Let $X_1,X_2\in\mathcal{G}$ be elements of $\mathcal{G}$ that contain v_1,v_2 respectively. Then clearly $\{X_1,X_2\}$ is a basis of $\mathcal{G}$. See Figure 4.

Monotonicity: for any $\mathcal{F}\subseteq\mathcal{G}\subseteq\mathcal{H}$, $w(\mathcal{F})\le w(\mathcal{G})$. This is because $\mathcal{F}\subseteq\mathcal{G}$ implies $\operatorname{conv}\bigcup_{X\in\mathcal{F}}X\subseteq\operatorname{conv}\bigcup_{X\in\mathcal{G}}X$.

Locality: for any $\mathcal{F}\subseteq\mathcal{G}\subseteq\mathcal{H}$ with $w(\mathcal{G})=w(\mathcal{F})>-\infty$, and any $X\in\mathcal{H}$, $w(\mathcal{G})<w(\mathcal{G}\cup\{X\})$ implies $w(\mathcal{F})<w(\mathcal{F}\cup\{X\})$. Let ℓ be the line containing the edge of $\operatorname{conv}\bigcup_{X\in\mathcal{G}}X$ that p projects onto. The condition $w(\mathcal{G})<w(\mathcal{G}\cup\{X\})$ implies that X contains a point lying above ℓ. By $w(\mathcal{F})=w(\mathcal{G})$ and $\mathcal{F}\subseteq\mathcal{G}$, it can be shown that ℓ is also the line containing the edge of $\operatorname{conv}\bigcup_{X\in\mathcal{F}}X$ that p projects onto. It follows that $w(\mathcal{F})<w(\mathcal{F}\cup\{X\})$.

Matoušek *et al.* [15] showed that such an optimization problem $(\mathcal{H},w)$ can be solved in $O(|\mathcal{H}|\cdot T+E\cdot\log|\mathcal{H}|)$ expected time, where T is the time to test whether $w(\mathcal{B})=w(\mathcal{B}\cup\{X\})$ for some basis $\mathcal{B}$ and element $X\in\mathcal{H}$, and E is the time to compute a basis of $\mathcal{B}\cup\{X\}$ for some basis $\mathcal{B}$ and element $X\in\mathcal{H}$. In our context,

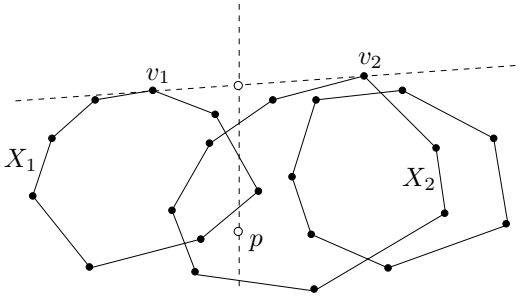


Fig. 4. Reducing computing the projection to an LP-type problem.

$|\mathcal{H}| = m$. Furthermore, any basis consists of at most two elements in $\mathcal{H}$, both T and E are in $O(\log(n/m))$, by applying the aforementioned slower procedure. Hence, the projection of p on $\mathcal{C}_u$ can be computed in $O(m\log(n/m))$ expected time.

Theorem 3.1. *Let P be the convex hull of a set of n points in $\mathbb{R}^3$ under algebraic motion. For any parameter $1 \le m \le n$, there is a kinetic data structure that can be used to report, in $O(m\log(n/m))$ expected time, a constant-factor approximation to the geodesic distance between two arbitrary query points s,t in the free space. The data structure has $O(n)$ size and processes $O(n\lambda_c(n/m))$ events in total, each requiring $O(\log^2 n)$ time. A point (used for defining one of the convex hulls) can be inserted or deleted or change its motion in $O(\log^2 n)$ time.*

3.3 Multiple Polytopes

Let $\mathbb{P} = \{P_1, \cdots, P_k\}$ be a collection of k pairwise disjoint deforming convex polytopes in $\mathbb{R}^3$, where each P_i is the convex hull of a set of n_i moving points. Set $n = \sum_{i=1}^{k} n_i$. We maintain a separate data structure of Theorem 3.1 for each P_i. The total space of these data structures is $\sum_i O(n_i) = O(n)$, and the total number of events is $\sum_i O(n_i\lambda_c(n_i/m)) = O(n\lambda_c(n/m))$.

Let s,t be two query points in the free space $\mathcal{F}$. As observed in [12], one can obtain an $O(k)$-approximation to the geodesic distance between s,t by summing up the query results for each of the k separate data structures. We can improve the approximation factor to $O(k_{st})$ by a simple trick, where k_{st} is the number of polytopes in $\mathbb{P}$ intersected by the line segment st, as follows: among the k returned distances, we simply add up those whose values are $\Omega(\|st\|)$. We omit the details.

Theorem 3.2. *Let $\mathbb{P}$ be a collection of k deforming obstacles each of which is the convex hull of a dynamic point cloud under algebraic motion. Let n be the total number of points in all the point clouds. For any parameter $1 \le m \le n$, there is a kinetic data structure that can be used to report, in $O(mk\log(n/m))$ expected time, a $O(k_{st})$-approximation to the geodesic distance between two arbitrary query points s,t in the free space. The data structure has $O(n)$ size and processes $O(n\lambda_c(n/m))$*

events in total, each requiring $O(\log^2 n)$ time. A point (used for defining one of the convex hulls) can be inserted or deleted or change its motion in $O(\log^2 n)$ time.

References

1. Agarwal, P.K., Aronov, B., O'Rourke, J., Schevon, C.: Star unfolding of a polytope with applications. SIAM J. Comput. 26, 1689–1713 (1997)
2. Agarwal, P.K., Guibas, L., Hershberger, J., Veach, E.: Maintaining the extent of a moving point set. Discrete Comput. Geom. 26, 353–374 (2001)
3. Agarwal, P.K., Sharathkumar, R., Yu, H.: Approximate Euclidean shortest paths amid convex obstacles. In: Proc. 20th ACM-SIAM Sympos. Discrete Algorithms (to appear)
4. Alexandron, G., Kaplan, H., Sharir, M.: Kinetic and dynamic data structures for convex hulls and upper envelopes. Comput. Geom. Theory Appl. 36, 144–158 (2007)
5. Arikati, S., Chen, D., Chew, L., Das, G., Smid, M., Zaroliagis, C.: Planar spanners and approximate shortest path queries among obstacles in the plane. In: Díaz, J. (ed.) ESA 1996. LNCS, vol. 1136, pp. 514–528. Springer, Heidelberg (1996)
6. Basch, J., Guibas, L.J., Hershberger, J.: Data structures for mobile data. J. Algorithms 31, 1–28 (1999)
7. Chen, D.: On the all-pairs Euclidean short path problem. In: Proc. 6th Annu. ACM-SIAM Sympos. Discrete Algorithms, pp. 292–301 (1995)
8. Chiang, Y.-J., Mitchell, J.S.B.: Two-point Euclidean shortest path queries in the plane. In: Proc. 10th Annu. ACM-SIAM Sympos. Discrete Algorithms, pp. 215–224 (1999)
9. Clarkson, K.: Approximation algorithms for shortest path motion planning. In: Proc. 19th Annu. ACM Sympos. Theory Comput., pp. 56–65 (1987)
10. Dobkin, D.P., Kirkpatrick, D.G.: Determining the separation of preprocessed polyhedra — a unified approach. In: Paterson, M. (ed.) ICALP 1990. LNCS, vol. 443, pp. 400–413. Springer, Heidelberg (1990)
11. Har-Peled, S.: Approximate shortest-path and geodesic diameter on convex polytopes in three dimensions. Discrete Comput. Geom. 21, 217–231 (1999)
12. Hershberger, J., Suri, S.: Practical methods for approximating shortest paths on a convex polytope in $\mathbb{R}^3$. Comput. Geom. Theory Appl. 10, 31–46 (1998)
13. Hershberger, J., Suri, S.: An optimal algorithm for Euclidean shortest paths in the plane. SIAM J. Comput. 28, 2215–2256 (1999)
14. Ling, M., Manocha, D.: Collision and proximity queries. In: Goodman, J., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, 2nd edn., pp. 787–808. CRC Press, Boca Raton (2004)
15. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica 16, 498–516 (1996)
16. Overmars, M., van Leeuwen, J.: Maintenance of configurations in the plane. J. Comput. Syst. Sci. 23, 166–204 (1981)
17. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. J. Assoc. Comput. Mach. 41, 764–790 (1994)
18. van den Berg, J.: Path Planning in Dynamic Environments. PhD thesis, Utrecht University (2007)

Constrained Motion Interpolation with Distance Constraints

Liangjun Zhang and Dinesh Manocha

Abstract. We present a novel constraint-based motion interpolation algorithm to improve the performance of local planners in sample-based motion planning. Given two free-space configurations of a robot, our algorithm computes a one-dimensional trajectory subject to distance constraints between the closest features of the robot and the obstacles. We derive simple and closed form solutions to compute a path that guarantees no collisions between these closest features. The resulting local planner is fast and can improve the performance of sample-based planners with no changes to the underlying sampling strategy. In practice, we observe speedups on benchmarks for rigid robots with narrow passages.

1 Introduction

The problem of computing an interpolating motion between two configurations arises in different applications including robot motion planning, kinematics, dynamic simulation, CAD/CAM, and keyframe animation. Given the initial and final configurations, the goal is to compute a one-dimensional function that interpolates the two configurations. Moreover, some applications impose constraints on the resulting trajectory such as smoothness or limits on its derivatives.

In this paper, we address the problem of computing a collision-free interpolating motion between two free-space configurations. The goal is to compute a trajectory that is less likely to intersect with any obstacles in the configuration space (C-space). The main motivation is the local planning step in sample-based planners, which attempts to connect two nearby free-space samples with a collision-free path. Typically, the local planners operate in two steps: computation of an interpolating path and checking that path for collisions with the obstacles. In practice, one of

Liangjun Zhang and Dinesh Manocha
University of North Carolina at Chapel Hill
e-mail: {zlj,dm}@cs.unc.edu
http://gamma.cs.unc.edu/RRRT/CMI

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 367–384.
springerlink.com

the most time-consuming steps in sample-based planners is checking whether the motion produced by the local planner is collision-free or not [3, 7, 15, 17, 18].

The performance of sample-based planners may degrade when the free space has narrow passages. The narrow passages are defined as small regions of free space whose removal or perturbation can change the connectivity of the free space. Most of the prior work in improving the performance of planners has focused on increasing the probability of sampling in these regions [2, 6, 9, 22]. The underlying philosophy of these sampling strategies is that the planner would eventually generate a sufficient number of samples in and around the narrow passages, which can be easily connected using simple local planning algorithms (e.g. linear interpolation). However, the problem of generating sufficient number of samples in narrow passages is non-trivial. Moreover, the narrow passages may have poor visibility properties [9], which can result in high failure rates for the interpolation paths computed by the local planners. Some powerful local planners have been proposed to connect the samples [1, 7, 10], but they can either result in more expensive collision checking or do not take into account the position of the obstacles in the environment.

Main Results: We present a novel motion interpolation algorithm to improve the performance of local planners for sample-based motion planning. Given two free-space configurations, $\mathbf{q}_0$ and $\mathbf{q}_1$, our algorithm maintains distance constraints between the closest feature pairs at these configurations. We derive simple and closed form solutions to guarantee that the sign of the distance between each feature pair does not vary along the trajectory. As a result, there are no collisions between the closest features along that trajectory. Since a local planning algorithm is typically invoked between nearby configurations, the closest features are the most likely candidates for collisions. As a result, our motion interpolation algorithm is more likely to result in a collision-free path as compared to other interpolation schemes that ignore the position of the obstacles in the environment. Our local planner can be combined with sample-based planners with no changes to the sampling strategy or techniques used to compute nearest neighbors. We have combined our algorithm with a retraction-based planner that generates more samples near the contact space and narrow passages. We observe performance gains of the overall planner on benchmarks with narrow passages. Overall, our constrained interpolating motion offers the following benefits:

- **Simplicity:** We present simple and closed form solutions to compute the path based on the closest features.
- **Efficiency:** The main additional overhead of our interpolation algorithm is the computation of closest features at the initial and final configurations, which only takes up to a few milli-seconds.
- **Generality:** Our local planning algorithm is general to all rigid robots that can be represented as polygonal soups. It can also maintain constraints for compliant motion planning.
- **Improved performance:** Our local planning algorithm explicitly takes into account the position of the obstacles in the environment and is effective in connecting nodes in or near the narrow passages.

Organization: The rest of the paper is organized in the following manner. We survey related work on motion interpolation and local planning algorithms in Section 2. Section 3 introduces the notation and gives an overview of our approach. We present the constrained interpolation algorithm for a single distance constraint in Section 4 and extend it to handle multiple constraints in Section 5. We highlight the performance of our local planning algorithm on challenging benchmarks in Section 6. We discuss some properties and extensions of our interpolation scheme in Section 7.

2 Previous Work

In this section, we give a brief overview of previous work on motion interpolation and local planning algorithms.

2.1 Motion Interpolation

Many formulations have been proposed to interpolate the motion between two configurations. The simplest algorithms use straight-line linear interpolation or spherical linear interpolation [12]. Other algorithms tend to compute the minimal-length curve based on appropriate distance metrics, or maintain smoothness constraints [5, 14, 20]. However, these algorithms may not take into account the position of the obstacles in the environment. A related problem is to generate constrained motion that maintains a contact with the given surface or avoid obstacles [11]. Other more general variational-based interpolation schemes [8, 19] have also been proposed. However, the formulation and path computation using these approaches can be expensive as compared to other interpolation techniques.

2.2 Local Planning

The local planning algorithms generate an interpolating motion and check the resulting path for collision with the obstacles. The simplest local planners perform discrete collision detection along a finite number of samples on the continuous path. Discrete collision checking is easy and can be efficiently performed using bounding volume hierarchies. However, there is no guarantee that the portions on the path that are not sampled are collision-free. In order to overcome this issue, some continuous collision detection algorithms based on distance bounds or adaptive bisection have been proposed [18, 15, 23].

Many researchers have analyzed the performance of local planning algorithms and distance metrics, and suggested techniques to improve the performance of the overall motion planner [1, 7, 12]. Improved algorithms for local planning have been designed by combining them with potential field approaches [7] or path optimization [10]. However, these improved local planning algorithms are expensive and can have additional overhead in terms of collision checking.

3 Overview

In this section, we give an overview of our constrained motion interpolation scheme. We further present our formulation for specifying distance constraints algebraically. The interpolation scheme discussed here is for rigid robots. Later in Section 7.4, we present a simple heuristic to extend it to articulated models.

3.1 Notation

We assume that the rigid robot and the obstacles are polyhedral models. We denote the features of vertices, edges, and faces on the boundary of the robot or obstacles as $\mathbf{V}$, $\mathbf{E}$, and $\mathbf{F}$, respectively. We use superscripts to enumerate the features, e.g. $\mathbf{V}^0$, $\mathbf{V}^1$, $\mathbf{V}^2$, and so forth. For the moving robot, we use subscripts to denote the position of its feature at time t, e.g. a specific vertex $\mathbf{V}$ at $t = 0$, $t = 1$ or t is denoted as $\mathbf{V}_0$, $\mathbf{V}_1$, or $\mathbf{V}_t$, respectively. A configuration $\mathbf{q}$ for a rigid robot is represented by using a vector $\mathbf{T}$ for translation and a rotation matrix $\mathbf{R}$, i.e. $\mathbf{q} = (\mathbf{R}, \mathbf{T})$. In order to interpolate two given configurations $\mathbf{q}_0$ and $\mathbf{q}_1$, without loss of generality, we assume $\mathbf{q}_0 = (\mathbf{I}, [0,0,0]^t)$ and the rotational component of $\mathbf{q}_1$ is the rotation matrix about z-axis by θ.

The motion interpolation problem is to compute a one-dimensional trajectory - $\{\mathbf{M}_t = (\mathbf{R}_t, \mathbf{T}_t) | t \in [0,1]\}$ with $\mathbf{M}_0 = \mathbf{q}_0$ and $\mathbf{M}_1 = \mathbf{q}_1$. For example, undergoing a linear interpolating motion, the robot has constant angular and translation velocities with

$$\mathbf{R}_t = \begin{bmatrix} cos(\theta t) & -sin(\theta t) & 0 \\ sin(\theta t) & cos(\theta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and $\mathbf{T}_t$ is $(1-t)\mathbf{T}_0 + t\mathbf{T}_1$.

When the robot moves along an interpolating motion, the position of any point $\mathbf{x}$ on the model at time t can be computed:

$$\mathbf{x}_t = \mathbf{R}_t \mathbf{x}_0 + \mathbf{T}_t, \tag{2}$$

where $\mathbf{x}_0$ is the position of $\mathbf{x}$ at time $t = 0$.

3.2 Constrained Motion Interpolation

Most prior motion interpolation schemes do not take into account the position of the robot and the obstacles in the environment. The resulting interpolating motion may not be collision-free when the robot is near an obstacle. Formally speaking, this corresponds to the situation when the two interpolated configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ are close to the *contact space*, a subset of configurations in C-space at which the robot only touches one or more obstacles without any penetration. Fig. 1(a) shows one example where the robot is near the obstacle. A simple linear interpolation scheme will generate a motion where the robot's rotational center $\mathbf{o}$ undergoes a

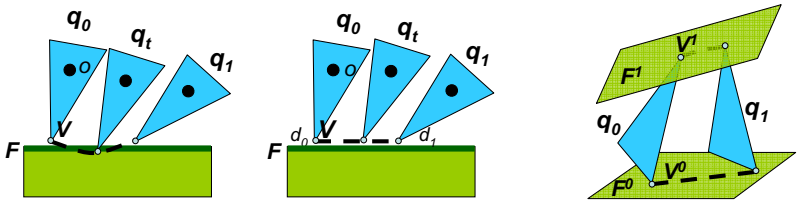


Fig. 1. Motion Interpolation between two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$: (a) There is collision at the intermediate configuration $\mathbf{q}_t$ if we use a linear interpolation; (b) Using our constrained interpolation algorithm, we obtain a collision-free trajectory for this case. (c) We take into account multiple closest feature pairs (($\mathbf{V}^0, \mathbf{F}^0$) and ($\mathbf{V}^1, \mathbf{F}^1$) in this case) at the two configurations, and guarantee no collisions among these feature pairs along the trajectory.

straight line motion. However, due to the affect of the rotation, the other points on the robot (e.g. **V**) follow a non-linear trajectory. Since the robot is near the obstacle in this example, the vertex **V** collides with the face **F** on the obstacle when the robot moves. The collision may happen even when the robot undergoes a small rotation only if the vertex **V** is far from **o**.

Our goal is to compute an interpolating motion that is less likely to result in a collision between the robot and the obstacles. In order to generate such a trajectory between $\mathbf{q}_0$ and $\mathbf{q}_1$, we compute the *closest feature pairs* between the robot at both configurations and the obstacles. Every feature corresponds to a vertex (**V**), an edge (**E**), or a face (**F**) on the boundary. Moreover, we impose constraints so that there is no collision among these closest features along the interpolated trajectory. This is also highlighted in Fig. 1(b), where the constrained motion ensures that the closest feature pair (**V**,**F**) does not collide when the robot moves. Intuitively, in many cases the closest features are the most likely candidates for a collision between the robot and the obstacles. By ensuring that there is no collision amongst these closest features, our interpolation algorithm is more likely to compute a collision-free trajectory for the entire robot. In this manner, our interpolation scheme takes into account the position of the robot and the obstacles in the environment. Moreover, we show that there is very little extra overhead of using our interpolation algorithm over prior methods.

3.3 Distance Constraints

The main issue to formulate the constrained interpolation is the representation of non-collision constraints among the closest features. We use a sufficient condition given by the following lemma:

Lemma 1. *Let* $\{P^i\}$ *the set of the closest feature pairs between the robot at the configuration* $\mathbf{q}_0$ *and the configuration* $\mathbf{q}_1$*, and the obstacles. If the sign of the distance function* d_t *for each feature pair (formally defined in Section 3.4) dose not change*

when the robot moves along this trajectory, then there is no collision in each feature pair P^i.

We highlight the use of distance constraints based on an example. Consider the case in Fig. 1(b), where there is only one closest feature pair (**V**, **F**) between the robot and the obstacles. Let $\mathbf{V}_0$ ($\mathbf{V}_1$) as the position of the vertex **V** at the configuration $\mathbf{q}_0(\mathbf{q}_1)$. The signed distance between $\mathbf{V}_0$ ($\mathbf{V}_1$) and the plane containing the face **F** is d_0 (d_1), respectively. Furthermore, we assume the signed distances d_0 and d_1 have the same sign. In this case, a constrained motion can be computed by imposing the distance d_t is a linear interplant of d_0 and d_1 with t. This ensures the sign of distance function does change along the motion. In this way, our constrained motion guarantees that there is no collision between these features along the trajectory (as per Lemma 1). Given d_0 and d_1 with the same signs, in order to guarantee that the sign of the whole distance function does not change, a simple but sufficient way is to perform a linear interpolation on the signed distances. Quadratic or more complex interpolation functions may be chosen.

Multiple Distance Constraints: In many cases, taking multiple distance constraints into account increases the probability of computing a collision-free path for the local planner. Fig. 1(c) shows such an example. If we can guarantee that the signs of the distance functions between feature pairs $(\mathbf{V}^0, \mathbf{F}^0)$ and $(\mathbf{V}^1, \mathbf{F}^1)$ do not change along the trajectory, then there is no collision between these feature pairs throughout the motion.

Solving the Constrained System: Under multiple constraints, the motion interpolation is formulated as a constrained system:

$$\mathbf{C}(\mathbf{R}_t, \mathbf{T}_t) = 0, \tag{3}$$

where **C** denotes the collection of distance constraints.

The system is non-linear due to the rotational component $\mathbf{R}_t$. For simplicity, we choose a simple interpolation scheme for $\mathbf{R}_t$ that is independent of the position of the obstacles (e.g. a linear interpolation in Eq. (1)). By plugging $\mathbf{R}_t$, the system reduces to a linear one with three variables in the translational component $\mathbf{T}_t$. If the total number of closest features is less than three, this system is under-constrained and one has to choose a meaningful solution from the infinite solution set. If there are exactly three independent distance constraints, the system has a unique solution. We address these issues in more details in Sections 4 and 5.

3.4 Formulation of Distance Constraints

Given a pair of features from the robot and the obstacles, we want to impose a constraint that the signed distance between the pair of features varies linearly. We consider three possible types of closest feature pairs between the boundaries:

1. (**V**,**F**): the closest features are a vertex on the robot and a face of the obstacle,
2. (**F**,**V**): the closest features are a face of the robot and a vertex of the obstacle,
3. (**E**,**E**): the closest features are an edge of the robot and an edge of the obstacle.

To handle other types of closest feature pairs, we first decompose them. For example, a (**V**,**E**) pair can be decomposed into two (**V**,**F**) pairs.

(V,F) Distance Constraint: Suppose the equation of the plane containing the face **F** is $\{\mathbf{x}|\mathbf{N}\cdot\mathbf{x}+D=0,\|\mathbf{N}\|=1,\mathbf{x}\in\mathbb{R}^3\}$. Let d_0 as the signed distance between the vertex $\mathbf{V}_0$ (**V** at $\mathbf{q}_0$) and this plane. Similarly, d_1 is defined as the distance for $\mathbf{q}_1$. Given d_0 and d_1 with same signs, we want the sign of distance function between the vertex **V** and the plane does not change when the robot moves. This constraint can be satisfied by:

$$\mathbf{N}\cdot(\mathbf{R}_t\mathbf{V}_0+\mathbf{T}_t)+D-d_t=0, \tag{4}$$

with $d_t=(1-t)d_0+td_1$, a linear interpolant of the signed distances d_0 and d_1.

This constraint can be reformulated as $\mathbf{N}_t\cdot\mathbf{T}_t+s_t=0$, where:

$$\begin{cases}\mathbf{N}_t &= \mathbf{N},\\ s_t &= \mathbf{N}\cdot\mathbf{R}_t\mathbf{V}_0+D-d_t.\end{cases} \tag{5}$$

(F,V) Distance Constraint: Given a face on the robot at $\mathbf{q}_0$ with the plane equation $\{\mathbf{x}|\mathbf{N}_0\cdot\mathbf{x}+D_0=0,\|\mathbf{N}_0\|=1,\mathbf{x}\in\mathbb{R}^3\}$ and a vertex **V** on the obstacle, the sign of the distance function between the plane and the point should not change between $t=0$ and $t=1$ (Fig. 2(a)). This can be expressed as:

$$(\mathbf{R}_t\mathbf{N}_0)\cdot(\mathbf{V}-\mathbf{T}_t)+D_0-d_t=0, \tag{6}$$

where $d_t=(1-t)d_0+td_1$.

We reformulate this constraint as $\mathbf{N}_t\cdot\mathbf{T}_t+s_t=0$, where:

$$\begin{cases}\mathbf{N}_t &= -\mathbf{R}_t\mathbf{N}_0,\\ s_t &= (\mathbf{R}_t\mathbf{N}_0)\cdot\mathbf{V}+D-d_t.\end{cases} \tag{7}$$

(E,E) Distance Constraint: Given an edge on the robot at $\mathbf{q}_0$ with end points $\mathbf{a}_0^0$ and $\mathbf{a}_0^1$, and an edge on the obstacle with end points $\mathbf{b}^0$ and $\mathbf{b}^1$, the sign of the distance function between the lines containing the edges should not change between $t=0$ and $t=1$ (Fig. 2(b)). In this case, the normal of the plane that contains both the lines is given as:

$$\mathbf{N}_t=Normalize((\mathbf{b}^1-\mathbf{b}^0)\times(\mathbf{R}_t(\mathbf{a}_0^1-\mathbf{a}_0^0))) \tag{8}$$

This constraint can be expressed as:

$$\mathbf{N}_t\cdot(\mathbf{R}_t\mathbf{a}_0^0+\mathbf{T}_t-\mathbf{b}^0)-d_t=0, \tag{9}$$

where $d_t=(1-t)d_0+td_1$.

The constraint then can be reformulated as $\mathbf{N}_t\cdot\mathbf{T}_t+s_t=0$, with $s_t=\mathbf{N}_t\cdot(\mathbf{R}_t\mathbf{a}_0^0-\mathbf{b}^0)-d_t$.

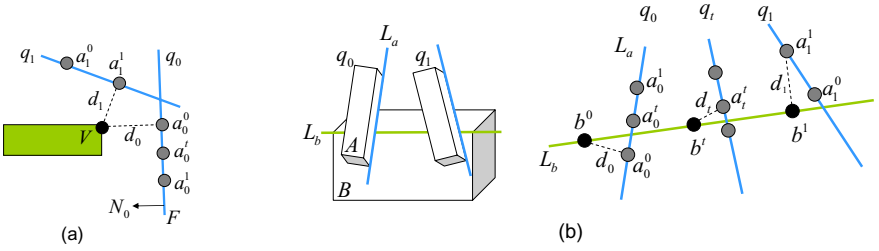


Fig. 2. (a) Motion Interpolation with a single (F,V) distance constraint (Section 4.2). We highlight the closest points on the face F of the robot at $t = 0$, $\mathbf{a}_0^0$ and at $t = 1$, $\mathbf{a}_1^1$. The closest feature on the obstacle is the vertex **V**. (b) *Motion Interpolation with a single (E,E) Distance Constraint (Section 4.3). These edges correspond to L_a on the robot and L_b on the obstacle. We also highlight the closest points on the edges at $t = 0$ and $t = 1$.*

4 Motion Interpolation: Single Distance Constraint

In the previous section, we presented our formulation for specifying the distance constraints for different types of feature pairs. In order to compute the interpolating motion, we need to consider different combinations of closest feature pairs at the configuration $\mathbf{q}_0$ and $\mathbf{q}_1$. In this section, we present our motion interpolation algorithm for a single (**V**, **F**), (**F**, **V**), or (**E**,**E**) *distance constraint*. In other words, the closest feature pair at $\mathbf{q}_0$ and $\mathbf{q}_1$ is identical. With only one distance constraint, the system for computing the motion (Eq. (3)) is under-constrained even if the rotational component $\mathbf{R}(t)$ is given. Its solution set at any time t is a two-dimensional plane. We compute an interpolation motion using a simple geometric construction scheme. The resulting motion satisfies the distance constraint specified in Section 3.4.

4.1 Motion Interpolation with (V,F) Distance Constraint

We first consider the case when the closest features of the robot and the obstacle are a vertex **V** and a face **F**, respectively. In order to generate an interpolation motion that satisfies the distance constraint between features **V** and **F** at any time t, our algorithm rotates the robot around the vertex **V** at $\mathbf{q}_0$ (denoted as $\mathbf{V}_0$), instead of its origin **o** (Fig. 1). The robot is meanwhile translated along the vector from $\mathbf{V}_0$ to $\mathbf{V}_1$ (**V** at $\mathbf{q}_1$). More specifically, the equation to compute the coordinate of any point on the robot **x** at time t can be expressed as:

$$\mathbf{x}_t = \mathbf{R}_t(\mathbf{x}_0 - \mathbf{V}_0) + \mathbf{V}_0 + t(\mathbf{V}_1 - \mathbf{V}_0), \tag{10}$$

Therefore, we can represent this motion as:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{R}_t\mathbf{x}_0 + \mathbf{T}_t, \\ \mathbf{T}_t &= -\mathbf{R}_t\mathbf{V}_0 + \mathbf{V}_0 + t(\mathbf{V}_1 - \mathbf{V}_0), \end{aligned} \tag{11}$$

where $\mathbf{R}_t$ is any interpolant on the rotational component.

4.2 Motion Interpolation with (F,V) Distance Constraint

Lets consider the case when the closest features of the robot and the obstacle are a face $\mathbf{F}$ and a vertex $\mathbf{V}$, respectively. The goal is to generate an interpolating motion that satisfies the distance constraint for these features. Based on Fig. 2(a), we denote $\mathbf{a}_0^0$ and $\mathbf{a}_1^1$ as the projected points of the point $\mathbf{V}$ on the plane containing $\mathbf{F}$ at the configurations $\mathbf{q}_0$ and $\mathbf{q}_1$, respectively. We use the symbol $\mathbf{a}_0^1$ to denote the point $\mathbf{a}^1$ at time $t = 0$. At time t, we first compute the point $\mathbf{a}_0^t$, which is the linear interplant of t on the points $\mathbf{a}_0^0$ and $\mathbf{a}_0^1$; we then compute $\mathbf{a}_t^t$, which is the point $\mathbf{a}_0^t$ at time t by using Eq. (2). If we impose the motion such that $\mathbf{a}_t^t$ is the closest point between the point $\mathbf{V}$ and the plane containing $\mathbf{F}$ at time t, the distance constraint between the point $\mathbf{V}$ and this plane can be expressed as:

$$\mathbf{V} - d_t \mathbf{N}_t = \mathbf{R}_t \mathbf{a}_0^t + \mathbf{T}_t, \tag{12}$$

where d_t is a linear interpolant on the signed distances d_0 and d_1; $\mathbf{N}_t = \mathbf{R}_t \mathbf{N}_0$, where $\mathbf{N}_0$ as the normal of the plane at $\mathbf{q}_0$.

Therefore, the translational component $\mathbf{T}_t$ is given as:

$$\mathbf{T}_t = \mathbf{V} - d_t(\mathbf{R}_t \mathbf{N}_0) - \mathbf{R}_t \mathbf{a}_0^t. \tag{13}$$

4.3 Motion Interpolation with (E,E) Distance Constraint

Lets consider the case when the closest features are both edges. The line containing such edges on the robot and the obstacle are denoted as L_a and L_b, respectively (Fig. 2(a)). Moreover, we use the symbols $\mathbf{b}^0$ and $\mathbf{a}_0^0$ as the closest points on the lines L_b and L_a, respectively, at time $t = 0$, while $\mathbf{b}^1$ and $\mathbf{a}_1^1$ are the closest points between the lines at time $t = 1$. We further denote:

$$\begin{aligned} \mathbf{b}^t &= (1-t)\mathbf{b}^0 + t\mathbf{b}^1, \\ \mathbf{a}_0^t &= (1-t)\mathbf{a}_0^0 + t\mathbf{a}_0^1, \\ \mathbf{a}_t^t &= \mathbf{R}_t \mathbf{a}_0^t + \mathbf{T}_t, \\ \mathbf{N}_t &= Normalize(\mathbf{R}_t(\mathbf{a}_0^1 - \mathbf{a}_0^0) \times (\mathbf{b}^1 - \mathbf{b}^0)). \end{aligned} \tag{14}$$

If we impose the motion such that $\mathbf{a}_t^t$ and $\mathbf{b}^t$ are the closest points between the line L_a at time t and the line L_b, their distance constraint now can be expressed as:

$$\mathbf{R}_t \mathbf{a}_0^t + \mathbf{T}_t = \mathbf{b}^t + d_t \mathbf{N}_t, \tag{15}$$

where d_t is a linear interpolant on the signed distances of d_0 and d_1.

Therefore, $\mathbf{T}_t$ can be represented as:

$$\mathbf{T}_t = d_t \mathbf{N}_t + \mathbf{b}^t - \mathbf{R}_t \mathbf{a}_0^t. \tag{16}$$

One can show that our motion interpolation algorithm satisfies the distance constraint, as stated by the following lemma:

Lemma 2. *The motion interpolated by Eq. (11), (13), or (16) satisfies the input distance constraint between the given feature pairs specified by Eq. (5), (7), or (9), respectively.*

5 Motion Interpolation: Multiple Distance Constraints

In the previous section, we presented our motion interpolation algorithm for a single constraint. Our formulation had assumed that the closest features at configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ are same and thereby handle a single distance constraint. In this section, we extend to the case of multiple distance constraints. We compute the locally closest feature pairs between the robot and obstacles. We derive closed forms for our constrained interpolation which can consider up to three locally closest feature pairs. By taking into account multiple constraints, the resulting interpolating motion conforms better to the local geometry of C-obstacles in the configuration space.

5.1 Two Distance Constraints

Similar to the earlier cases, we again assume that the rotational component of the motion, $\mathbf{R}_t$, is interpolated by a simple interpolant (e.g. linear interpolation). The goal of our motion interpolation is to compute so that the signed distances between each of two locally closest features vary according a given interpolant. This can be expressed as:

$$\begin{cases} \mathbf{N}_t^0 \cdot \mathbf{T}_t + s_t^0 = 0, \\ \mathbf{N}_t^1 \cdot \mathbf{T}_t + s_t^1 = 0, \end{cases} \tag{17}$$

where $\mathbf{N}^i$ and s^i is any type of distance constraint formulated in Section 3.4.

The goal is to compute $\mathbf{T}_t$ for this system. With three unknown variables and two equations, this is an under-constrained system. Furthermore, for a given value of time t, the system is linear since both $\mathbf{N}_t^i$ and s_t^i can be calculated according to our distance constraint formulation in Section 3.4. In general, the solution to this linear system is a one-dimensional set. In order to solve the system, we specify one more constraint explicitly as follows. The two equations at time t in Eq. (17) determine a line in $\mathbb{R}^3$. We first compute a base point ζ at any time t (denoted as ζ_t) on this line:

$$\begin{cases} \mathbf{N}_t^0 \cdot \zeta_t + s_t^0 = 0, \\ \mathbf{N}_t^1 \cdot \zeta_t + s_t^1 = 0, \\ \mathbf{N}_t^2 \cdot \zeta_t = 0. \end{cases} \tag{18}$$

Here $\mathbf{N}_t^2$ is given as:

$$\mathbf{N}_t^2 = Normalize(\mathbf{N}_t^0 \times \mathbf{N}_t^1). \tag{19}$$

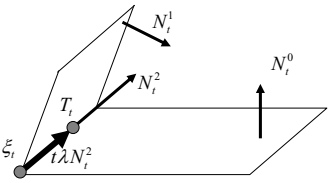


Fig. 3. Computation of the translational component $\mathbf{T}_t$ of the interpolating motion with two distance constraints. We use the distance constraints for each type of feature pair defined in Section 3.4, and compute the vector $\mathbf{N}_t^2$ accordingly.

ζ at time t can be easily computed by the following equation:

$$\zeta_t = \begin{bmatrix} \mathbf{N}_t^0 \\ \mathbf{N}_t^1 \\ \mathbf{N}_t^2 \end{bmatrix}^{-1} [-s_t^0, -s_t^1, 0]^T. \tag{20}$$

Based on it, we can show that both ζ_0 and ζ_1 are 0.

Now, the base point ζ_t is displaced along the direction $\mathbf{N}_t^2$. The equation to compute $\mathbf{T}_t$ can be expressed as:

$$\begin{aligned} \mathbf{T}_t &= \zeta_t + t\lambda \mathbf{N}_t^2 \\ &= \begin{bmatrix} \mathbf{N}_t^0 \\ \mathbf{N}_t^1 \\ \mathbf{N}_t^2 \end{bmatrix}^{-1} [-s_t^0, -s_t^1, 0]^T + t\lambda \mathbf{N}_t^2, \end{aligned} \tag{21}$$

where $\lambda = \mathbf{T}_1 \cdot \mathbf{N}_1^2$, so that the result of this equation at time $t = 1$ interpolates $\mathbf{T}_1$.

5.2 *Three Distance Constraints*

Next we consider the case when there are three pairs of closest features between the robot and obstacles. In this case, we obtain a linear system with three constraints. In general, $\mathbf{T}_t$ can be calculated by solving this linear system:

$$\begin{cases} \mathbf{N}_t^0 \cdot \mathbf{T}_t + s_t^0 = 0, \\ \mathbf{N}_t^1 \cdot \mathbf{T}_t + s_t^1 = 0, \\ \mathbf{N}_t^2 \cdot \mathbf{T}_t + s_t^2 = 0, \end{cases} \tag{22}$$

where $\mathbf{N}^i$ and s^i is any type of distance constraint formulated in Section 3.4, as shown in Eqs. (6), (8) or (10).

Therefore:

$$\mathbf{T}_t = \begin{bmatrix} \mathbf{N}_t^0 \\ \mathbf{N}_t^1 \\ \mathbf{N}_t^2 \end{bmatrix}^{-1} [-s_t^0, -s_t^1, -s_t^2]^T. \tag{23}$$

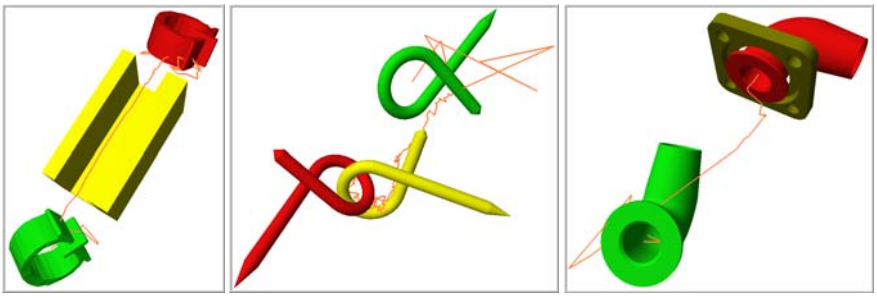


Fig. 4. Benchmarks: We highlight the collision-free paths computed for Notch-g, Alpha Puzzle and Flange benchmarks. Each benchmark has narrow passages. The performance improvement using our constrained motion interpolation algorithm is summarized in Table 2.

Lemma 3. *The motion computed by Eq. (27) or (23) satisfies the input two or three distance constraints between the given feature pairs.*

5.3 Degenerate Situations

Our algorithm to compute the interpolating motion with multiple distance constraints is general. Depending on the specific pairs of closest features, it uses the appropriate formulations derived in Section 3.4. However, the formulation can result in degenerate situations, which can happen during the normalization operation in Eq. (19) or the matrix inverse operation in Eqs. (21) and (23). Conceptually, the degeneracy happens when $\mathbf{N}_t$ for two specific constraints becomes parallel at some time t. We can easily detection these situations by computing a bound on dot-product of $\mathbf{N}_t$ for the whole interval $t \in [0, 1]$, either using discrete sampling or a continuous scheme based on interval arithmetic computation. Once a degenerate case is detected, our algorithm generates a new motion interpolation by only considering a subset of the given distance constraints (e.g. only one constraint instead of two constraints).

6 Implementations and Performance

We have implemented our constrained motion interpolation algorithm and used it for local planning step in sample-based motion planning. All the timings reported in this section were taken on a 3.6GHz Xeon PC.

6.1 Implementation and Performance

Our implementation makes no assumption about the models or connectivity and is applicable to all general rigid models that can be represented as polygonal soups.

Table 1. This table gives a breakdown of the timing among different steps of a local planner that uses our constrained interpolation algorithm. This table presents the average timings per query in milli-seconds on different benchmarks. Most of the time is spent is spent in collision checking, which is similar to other local planners.

	Notch-g	Alpha Puzzle	Flange
Compute closest features (ms)	1.315	2.045	18.619
Formulate distance constraints (ms)	0.046	0.137	0.179
Collision checking (ms)	9.418	26.193	69.452

Distance Computation for Polygonal Soup Models: In order to compute multiple closest feature pairs between the robot and the obstacles, we extend a distance computation algorithm in library - *PQP* [13]. More specifically, we determine all feature pairs between the robot at $\mathbf{q}_0$ and $\mathbf{q}_1$, and obstacles, whose distances are less than a user-specified tolerance. This can be efficiently performed by making use of bounding volume hierarchy in *PQP*. We then use a simple heuristic of clustering to choose a set of representative feature pairs, i.e. the locally closest feature pairs between the models [21]. As shown in Fig. 5, we compute two locally closest feature pairs in (a) and three pairs in (b). Compared with other distance computation algorithms (such as Lin-Canny or GJK algorithms), our implementation is able to handle polygonal soup models and compute a set of locally closest feature pairs.

Constrained Motion Interpolation: We use the set of locally closest feature pairs to setup the distance constraints for computing the interpolation motion. Our formulation can take up to three feature pairs. If any feature pair results in a degeneracy situation, we ignore that pair. Table 1 highlights the performance of our constrained interpolation algorithm, showing a breakdown of timing in different steps of the algorithm on various benchmarks. There is very little extra overhead of using our algorithm to compute the motion.

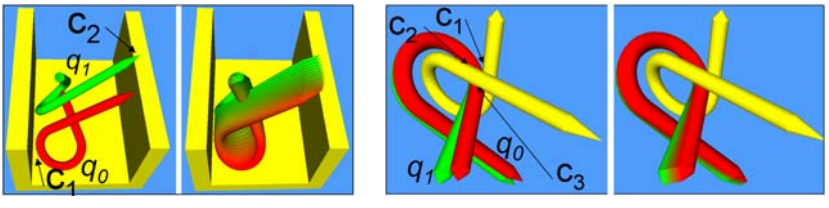
Collision Checking: Though our algorithm guarantees no collisions between the closest features, we still need to check whether there are collision between other features of the robot and obstacles. Currently, we use the simplest method by generating a finite number of discrete samples and performing discrete collision detection at those samples.

6.2 Integration with Sample-Based Planners

Sample-based planners randomly generate samples and connect nearby free-space samples using local planning algorithms. To improve the overall performance of planners, many sampling strategies have been proposed to increase the probability of sampling in narrow passages. Instead, our motion interpolation algorithm is used to improve the local planning step. In general, our interpolation algorithm is more useful for sampling strategies which tend to generate more samples near contact space [2, 15, 16, 4, 22]. For samples in open free space, simple interpolation schemes such linear interpolation can work well. However, when samples are closer

Table 2. Benefit of our constrained motion interpolation algorithm: We compare our local planner based on constrained motion interpolation with other interpolation schemes. All these local interpolation algorithms were integrated with a retraction-based RRT planner and applied to different benchmarks. This table presents the average timings on computing a collision-free path on different benchmarks. In models with complex narrow passages, like Notch-g and Alpha Puzzle, we observe significant speedups due to our planner. Moreover, the resulting planner needs to generate fewer samples.

	Notch-g	Alpha Puzzle	Flange
Constrained Motion (timing)	56.4s	1,043.0s	33.4s
Constrained Motion (# samples)	2,659	53,535	80
Linear Interpolation (timing)	272.1s	1,861.4s	46.3s
Linear Interpolation (# samples)	11,762	94,283	117
Slerp Interpolation (timing)	295.9s	2,029.8s	50.6s
Slerp Interpolation (# samples)	12,568	113,298	113



(a) (b)

Fig. 5. Motion Interpolation With Distance Constraints: given two configurations $\mathbf{q}_0$ and $\mathbf{q}_1$, our algorithm computes a constrained interpolating motion (two constraints in (a) and three constraints in (b)). The resulting motion is more likely collision-free.

to the C-obstacle boundary or lie in narrow passages, they are more difficult to be connected. To address this issue, our constrained motion interpolation formulation takes into account the position of C-obstacle.

In our experiment, we integrate our interpolation algorithm with a variant RRT planner, namely RRRT [22]. The RRRT planner uses a retraction-based sampling strategy to generate more samples near the contact space and bias the exploration towards difficult or narrow regions. We use our constrained motion interpolation to connect samples near contact space. For the other samples (e.g. in the open free space), we use a simpler interpolation scheme, such as linear interpolation. In our current implementation, we perform discrete collision checking on a finite number of samples along the interpolated motion.

Table 2 highlights the performance improvement in our new planner on different benchmarks. The total timing and the number of nodes in the resulting RRT tree are reported. We observe the overall performance improvement on these benchmarks. In Fig. 7, we highlight the performance on a more complex benchmark for part disassembly application.

7 Properties and Extensions

In this section, we first highlight some properties of our constrained motion interpolation scheme. We further show the extensions to compliant motion planner and articulated robots.

7.1 Coordinate-Invariance Property

Our formulation of constrained motion interpolation is coordinate-invariant. We outline the proof as follows and omit the detailed proof due to the space limitation. First, the formulation is left-invariant w.r.t the choice of the inertial frame of the robot. This can be proved in a similar way as [20] by replacing the exponential map with the transformation matrix for representing rotational components. Secondly, the formulation is also right-invariant w.r.t the choice of the body-fixed frame of the robot. Consider the simplest $(\mathbf{V}, \mathbf{F})$ closest feature case. In this motion equation Eq. (11), the terms $\mathbf{x}_0$, $\mathbf{V}_0$ and $\mathbf{V}_1$ do not vary w.r.t the change of the body frame. Therefore, the position $\mathbf{x}_t$ does not change as well. This guarantees right-invariance. For the other cases, we can prove this property in a similar manner.

7.2 Visibility Function Formulation

Our interpolation scheme can improve the visibility between samples near contact space and in narrow passages. In many ways, our constrained motion interpolation algorithm can be interpreted as defining a new visibility function between nearby configuration that is more effective for sample-based planners. This is shown by Table 3. We generate a set of samples near the contact space of the robot and obstacles. Then we performs link queries among adjacent samples by using different interpolation schemes in each experiment. We compute the failure ratio of link queries, which is defined as the number of link queries reporting collisions divided by the total number of link queries performed between the samples near the contact space. Table. 3 highlights the improved ratios obtained using constrained motion interpolation algorithm.

7.3 Compliant Motion Generation

Given two contact configurations $\mathbf{q}_0$ and $\mathbf{q}_1$, our algorithm can be used to interpolate a motion, which maintains a contact among the closest features pairs at $\mathbf{q}_0$ and $\mathbf{q}_1$. In order to generate such a motion, the signed distances d_0 and d_1 in our distance constraint formulations are set as zero, since $\mathbf{q}_0$ and $\mathbf{q}_1$ are contact configurations. Fig. 5(b) shows the motion that maintains the contact among the given feature pairs along the trajectory. We still need to check whether any other features of the robot have penetrated into the obstacle. This formulation can be combined with sample-based compliant motion planners [11].

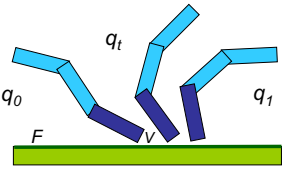


Fig. 6. Articulated Models: our interpolation scheme can be potentially applied to articulated models.

Table 3. Improved Visibility Formulation near Contact Space: Our constrained motion interpolation algorithm is more effective in connecting the samples near the contact space.

	Notch-g	Alpha Puzzle	Flange
Constrained Motion	27.62%	13.22%	16.41%
Linear Interpolation	47.02%	27.82%	22.19%
Slerp Interpolation	46.60%	28.63%	26.57%

7.4 Articulated Models

In Sections 4 and 5, we described our algorithm for rigid robots. In this section, we present a simple heuristic to extend the approach to articulated models. Fig. 6 illustrates a simple scheme for a serial chain robot with movable base. We first compute the link of the robot that is closest to the obstacles. When the robot is near the obstacle, this link would be a likely candidate for collisions with the obstacle. We treat this link as a rigid robot and compute a constrained interpolating motion for it between the two configurations based on the algorithm described in Section 5. Given the motion of this link, we use inverse kinematics to compute the interpolating path

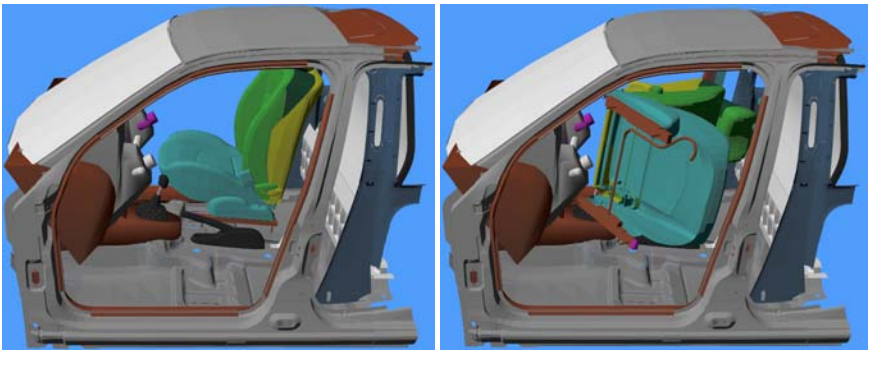


Fig. 7. Disassembly of a Seat outside a Car Body: for this complex benchmark, our planner using constrained motion takes 181.2s while the same planner using linear motion takes 242.7s.

for the other links. Intuitively speaking, we are imposing the geometric constraint on the motion of the link which is most likely to result in collisions with the obstacles.

8 Conclusion and Future Work

We present a simple and general algorithm for motion interpolation and local planning. Based on the closest features at the two configurations, we derive closed formulations of the trajectories. The main additional overhead is the computation of closest features. Our local planning algorithm can be combined with sample-based planners. The main benefit of our approach arises in computing collision-free paths in narrow passages, especially when the samples are very close to the boundary of the contact space. In that case, the paths computed using straight-line linear interpolation, spherical linear interpolation or screw motion may overlap with the obstacles. On the other hand, our formulation adapts to the boundary of the contact space and can generate collision-free paths more likely. As a result, we observe speedups in the performance of overall planner.

Limitations: Our approach has a few limitations. The closed-form formulas for motion interpolation are more complex as compared to other interpolation schemes (e.g. linear interpolation). As a result, there is some additional overhead of computing feature pairs. Furthermore, the parameterization of the constrained motion is not uniform, and we may need to reparameterize in order to compute appropriate discrete samples for collision checking. Most of the prior algorithms for local planning and exact collision checking [18] work well in relatively open space. Therefore, our algorithm is applied only for the cases when the robot becomes nearer any obstacle within a user-defined parameter. This introduces a new parameter that has to be optimized.

Future Work: There are many avenues for future work. We would like to design efficient continuous collision detection algorithm based on motion bound computation. One difficulty is to compute the bounds for the constrained motion with multiple constraints. Furthermore, we would like to perform a detailed performance evaluation on articulated robots with serial and parallel joints. Finally, it may be possible to further improve the performance of the planner by designing appropriate sampling strategies and nearest neighbor selection algorithms that can take into account some of the properties of our motion interpolation algorithm.

Acknowledgements. This project was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134 and 0118743, ONR Contract N00014-01-1-0496, DARPA/RDECOM Contract N61339-04-C-0043 and Intel.

References

1. Amato, N., Bayazit, O., Dale, L., Jones, C., Vallejo, D.: Choosing good distance metrics and local planners for probabilistic roadmap methods. In: Proceedings of ICRA, pp. 630–637 (1998)

2. Amato, N., Bayazit, O., Dale, L., Jones, C., Vallejo, D.: OBPRM: An obstacle-based prm for 3d workspaces. In: Proceedings of WAFR, pp. 197–204 (1998)
3. Bohlin, R., Kavraki, L.E.: Path planning using lazy prm. In: Proceedings of International Conference on Robotics and Automation, pp. 521–528 (2000)
4. Cheng, H.-L., Hsu, D., Latombe, J.-C., Sánchez-Ante, G.: Multi-level free-space dilation for sampling narrow passages in PRM planning. In: Proc. IEEE Int. Conf. on Robotics & Automation, pp. 1255–1260 (2006)
5. Chirikjian, G.S., Kyatkin, A.B.: Engineering Applications of Noncommutative Harmonic Analysis, 1st edn. CRC, Boca Raton (2000)
6. Foskey, M., Garber, M., Lin, M., Manocha, D.: A voronoi-based hybrid planner. In: Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2001)
7. Geraerts, R., Overmars, M.H.: Reachability-based analysis for probabilistic roadmap planners. Robotics and Autonomous Systems 55, 824–836 (2007)
8. Hofer, M., Pottman, H., Ravani, B.: Geometric design of motions constrained by a contacting surface pair. Computer-Aided Geometric Design 20, 523–547 (2003)
9. Hsu, D., Latombe, J., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robotics Research 25(7), 627–643 (2006)
10. Isto, P.: Constructing probabilitic roadmaps with powerful local planning and path optimization. In: Proc. of IROS, pp. 2323–2328 (2002)
11. Ji, X., Xiao, J.: Planning motion compliant to complex contact states. In: Proc. of IROS, pp. 1512–1517 (2001)
12. Kuffner, J.: Effective sampling and distance metrics for 3d rigid body path planning. In: IEEE Int'l Conf. on Robotics and Automation (2004)
13. Larsen, E., Gottschalk, S., Lin, M., Manocha, D.: Distance queries with rectangular swept sphere volumes. In: Proc. of IEEE Int. Conference on Robotics and Automation, pp. 3719–3726 (2000)
14. Park, F.: Distance metrics on the rigid-body motions with applications to mechanism design. ASME J. Mechanical Design 117(1), 48–54 (1995)
15. Redon, S., Lin, M.: Practical local planning in the contact space. In: Proc. of IEEE ICRA (2005)
16. Saha, M., Latombe, J., Chang, Y., Lin, Prinz, F.: Finding narrow passages with probabilistic roadmaps: the small step retraction method. Intelligent Robots and Systems 19(3), 301–319 (2005)
17. Sanchez, G., Latombe, J.: A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In: Int. Symposium on Robotics Research (2001)
18. Schwarzer, F., Saha, M., Latombe, J.: Adaptive dynamic collision checking for single and multiple articulated robots in complex environments. IEEE Tr. on Robotics 21(3), 338–353 (2005)
19. Zefran, M., Kumar, V.: A variational calculus framework for motion planning. In: Proc. IEEE International Conference on Robotics and Automation, pp. 415–420 (1997)
20. Zefran, M., Kumar, V.: Interpolation schemes for rigid body motions. Computer-aided Design 30(3), 179–189 (1998)
21. Zhang, L., Huang, X., Kim, Y., Manocha, D.: D-plan: Efficient collision-free path computation for part removal and disassembly. Journal of Computer-Aided Design and Applications (2008)
22. Zhang, L., Manocha, D.: A retraction-based RRT planner. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 3743–3750 (2008)
23. Zhang, X., Lee, M., Kim, Y.: Interactive continuous collision detection for non-convex polyhedra. In: Pacific Graphics 2006, Visual Computer (2006)

Generating Uniform Incremental Grids on $SO(3)$ Using the Hopf Fibration

Anna Yershova, Steven M. LaValle, and Julie C. Mitchell

Abstract. The problem of generating uniform deterministic samples over the rotation group, $SO(3)$, is fundamental to many fields, such as computational structural biology, robotics, computer graphics, astrophysics. We present the best-known method to date for constructing incremental, deterministic grids on $SO(3)$; it provides the: 1) lowest metric distortion for grid neighbor edges, 2) optimal dispersion-reduction with each additional sample, 3) explicit neighborhood structure, and 4) equivolumetric partition of $SO(3)$ by the grid cells. We also demonstrate the use of the sequence on motion planning problems.

1 Introduction

Discretization of $SO(3)$, the space of 3D rotations, is a difficult problem that arises in numerous engineering and scientific fields. Examples include biological protein docking problems, robot motion planning, aerospace trajectory design, and quantum computations. Typical operations on this space include numerical optimization, searching, integration, sampling, and path generation. Multiresolution grids are widely used for many of these operations if the space is nicely behaved, as in the case of rectangular subsets of $\mathbb{R}^2$ or $\mathbb{R}^3$.

It would be wonderful to achieve the same for $SO(3)$; however, the space of 3D rotations is substantially more complicated. In its basic form, $SO(3)$ is defined as a

Anna Yershova
Duke University, Durham, NC 27707, USA
e-mail: yershova@cs.duke.edu

Steven M. LaValle
University of Illinois, Urbana, IL 61801 USA
e-mail: lavalle@cs.uiuc.edu

Julie C. Mitchell
University of Wisconsin-Madison, Madison, WI 53706
e-mail: mitchell@math.wisc.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 385–399.
springerlink.com

set of matrices that satisfy orthogonality and orientation constraints. It is an implicitly defined, three-dimensional surface embedded in $\mathbb{R}^9$. One approach is to place a coordinate systems on the surface, causing it to behave like a patch in $\mathbb{R}^3$. However, any such coordinates cause metric distortions in comparison to distances on the original surface. Only few representations of $SO(3)$, such as quaternions, preserve distances and volumes. They treat $SO(3)$ as a unit sphere $S^3 \subset \mathbb{R}^4$ with antipodal points identified. The volumes of surface patches on S^3 correspond to the unique Haar measure for $SO(3)$, which is the only way to obtain distortion-free notions of distance and volume. This implies that if we want to make multiresolution grids on $SO(3)$, we are faced with warping them onto S^3. Such curvature prohibits the introduction of distortion-free grids, similarly to the problem of making distance-preserving maps of the world (e.g., Greenland usually looks too big on a flat map). In addition, the identification of antipodal points causes a minor complication in that only half of S^3 is used, with unusual connectivity in the equatorial three-plane.

Due to widespread interest in discretizing $SO(3)$ in numerous fields, there have been considerable efforts in the past. The problem of generating point sets on spheres minimizing such criteria as energy functions, discrepancy, dispersion, and mutual distances has been extensively studied in mathematics and statistics [5, 18, 19]. Problems of sampling rotational groups and spheres have been applied in the context of computational structural biology, physics, chemistry, computer graphics, and robotics [4, 12, 14, 17, 20, 21].

In this paper, we introduce the best-known deterministic method to date for $SO(3)$ that provides the: incremental generation, optimal dispersion reduction with each additional sample, explicit neighborhood structure, lowest metric distortion for grid neighbor edges, and equivolumetric partition of $SO(3)$ into grid cells.

The rest of the paper is organized around the presentation of the method. Section 2 defines the topological properties of $SO(3)$ together with different parametrizations and coordinate systems that are crucial for presenting our method. Section 3 overviews sampling requirements for the sequence. We discuss the relevant sampling methods that influenced our work in Section 4. Finally, we present our method in Section 5 and experimental results in application to several motion planning problems in Section 6. We conclude our work in Section 7.

2 Properties and Representations of $SO(3)$

The *special orthogonal group*, $SO(3)$, arises from rotations around the origin in $\mathbb{R}^3$. Each rotation, by definition, is a linear transformation that preserves the length of vectors and orientation of space. The elements of $SO(3)$ form a group, with the group action being the composition of rotations. $SO(3)$ is not only a group, but also a manifold, which makes it a *Lie group*.

- *Topology of* $SO(3)$. $SO(3)$ is diffeomorphic to the *real projective space*, $\mathbb{RP}^3$. It is hard to visualize the real projective space, because it can not be embedded in $\mathbb{R}^3$. Fortunately, it can be represented as $\mathbb{RP}^3 = S^3/(x \sim -x)$, the more familiar *3-sphere*, S^3, embedded in $\mathbb{R}^4$, with antipodal points identified. Topologists say

that the 3-sphere is a *double covering* of $\mathbb{RP}^3$, since one point of the projective space has two corresponding points on the 3-sphere.

- *Haar Measure on $SO(3)$.* Up to a scalar multiple, there exists a unique measure on $SO(3)$ that is invariant with respect to group actions. This is called the *Haar measure*. That is, the Haar measure of a set is equal to the Haar measure of all of the rotations of the set. In our particular situation, we can think of the Haar measure as being invariant under all orthogonal coordinate changes. The Haar measure is an intrinsic property of $SO(3)$ which comes from the group structure, and is independent of its topological structure.

We have not used any coordinate system or parametrization of $SO(3)$ yet, since the notion of Haar measure is very abstract. One has to use extreme caution when expressing the measure in terms of any of the representations we describe next. Not all of these naturally preserve the Haar measure.

- *Quaternions.* One of the most useful representations of the projective space is the set of quaternions. Let $x = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4$ be a unit quaternion, $x_1 + x_2\mathrm{i} + x_3\mathrm{j} + x_4\mathrm{k}, ||x|| = 1$, representing a 3D rotation. Because of the topological relationship between the projective space and the 3-sphere, once the identifications of the opposite points on the 3-sphere are taken into account, metrics similar to those defined for the 3-sphere can be used for the projective space. Moreover, such metrics will respect the Haar measure on $SO(3)$.

  The most natural way to define a metric for any two points $x, y \in \mathbb{RP}^3$ is as the length of the shortest arc between x and y on the 3-sphere:

$$\rho_{\mathbb{RP}^3}(x,y) = \cos^{-1} |(x \cdot y)|, \tag{1}$$

  in which $(x \cdot y)$ denotes the dot product for vectors in $\mathbb{R}^4$, and the absolute value, $|.|$, guarantees that the shortest arc is chosen among the identifications of the two quaternions [7].

  Quaternion representation is also useful for calculating the composition of rotations, which is expressed as multiplication of quaternions. Any rotation invariant surface measure on S^3 naturally preserves the Haar measure for $SO(3)$ and can be used for quaternions. However, the surface measure is not straightforwardly expressed using quaternions. Other representations, such as spherical or Hopf coordinates, are more convenient for measuring the volume of surface regions.
- *Spherical Coordinates for $SO(3)$.* Because of the relationship between the 3-sphere and $\mathbb{RP}^3$, hyperspherical coordinates can be used for $SO(3)$. Consider a point $(\theta, \phi, \psi) \in S^3$, in which ψ has a range of $\pi/2$ (to compensate for the identifications, we consider only one hemisphere of S^3), θ has a range of π, and ϕ has a range of 2π. For each ψ, the ranges of θ and ψ define a 2-sphere of radius $\sin(\psi)$. The quaternion $x = (x_1, x_2, x_3, x_4)$ corresponding to the rotation (θ, ϕ, ψ) can be obtained using the formula:

$$\begin{aligned} x_1 &= \cos(\psi) \\ x_2 &= \sin(\psi)\cos(\theta) \\ x_3 &= \sin(\psi)\sin(\theta)\cos(\phi) \\ x_4 &= \sin(\psi)\sin(\theta)\sin(\phi). \end{aligned} \tag{2}$$

The volume element of the $SO(3)$ defines the Haar measure and has the following expression in spherical coordinates:

$$dV = \sin^2(\psi)\sin(\theta)d\theta\, d\phi\, d\psi \tag{3}$$

This representation is not convenient for integration though, because of the complicated expression for the Jacobian. Spherical coordinates are also cumbersome for computing compositions of rotations.

- *Hopf Coordinates for $SO(3)$.* As opposed to spherical coordinates for hyperspheres, the *Hopf coordinates* are unique for the 3-sphere, and thus for $\mathbb{RP}^3$. They naturally describe the intrinsic structure of both the 3-sphere and $\mathbb{RP}^3$ and provide a natural tool for obtaining uniform distributions on these spaces.

  The *Hopf fibration* describes $\mathbb{RP}^3$ in terms of a circle S^1 and an ordinary 2-sphere S^2. Intuitively, $\mathbb{RP}^3$ is composed of non-intersecting fibers, such that each

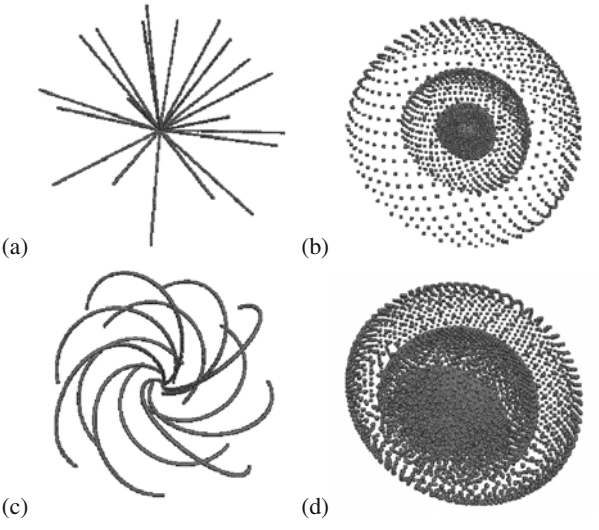


Fig. 1. Visualization of the spherical and Hopf coordinates on $SO(3)$ using angle and axis representation. This representation corresponds to a projection of the S^3 onto the equatorial solid sphere which we draw in R^3. (a) The full range of the spherical coordinate $\psi \in [0, \pi/2]$ is shown while the coordinates (θ, ϕ) form a discretization of size 20 over S^2. (b) The half-spheres show the ranges of the spherical coordinates (θ, ϕ), while ψ takes four discrete values over $[0, \pi/2]$. (c) The full range of the Hopf coordinate $\psi \in [0, 2\pi]$ is shown while the coordinates (θ, ϕ) form a discretization of size 12 over S^2. (b) The spheres show the ranges of the Hopf coordinates (θ, ϕ), while ψ takes four discrete values over S^1.

fiber is a circle corresponding to the 2-sphere. This fiber bundle structure is denoted as $\mathbb{RP}^3 \cong S^1 \widetilde{\otimes} S^2$. The Hopf fibration has the important property of locally being a Cartesian product space. The projective space, $\mathbb{RP}^3$, however, is not (globally) a Cartesian product of S^2 and S^1. Intuitively, $\mathbb{RP}^3$ is the product of S^2 and S^1 similarly to the way the Möbius band is locally the Cartesian product of an interval and a circle S^1. That is, locally a sequence of coordinates from each subspace results in a global parametrization of the space, whereas the global embedding into the Euclidean space does not have the Cartesian product structure. The Hopf coordinates can also be used for the 3-sphere, because of the topological relationship between the 3-sphere and $\mathbb{R}P^3$.

Each rotation in Hopf coordinates can be written as (θ, ϕ, ψ), in which ψ parametrizes the circle S^1 and has the range of 2π. The ranges of θ and ϕ are π and 2π respectively, and they represent spherical coordinates for S^2. The transformation to a quaternion $x = (x_1, x_2, x_3, x_4)$ can be expressed using the formula:

$$\begin{aligned} x_1 &= \cos(\theta/2)\cos(\psi/2) \\ x_2 &= \cos(\theta/2)\sin(\psi/2) \\ x_3 &= \sin(\theta/2)\cos(\phi + \psi/2) \\ x_4 &= \sin(\theta/2)\sin(\phi + \psi/2). \end{aligned} \tag{4}$$

Equation (4) represents each rotation from $SO(3)$ as the rotation by an angle $\psi \in S^1$ around an axis z, *followed* by the rotation, which places z in a position $(\theta, \phi) \in S^2$. Formula (4) is obtained after the *composition* of the two rotations. The volume element on $\mathbb{RP}^3$ which respects the Haar measure is then defined as the surface volume on S^3:

$$\mathrm{d}V = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \, \mathrm{d}\psi. \tag{5}$$

Note that $\sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi$ represents the surface area on the 2-sphere, and $\mathrm{d}\psi$ is the length element on the circle. This formula additionally demonstrates that the volumes from the two subspaces, S^2 and S^1, are simply multiplied to obtain the volume on $SO(3)$. The Hopf coordinates, though, are not convenient for expressing compositions of rotations.

- *Axis-Angle Representation for SO(3).* One of the most intuitive ways to represent rotations is by using *Euler's theorem*, which states that every 3D rotation is a rotation by some angle θ around a unit axis $\mathrm{n} = (n_1, n_2, n_3), ||\mathrm{n}|| = 1$. The transformation from angle and axis representation to quaternions is achieved by:

$$x = (\cos(\theta/2), \sin(\theta/2)n_1, \sin(\theta/2)n_2, \sin(\theta/2)n_3). \tag{6}$$

The angle and axis representation is useful for visualizing the projective space in R^3. Each rotation is drawn as a vector with direction n and a magnitude corresponding to θ (a multiple or a function of θ can be used; see Section 5.5, and [1]). Figure 1 shows the visualization of the spherical and Hopf coordinates on $SO(3)$ using the angle and axis representation. From this visualization one can immediately notice the singularities produced by the spherical coordinates. It is

also possible to see the advantage of using Hopf coordinates from this visualization. Hopf coordinates do not introduce singularities. The circles represented by the range of the variable ψ are non-intersecting; they uniformly cover the $SO(3)$. The fiber structure formed by these circles is also seen on the figure.

- *Euler Angles Representation.* Euler angles are often used in robotics to represent rotations. Each rotation is then a vector $(x_1, x_2, x_3), x_i \in [-\pi, \pi] / -\pi \sim \pi$. The topology of the resulting space is $S^1 \times S^1 \times S^1$, and, therefore, Euler angles do not correctly capture the structure of $SO(3)$. There are many detrimental consequences of this. Special tricks (see [7]) are needed to implement metric and measure that preserve Haar measure. Moreover, Euler angles are hard to compose, and present problems of singularities and the gimbal lock [15, 22]. In the rest of the paper we use Hopf coordinates and quaternions to represent rotations.

3 Sampling Terminology and Problem Formulation

In applications such as motion planning the algorithms are often terminated early and the particular order in which samples are chosen becomes crucial. Sampling literature distinguishes between a sample *set* and a sample *sequence*. For a sample set, the number of points, n, is specified in advance, and a set of n points is then chosen to satisfy the requirements of the method. The notion of ordering between points is not defined for a sample set but becomes important for sequences. Successive points in a sequence should be chosen carefully so that the resulting sample sets are all of good quality. Sequences are particularly suitable for motion planning algorithms, in which the number of points needed to solve the problem is not known in advance.

Now that the background definitions for $SO(3)$ have been presented in Section 2, to generate samples over $SO(3)$ we need to formulate the desirable properties for the samples. The first requirement is that samples form a sequence. We also require that samples get arbitrarily close to every point in $SO(3)$, i.e. that the sequence of samples is *dense* in $SO(3)$. Next we formulate several requirements on the uniformity properties of samples.

3.1 Discrepancy and Dispersion

Additional requirements that the sequence needs to satisfy are described by the uniformity measures, *discrepancy* and *dispersion*.

Intuitively, discrepancy can be thought of as enforcing two criteria: first, that no region of the space is left uncovered; and second, that no region is left too full. Dispersion eliminates the second criterion, requiring only the first. It can be shown that low discrepancy implies low dispersion [11].

To define discrepancy, choose a *range space*, $\mathscr{R}$, as a collection of subsets of $SO(3)$. Let $R \in \mathscr{R}$ denote one such subset. Range spaces that are usually considered on spheres are the set of spherical caps (intersections of the 3-sphere with half-spaces) or the set of spherical slices (intersections of two 3-hemispheres) [13], which can be used on $SO(3)$ once the identifications of the 3-sphere are taken into account.

	Random	Succ. Orth. Images	Layered Sukharev	HEALPix	this work
incremental	yes	no	yes	no	yes
uniform	yes	yes	no	yes	yes
deterministic	no	yes	yes	yes	yes
grid	no	no/yes	yes	yes	yes
spaces	$SO(3)$	$SO(n)$	$SO(3)$ and S^n	S^2	SO(3)

Fig. 2. The comparison of different sampling methods related to the problem of Section 3.2. The rows correspond to the desired properties of these methods.

Let $\mu(R)$ denote the Haar measure of the subset R. If the samples in the set P are uniform in some ideal sense, then it seems reasonable that the fraction of these samples that lie in any subset R should be roughly $\mu(R)$ divided by $\mu(SO(3))$ (which is simply π^2). We define the *discrepancy* [11] to measure how far from ideal the sample set P is:

$$D(P,\mathscr{R}) = \sup_{R\in\mathscr{R}} \left| \frac{|P\cap R|}{N} - \frac{\mu(R)}{\mu(SO(3))} \right| \tag{7}$$

in which $|\cdot|$ applied to a finite set denotes its cardinality.

While discrepancy is based on measure, a metric-based criterion, *dispersion*, can be introduced:

$$\delta(P,\rho) = \max_{q\in SO(3)} \min_{p\in P} \rho(q,p). \tag{8}$$

Above, ρ denotes any metric on $SO(3)$ that agrees with the Haar measure, such as (1). Intuitively, this corresponds to the spherical radius of the largest empty ball that fits in between the samples (assuming all ball centers lie on $SO(3)$).

3.2 Problem Formulation

In summary, the goal of this paper is to define a sequence of elements from $SO(3)$ which:

- is incremental and deterministic
- minimizes the dispersion (8) and discrepancy (7) on $SO(3)$,
- has grid structure with respect to the metric (1) on $SO(3)$.

4 Sampling Methods Overview

Our work was influenced by many successful sampling methods developed recently for spheres and $SO(3)$. As demonstrated in the table of Figure 2, several of them are highly related to the problem formulation in Section 3.2. However, none of the methods known to date has all of the desired properties.

Random Sequence of $SO(3)$ Rotations. There are several ways of sampling the space of rotations uniformly at random [16, 23]. The subgroup algorithm [2] for

selecting random elements for $SO(3)$ is the most popular method for uniform random sampling of $SO(3)$. Essentially, this method utilizes the Hopf coordinates. Another way of obtaining uniformly distributed random samples over $SO(3)$ is by using spherical symmetry of the multidimensional Gaussian density function [3]. Random sequences of rotations are used in many applications, however, they lack deterministic uniformity guarantees, and the explicit neighborhood structure.

Successive Orthogonal Images on $SO(n)$. Related to the subgroup method for generating random rotations is the deterministic method of Successive Orthogonal Images [10], which generates lattice-like sets with a specified length step based on uniform deterministic samples from the subgroup, S^1, and the coset space, S^2. The method utilized Hopf coordinates, and is also generalized to arbitrary $SO(n)$.

The deterministic point sets can be applied to the problems in which the number of the desired samples is specified in advance. If the sample on S^2 is chosen so that it has a grid structure, the resulting sample on $SO(3)$ has the explicit neighborhood structure. Part of the current work will be in applying this method in a way that provides the incremental quality necessary for our motion planning applications.

Layered Sukharev Grid Sequence. Uniform, deterministic sequences were first designed for the unit cube [8]. To minimize dispersion, the method places one resolution of grid at a time inside of the unit cube. A discrepancy-optimal ordering is then generated for each resolution. The sequence can be extended to spheres, and $SO(3)$ [22] using the projection from faces of an inscribed cube. For $SO(3)$, though, the distortions produced by the method result in some grid cells being four times the volume of others.

The general method for designing Layered Sukharev Grid sequences inside Cartesian products was later presented in [9]. Our current paper builds on top of these works by combining the method in [9], with the Successive Orthogonal Images [10] generation of rotations, Hopf coordinates, and the HEALPix spherical sampling method [4] described in the next section.

HEALPix. The HEALPix package [4] was designed for applications in astrophysics. It provides a deterministic, uniform, and multiresolution sampling method for the 2-sphere with additional properties, such as equal-area partitioning, and isolatitude sampling on the 2-sphere.

The method takes advantage of the measure preserving cylindrical projection of the 2-sphere. This intrinsic property of the 2-sphere cannot be generalized directly to higher dimensional spheres. However, this work shows that an extremely uniform grid can be constructed on such a non-trivial curvature space as the 2-sphere. It is also not difficult to make this grid incremental using the method from [9].

5 Our Approach

In this section we present an approach which satisfies all of the requirements of Section 3.2, and Figure 2. The fiber bundle structure of $SO(3)$ locally behaves similarly to the Cartesian product of two spaces, S^1, and S^2. Therefore, the method

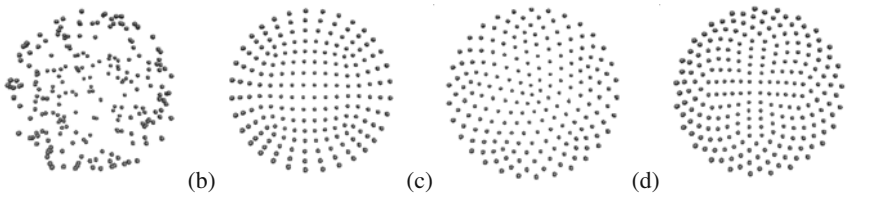


Fig. 3. Different sampling methods on S^2. (a) 200 random samples (b) 192 Sukharev grid samples [22] (c) icosahedron samples (d) 216 HEALPix samples [4]

presented in [9] for constructing multiresolution grid sequences for Cartesian products of spaces, can be used for constructing a grid sequence on $SO(3)$. The resulting rotations are computed using the Hopf coordinates, as was first described in [10]. It is a much simpler problem to construct nicely behaved grids on the 1-sphere and 2-sphere. Hopf coordinates allow the two grids be lifted to the space of rotations without loss of uniformity. Next we outline the details of this construction.

Let ψ be the angle parametrizing the circle, S^1, and (θ, ϕ) be the spherical coordinates parametrizing the sphere, S^2. Using these coordinates, define T_1 to be the multiresolution grid over the circle, and T_2 be the grid over the sphere. Let m_1 and m_2 be the number of points at the base resolution 0 of the grids T_1 and T_2 respectively.

There are numerous grids that can be defined on S^2 (see Figure 3 for an illustration of some). In this work we have selected the HEALPix grid [4] on S^2, and the ordinary grid for S^1. Both of these grids are uniform, have simple neighborhood structure, and can have multiple resolutions.

Next consider the space $S^2 \widetilde{\otimes} S^1$. The multiresolution grid sequence that we define for $SO(3)$ has $m_1 \cdot m_2 \cdot 2^{3l}$ points at the resolution level l, in which every successive 2^3 points define a cube in Hopf coordinates. Each element of the sequence is obtained by combining the corresponding coordinates in the subspaces, $p = (\theta, \phi, \psi)$, using formula (4). If the grid regions are defined on the two subspaces S^1 and S^2, the corresponding grid regions are also obtained on $SO(3)$ by combining the corresponding coordinates. The dispersion, and discrepancy of the resulting sequence can be easily computed using the representation for the metric and volume element from equations (1), and (5).

5.1 Choosing the Base Resolution

One of the issues arising when combining the two grids from S^1 and S^2 is the length of a grid cell along each of the coordinates. For this we have to match the number of cells in each base grids on both of the subspaces, so that they have cell sides of equal lengths [10]. That is, the following equation should hold for m_1 and m_2:

$$\frac{2\pi}{m_1} = \sqrt{\frac{4\pi}{m_2}}, \tag{9}$$

in which 2π is the circumference of the circle S^1, and 4π is the surface area of S^2.

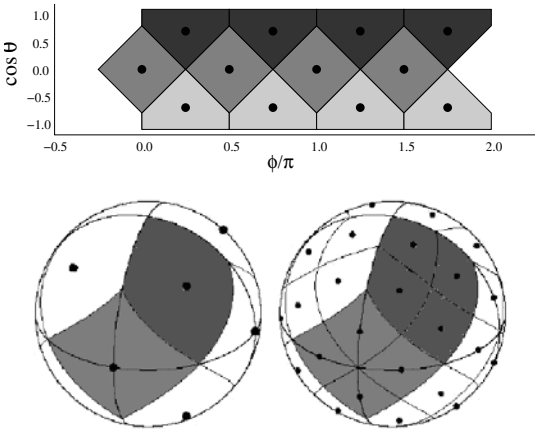


Fig. 4. The base grid of the HEALPix sequence consists of 12 points. The cylindrical projection of the grid cells from S^2 to $(\cos(\theta), \phi)$ coordinates is shown. Each next resolution subdivides each of the spherical squares into 4 squares of equal area[4].

In our particular case, the base HEALPix grid consists of $m_2 = 12$ cells, and the surface area of each cell is equal to $4\pi/12 = \pi/3$ (Figure 4). Therefore, the length of the side of each grid cell is approximately the square root of that value, that is 1. Then, the number of points in the base resolution of the grid on S^1 needs to be $m_1 = 6$, since it should be close to the length of the circle, which is 2π. Therefore, the base grid of the sequence for $SO(3)$ consists of $m_1 \cdot m_2 = 6 \cdot 12 = 72$ points (the projections of the grid regions on the Hopf coordinates are shown on Figure 5).

5.2 Choosing the Base Ordering

The next step is to choose the ordering of the $m = m_1 m_2$ points within the base resolution on $SO(3)$. In general, the initial ordering will influence the quality of the resulting sequence, and a method similar to [9] can be used for deciding the ordering of the general base sequences.

In our case we have to define the ordering on the first 72 points of the sequence (see Figure 5 for the illustration of the associated grid regions). To do this, one can notice that there are antipodal grid cells in both of the subspaces. Removing antipodal cells from the final sequence can significantly eliminate the number of the base resolution grid cells on $SO(3)$.

In our preliminary experiments in the application to motion planning problems (Section 6) we have manually selected such an ordering. However, it is possible to design a program that would run through the orderings and select the ones that minimize the discrepancy. For the further analysis results we assume that the optimal ordering function $f_{base} : \mathbb{N} \rightarrow [1, \ldots 72]$ is given.

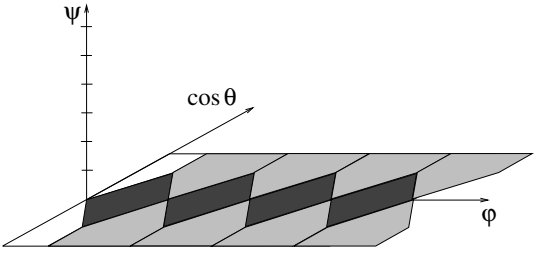


Fig. 5. The base grid of the proposed $SO(3)$ sequence consists of 72 points. For the Hopf coordinates (θ, ϕ, ψ) the projections of the grid cells on each of the coordinates are shown. Grid cells for ψ are chosen according to the ordinary grid on S^1. The grid cells for $(\cos(\theta), \phi)$ are obtained using HEALPix.

5.3 The Sequence

The sequence for $SO(3)$ is constructed one resolution level at a time. The order in which the points from each resolution level are placed in the sequence can be described as follows. The ordering $f_{base}()$ of the first m points in the base resolution determines the order of the grid regions within $SO(3)$ and is taken from the previous section. Every successive m points in the sequence should be placed in these grid regions in the same order. Each of the grid regions is isomorphic to the $[0,1]^3$, and is subdivided into 8 grid regions in each successive resolution. Where exactly each point should be placed within each of the grid regions is determined by the ordering $f_{cube} : \mathbb{N} \to [1,\ldots 8]$ and recursion procedure defined for the cube $[0,1]^3$ in [8].

The resulting procedure for obtaining the coordinates of the ith element in the sequence is the following:

1. Assign $f_{base}(i)$ to be the index of the base grid region that the i-th element has to be placed within.
2. Assign the ceiling of the division, $i_{cube} = \lceil i/m \rceil$, to be the index that determines the subregion of the region $f_{base}(i)$ that the i-th element has to be placed within.
3. Call the recursive procedure from [8] to determine the coordinates of the subregion of the cube $[0,1]^3$ determined by the index i_{cube} and the ordering f_{cube}. The i-th element is then placed within this subregion of the $f_{base}(i)$ region.

5.4 Analysis

Several claims, similar to those obtained in [8], can be made for the new approach. The most important distinction is that the new sequence provides equal volume partition of the $SO(3)$ which results in strong dispersion guarantees.

Proposition 5.1. *The dispersion of the sequence T at the resolution level l satisfies:*

$$\delta(T) \leq 2\sin^{-1}\left(\frac{1}{2}\sqrt{\delta^2(T_2)+\left(\frac{\pi}{m_1 2^l}\right)^2}\right),$$

in which $\delta(T_2)$ is the dispersion of the sequence T_2 defined over S^2.

Proof: The bound follows directly from the Pythagorean theorem, and the dispersion bound on the ordinary grid T_1 at the resolution level l. □

Proposition 5.2. *The sequence T has the following properties:*

- *It is discrepancy-optimal with respect to the set of grid regions defined over S^1 and S^2.*
- *The position of the i-th element of T can be generated in $O(\log i)$ time.*
- *For any i-th sample any of the $2d$ nearest grid neighbors from the same layer can be found in $O((\log i)/d)$ time.*

Proof: The proof closely follows similar considerations in [8]. □

5.5 Visualization of the Results

To visualize our sequence and compare it with other sequences designed for $SO(3)$, we use the angle and axis, (θ, n), representation from Section 2. It can be shown that if the rotations are uniformly distributed, then the distribution of an angle θ is $(\sin(\theta)-\theta)/\pi$ [1]. This allows us to draw the elements of $SO(3)$ as the points inside a ball in such a way that every radial line has uniform distribution of elements. This

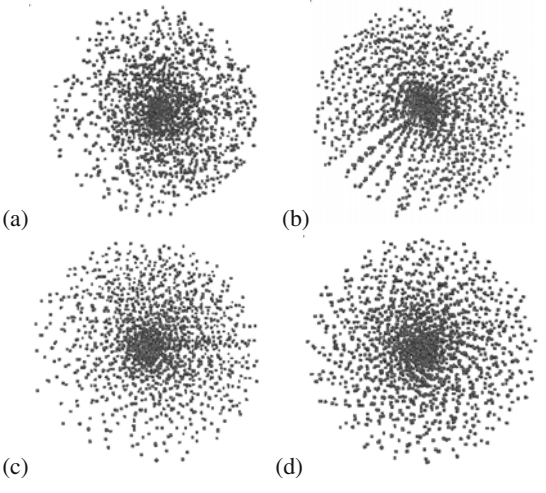


Fig. 6. Different sets of samples on $SO(3)$ (a) 2000 random samples (b) 2048 Sukharev grid samples (c) 1944 icosahedral samples (d) 1944 HEALPix samples

(a) (b)

Fig. 7. Motion planning problems involving: a) moving a robot (black) from the north pole to the south pole. Multiple views of the geometry of the problem are shown (obstacles are drawn in lighter shades); and b) moving a robot along the corridor.

provides a more intuitive visualization, which partially preserves the uniformity. See Figure 6 for visualization of several of the methods of sampling over $SO(3)$, compared to the proposed approach. Specifically, the images show points in the direction of the axis of rotation and with distance to the origin equal to $(\sin(\theta) - \theta)/\pi$. Using this representation, the distribution of points increases linearly as a function of distance from the origin. In comparison, a set of points that was uniform with respect to the measure on $\mathbb{R}^3$ would have a distribution that varies as the cube of distance from the origin.

6 Application: Motion Planning

We have implemented our algorithm in C++ and applied to implementations of PRM-based planner [6] in the Motion Strategy Library. The experiments were performed on a 2.2 Ghz Pentium IV running Linux and compiled under GNU C++.

It is important to note that the experiments we present here are just one of possible applications of the developed sequences to motion planning problems. Alternate applications may exist in other areas of computer science, or related fields.

In our experimental setup we consider the rotation-only models for which the configuration space is $SO(3)$. For the two problems shown in Figure 7 we have compared the number of nodes generated by the basic PRM planner using the pseudo-random sequence (with quaternion components [16]), the layered Sukharev grid sequence, and the new sequence derived from the use of Successive Orthogonal Images, the HEALPix method and incremental Sukharev ordering. For the first problem the results are: 258, 250, and 248 nodes, respectively. To solve the second problem the PRM planner needed 429, 446 and 410 nodes, respectively. In each trial a fixed, random quaternion rotation was premultiplied to each deterministic sample, to displace the entire sequence. The results obtained were averaged over 50 trials.

Based on our results we have observed that the performance of our method is equivalent or better than the performance of the previously known sequences for the basic PRM-based planner. This makes our approach an alternative approach for use in motion planning. It is important to note, however, that for some applications, such as verification problems, only strong resolution guarantees are acceptable.

7 Conclusions

In conclusion, we have developed and implemented a deterministic incremental grid sequence on $SO(3)$ that is highly uniform, can be efficiently generated, and divides the surface of $SO(3)$ into regions of equal volume. Sequences that minimize uniformity criteria, such as dispersion and discrepancy, at each step of generation are especially useful in applications in which the required number of samples is not known in advance. One such application is robotic motion planning. We have demonstrated the use of our method on several motion planning problems, showing that resolution completeness guarantees can be achieved at small computational cost.

There is a number of ways to improve current work, which we plan to address in the future. We plan to complete a more rigorous analysis, as well as comparison of different base sequences for S^2 to improve our understanding of the benefits of our method. A more extensive experimental evaluation of the sequences is also a part of our future work. It is yet inconclusive, but tempting, to assess the general rate of convergence for motion planning solution using different sampling sequences.

There are many general open problems arising from this research. Nicely distributed grids are not yet developed for general n-spheres, $n > 3$. Implicitly defined manifolds, such as the ones arising from motion planning for closed linkages, are very hard to efficiently and uniformly sample. Such manifolds also arise as the conformation spaces of protein loops. In such cases, efficient parametrization is the bottleneck for developing sampling schemes.

Acknowledgements. This work was partially supported by NSF CISE-0535007.

References

1. Chirikjian, G.S., Kyatkin, A.B.: Engineering Applications of Noncommutative Harmonic Analysis. CRC Press, Boca Raton (2001)
2. Diaconis, P., Shahshahani, M.: The subgroup algorithm for generating uniform random variables. Prob. in Eng. and Info. Sci. 1, 15–32 (1987)
3. Fishman, G.F.: Monte Carlo: Concepts, Algorithms, and Applications. Springer, Berlin (1996)
4. Górski, K.M., Hivon, E., Banday, A.J., Wandelt, B.D., Hansen, F.K., Reinecke, M., Bartelmann, M.: HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere. arXiv:astro-ph/0409513 622, 759–771 (April 2005)
5. Hardin, D.P., Saff, E.B.: Discretizing manifolds via minimum energy points. Notices of the American Mathematical Society 51(10), 1186–1194 (2004)
6. Kavraki, L.E., Svestka, P., Latombe, J., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. & Autom. 12(4), 566–580 (1996)
7. Kuffner, J.: Effective sampling and distance metrics for 3D rigid body path planning. In: IEEE Int. Conf. Robot. & Autom. IEEE, Los Alamitos (2004)
8. Lindemann, S.R., LaValle, S.M.: Incremental Low-Discrepancy lattice methods for motion planning. In: IEEE Int'l Conf. on Robotics and Automation, pp. 2920–2927 (2003)
9. Lindemann, S.R., Yershova, A., LaValle, S.M.: Incremental grid sampling strategies in robotics. In: Workshop on the Algorithmic Foundations of Robotics (2004)

10. Mitchell, J.C.: Sampling rotation groups by successive orthogonal images. SIAM J. Sci. Comput. 30(1), 525–547 (2007)
11. Niederreiter, H.: Random Number Generation and Quasi-Monte-Carlo Methods. Society for Industrial and Applied Mathematics, Philadelphia (1992)
12. Ramamoorthy, S., Rajagopal, R., Ruan, Q., Wenzel, L.: Low-discrepancy curves and efficient coverage of space. In: Proc. of the Workshop on Algorithmic Foundations of Robotics (2006)
13. Rote, G., Tichy, R.F.: Spherical dispersion with applications to polygonal approximation of the curves. Anz. Österreich. Akad. Wiss. Math.-Natur, Kl. Abt. II 132, 3–10 (1995)
14. Rovira, J., Wonka, P., Castro, F., Sbert, M.: Point sampling with uniformly distributed lines. In: Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proc. (June 2005)
15. Shoemake, K.: Animating rotation with quaternion curves. In: Proc. of the 12th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press, New York (1985)
16. Shoemake, K.: Uniform random rotations. In: Kirk, D. (ed.) Graphics Gems III, pp. 124–132. Academic Press, London (1992)
17. Sternberg, M.J.E., Moont, G.: Modelling protein-protein and protein-DNA docking. In: Lengauer, T. (ed.) Bioinformatics – From Genomes to Drugs, pp. 361–404. Wiley-VCH, Weinheim (2002)
18. Sun, X., Chen, Z.: Spherical basis functions and uniform distribution of points on spheres. J. Approx. Theory 151(2), 186–207 (2008)
19. Wagner, G.: On a new method for constructing good point sets on spheres. Journal of Discrete and Computational Geometry 9(1), 119–129 (1993)
20. Wales, D., Doye, J.: Global optimization by Basin-Hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. J. Phys. Chem. A 101 (1997)
21. Yan, A.K., Langmead, C.J., Donald, B.R.: A Probability-Based similarity measure for saupe alignment tensors with applications to residual dipolar couplings in NMR structural biology. Int. J. Robot. Res. 24(2-3), 162–182 (2005)
22. Yershova, A., LaValle, S.M.: Deterministic sampling methods for spheres and SO(3). In: Proc. IEEE International Conference on Robotics and Automation (2004)
23. Zyczkowski, K., Kus, M.: Random unitary matrices. J. Phys. A 27(12), 4235–4245 (1994)

A Simple Method for Computing Minkowski Sum Boundary in 3D Using Collision Detection

Jyh-Ming Lien

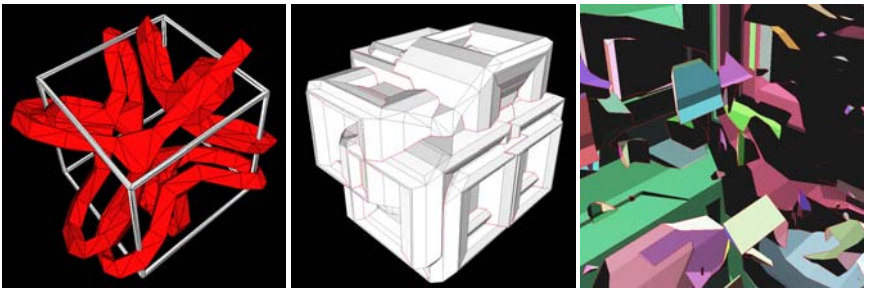
Abstract. Computing the Minkowski sum of two polyhedra exactly has been shown difficult. Despite its fundamental role in many geometric problems in robotics, to the best of our knowledge, no 3-d Minkowski sum software for general polyhedra is available to the public. One of the main reasons is the difficulty of implementing the existing methods. There are two main approaches for computing Minkowski sums: divide-and-conquer and convolution. The first approach decomposes the input polyhedra into convex pieces, computes the Minkowski sums between a pair of convex pieces, and unites all the pairwise Minkowski sums. Although conceptually simple, the major problems of this approach include: (1) The size of the decomposition and the pairwise Minkowski sums can be extremely large and (2) robustly computing the union of a large number of components can be very tricky. On the other hand, convolving two polyhedra can be done more efficiently. The resulting convolution is a superset of the Minkowski sum boundary. For non-convex inputs, filtering or trimming is needed. This usually involves computing (1) the arrangement of the convolution and its substructures and (2) the winding numbers for the arrangement subdivisions. Both computations are difficult to implement robustly in 3-d. In this paper we present a new approach that is simple to implement and can efficiently generate accurate Minkowski sum boundary. Our method is convolution based but it avoids computing the 3-d arrangement and the winding numbers. The premise of our method is to reduce the trimming problem to the problems of computing 2-d arrangements and collision detection, which are much better understood in the literature. To maintain the simplicity, we intentionally sacrifice the exactness. While our method generates exact solutions in most cases, it does not produce low dimensional boundaries, e.g., boundaries enclosing zero volume. We classify our method as 'nearly exact' to distinguish it from the exact and approximate methods.

1 Introduction

Given two geometric models and their configurations in the space, such as the knot and the frame models shown in Fig. 1(a), there are several important questions that

Jyh-Ming Lien
George Mason University, 4400 University Dr., Fairfax VA, 22030, USA
e-mail: jmlien@cs.gmu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 401–415.
springerlink.com



(a) (b) (c)

Fig. 1. Can you find a configuration that keeps the knot (in red) interlocked but without colliding with the cubic frame (in white) in the figure (a) above? Although it seems, from an external view (b), the Minkowski sum boundary of the knot and the frame models is simple, the inside view (c) shows that the Minkowski sum contains many holes. By placing the knot's reference point in one of these holes, the knot remains interlocked and collision free with the frame. There are in total 10 510 facets in this Minkowski sum boundary.

we can ask about these two models. For example, what is their shortest separation distance? Is it possible to physically separate the knot and the frame without intersections? If not, can we modify the knot, e.g., make the knot thinner, so the problem above becomes solvable? What are the set of the collision-free configurations that makes the knot and the frame interlocked? The answers to these questions play central and fundamental roles in algorithmic robotics, such as motion planning, penetration depth estimation, and object containment. However, all these questions are not easy to answer either visually or computationally due to the geometrical and topological complexity of the problem. In fact, these questions are all closely related to the concept of set sum (also known as the Minkowski sum). The Minkowski sum of two polyhedra P and Q is defined as:

$$P \oplus Q = \{p + q \mid p \in P, q \in Q\}. \tag{1}$$

In Fig. 1(b) and Fig. 1(c), we show the Minkowski sum of the knot and the frame. The inner view reveals a large number of holes in their Minkowski sum despite the simplicity of the input models. Indeed, computing the Minkowski sum of non-convex polyhedra can have the time complexity as high as $O(n^3m^3)$ [12], where m and n are the complexity of the input models.

Given two polyhedral models P and Q represented by their boundaries ∂P and ∂Q, the *boundary* of their Minkowski sum $\partial(P \oplus Q) \neq \partial P \oplus \partial Q$. Therefore, computing the boundary-based representation of the Minkowski sums is more than applying Eq. 1 to P and Q. Many methods have been proposed during the last three decades. Even though several methods [13, 4, 8, 5] are known to compute the Minkowski sum of *convex* polyhedra efficiently in 3-dimensions, most approaches proposed for general polyhedra remain in theoretical stage. Only a few practical implementations exist and none of them are available to the public. We will provide a more detailed review on the related work in Section 2.

Our approach. An important goal of our work is to provide a simple method that can efficiently and accurately compute the Minkowski sum boundaries. The proposed method is based on *convolution*. The convolution of two polyhedra P and Q is a set of facets in 3-d that is generated by 'combining' the facets of P and Q and forms a superset of the Minkowski sum boundary of P and Q. Convolution will be defined more carefully in Sections 2 and 3.1.

Briefly, our method first generates the convolution and computes the facet-facet intersections within the convolution. These intersections then induce an arrangement of line segments embedded on each facet. The cells from all the (2-d) arrangements are then merged into 'simple regions' (defined in Section 3.4), which are then filtered so that only the regions on the boundary are kept. We deliberately avoid computing the 3-d arrangement and the winding numbers, which have been shown difficult to compute robustly. Our method is designed to tolerate inaccuracy in the convolution and depends only on solving the problems of 2-d arrangement and collision detection, which are much better understood in the literature. We will discuss the details of our method in Section 3.

Our method does not solve the problem of 3-d Minkowski sum entirely. The simplicity of our method is gained by sacrificing the exactness. That is our method provides only *nearly exact* Minkowski sum whose low dimensional boundaries, e.g., boundaries enclosing zero volume, are *not* identified. Fortunately, when P and Q do not interlock too tightly, the proposed method keeps all boundaries exact (although may still suffer from numerical errors), thus provides more accuracy than the approximate methods [19, 14] do. We should also point out that our method shares some similarity with our previous work on the point-based method [14]. Beside the difference in their representations (mesh vs. points), the proposed method provides significant improvements over the point-based method in terms of both quality and efficiency. We will carefully compare these two approaches in Section 4.

2 Related Work

During the last three decades, many methods have been proposed to compute the Minkowski sums of polygons or polyhedra; see more detailed surveys in [6, 19, 4] for the Minkowski sums of the models in boundary-based representation. Despite the large volume of work, most methods can be categorized into one of the two main frameworks: divide-and-conquer and convolution.

Divide-and-Conquer. In the divide-and-conquer framework, the input models are decomposed into components. Because computing the Minkowski sum of convex shapes is easier than non-convex shapes, convex decomposition (either surface or solid) is widely used. The next step in this framework computes the pairwise Minkowski sums of the components. Finally, all these pairwise Minkowski sums are united to form the final Minkowski sum of the input shapes.

This approach is first proposed by Lozano-Pérez [16] to compute $\mathscr{C}$-obst for motion planning. Although the main idea of this approach is simple, the divide step (i.e., convex decomposition) and the merge step (i.e., union) can be very difficult

to implement robustly in practice, in particular when the input shapes are complex. For example, it is known that creating solid convex decomposition robustly is difficult, e.g., it is necessary to maintain the 2-manifold property after the split [2]. In addition, Agarwal et al. [1] have shown that different decomposition strategies can greatly affect the efficiency of this approach. Hachenberger [11] presents a robust and exact implementation using the Nef polyhedra in CGAL. However, his results are still limited to simple models.

The union step is even more troublesome. The decomposition step normally generates many components. Even though methods exist to perform union operation, no existing methods can robustly compute the union of thousands even millions of pairwise Minkowski sums. In particular, the size and the complexity of the geometry generated during the intermediate steps can be overwhelming. Flato [3] computes the unions using the cells induced by the arrangement of the line segments. He uses a hybrid strategy that combines arrangement with incremental insertion to gain better efficiency. Hachenberger [11] also studies how the order of the union operation affects the efficiency. To avoid this explicit union step, Varadhan and Manocha [19] proposed an approach that generates meshes approximating the Minkowski sum boundary using marching cube technique to extract the iso-surface from a signed distance field. They proposed an adaptive cell to improve the robustness and efficiency of their method. Because their approach still depends on convex decomposition, it still suffers from the excessive number of convex components from decomposition.

Convolution. The convolution of two shapes P and Q, denoted as $P \times Q$, is a set of line segments in 2-d or facets in 3-d that is generated by 'combining' the segments or the facets of P and Q [9]. One can think of the convolution as the Minkowski sum that involves only the boundary, i.e., $P \times Q = \partial P \oplus \partial Q$. It is known that the convolution forms a superset of their Minkowski sum [6], i.e., $\partial(P \oplus Q) \subset P \times Q$. To obtain the Minkowski sum boundary, it is necessary to trim the line segments or the facets of the convolution.

For 2-d polygons, Guibas and Seidel [10] show an output sensitive method to compute convolution curves. Later, Ghosh [6] proposed an approach, which unifies 2-d and 3-d, convex and non-convex, and Minkowski addition and decomposition operations. The main idea in his method is the negative shape and slope diagram. Slope diagram is closely related to *Gaussian map*, which has been recently used to compute to implement robust and efficient Minkowski sum computation of convex objects by Fogel and Halperin [4]. Kaul and Rossignac [13] proposed a linear time method to generate a set of Minkowski sum facets. Output sensitive methods that compute the Minkowski sum of polytopes in d-dimension have also been proposed by Gritzmann and Sturmfels [8] and Fukuda [5].

The main difficulty of the convolution-based methods is to remove the portion of the facets that are inside the Minkowski sum. Recently, Wein [20] shows a robust and exact method based on convolution for non-convex polygons. To obtain the Minkowski sum boundary from the convolution, his method computes the arrangement induced by the line segments of the convolution and keeps the cells with

non-zero winding numbers. No practical implementation is known for polyhedra using convolution due to the difficulty of computing the 3-d arrangement and its substructures [18].

Point-Based Representation. Alternatively, points have been used to represent the Minkowski sum boundary. Representing the boundary using only points has many benefits. First of all, generating such points is easier than generating meshes and can be done in parallel and in multi-resolution fashion. Moreover, point-based representation can be generalized to higher dimensional motion planning problems [15].

Peternell et al. [17] proposed a method to compute the Minkowski sum of two solids using points densely sampled from the solids, and compute local quadratic approximations of these points. However, their method only identifies the outer boundary of the Minkowski sum using a regular grid, i.e., no hole boundaries are identified. This can be a serious problem in particular when we study problems in motion planning and penetration depth computation.

We proposed a completely different method [14] that guarantees to produce a point set *covering* the boundary. However, our method also has drawbacks. For example, a large number of points are required if the Minkowski sum has small features (e.g., the models in Fig. 9). In addition, our method treats each point independently. This is good for the purpose of parallelization but the local relationship between the neighboring points is completely ignored. The method proposed in this paper does not suffer from these problems.

3 Our Method

In this section, we begin to discuss more details about the proposed method. The proposed method is convolution based and comprises five main steps. Our method first computes the convolution using a brute force method (Section 3.1). Then, we identify all the intersecting facets in the convolution (Section 3.2). Next, each facet is subdivided into sub-facets from the facet-facet intersections (Section 3.3). All sub-facets are stitched into *simple regions* based on the properties that will be discussed later (in Section 3.4). A simple region is either entirely inside or entirely on the boundary of the Minkowski sum. Finally, we use a collision detector to remove the regions inside the Minkowski sum (Section 3.5). We conclude this section by providing a discussion on the benefits provided by the proposed method and its current limitations (Section 3.6).

3.1 Brute Force Convolution

We use a brute force method to compute the convolution because of its simplicity. As we will see in our experiments, the convolution step actually takes very little time (on average 0.4% of the entire computation), even using the brute force method, comparing to all the other steps.

Our brute force method checks all possible facet/vertex and edge/edge pairs of P and Q and keeps all the facets that satisfy the criteria stated in Observation 3.1. The result of the brute force convolution is a set of facets that reside in the interior and on the boundary of the Minkowski sum.

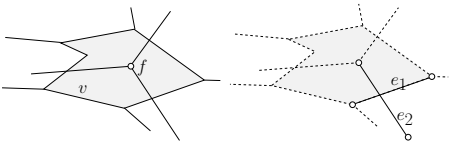


Fig. 2. Gaussian map of fv- (left) and ee- (right) facets.

Given two polyhedra P and Q, the convolution of P and Q can only come from two sources [13]: (i) the facets, called fv-facets, generated from a facet of P and a vertex of Q or vice versa and (ii) the facets, called ee-facets, generated from a pair of edges from P and Q, respectively.

Observation 3.1. *A facet f and a vertex v produce a valid fv-facet if and only if the normal of f is inside the region enclosed by the normals of the facets incident to v in the Gaussian map. Similarly, a pair of edges e_1 and e_2 form an ee-facet if the corresponding edges in the Gaussian map cross each other. Fig. 2 illustrates the necessary conditions of both fv- and ee-facets.*

Our goal in the next few steps is to remove the portions of the convolution inside the Minkowski sum.

3.2 Facet-Facet Intersections

The goal of this step is to identify all the intersecting facets for each facet in the convolution. To do so, we construct a bounding volume hierarchy from top-down using spheres that enclose all the facets. For each facet, we use its bounding sphere to identify all the intersecting spheres, which contain potential intersections. Finally, the intersecting facets are then determined from all these spheres. Because all the facets generated in the convolution must be convex if the input models have only convex facets, exact facet-facet intersection can be performed efficiently in 3-d. Without the loss of generality, we assume that the models used in this paper are composed of triangles.

3.3 Split Facets

We use the intersections above to split the convolution facets. Essentially, this step computes the 2-d arrangements of the facet-facet intersections obtained from the previous step. For each facet, we project the intersections to the supporting plane of the facet. The arrangement embedded in the facet is induced by these projected line segments and the boundary of the facet. It should be noted that when the interior of a segment partially or entirely overlaps with other segments, we handle this degenerate case by creating cells with zero areas enclosed by the overlapped segments. As we will see later, these 'area-less' regions also serve as a form of 'insulator' to prevent the facets from being stitched.

For the facet without any intersections, we simply treat it as an arrangement with a single cell (two cells if we count the unbounded subdivision). To simplify our discussion, we call a cell created in this step a 'sub-facet.'

3.4 Stitch Sub-facets

Our goal in this step is to stitch all the sub-facets into *simple regions*. A simple region is composed of a set of contiguous sub-facets that are completely on the Minkowski sum boundary or are completely inside the Minkowski sum. Our method constructs the simple regions by stitching the neighboring sub-facets iteratively until all sub-facets are stitched. We say that two sub-facets f_1 and f_2 are neighbors if they share an edge.

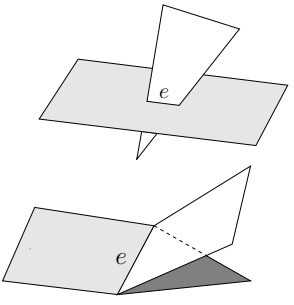


Fig. 3. Examples of facets that cannot be stitched.

Stitching criteria. Let C be an existing component and let f_1 be a facet on the boundary of C. We further let f_2 be a neighbor of f_1 that is not in C and let e_{12} be the edge shared by f_1 and f_2. Then f_1 and f_2 are stitched if they do not violate the following constraints.

1. e_{12} does not overlap with the intersections of the *interior* of the convolution facets.
2. e_{12} is 2-manifold.

Note that the first constraint can be readily checked from the intersection step earlier and is in fact a special case of the second constraint. This is because a pair of intersecting facets must generate a non-manifold region. The second constraint is used to check for non-manifold edges shared by more than two the adjacent (non-intersecting) sub-facets. Fig. 3 shows two examples that violate these criteria.

3.5 Determine the Boundary Regions

In this final step, we determine which simple regions are non-boundary regions and should be discarded using collision detection calls. Our method uses the close relationship between the Minkowski sum boundary and the concept of "contact space" in robotics. Every point in the contact space represents a configuration that places the robot in contact with (but without colliding with) the obstacles. Given a translational robot P and obstacles Q, the contact space of P and Q can be represented as $\partial((-P) \oplus Q)$, where $-P = \{-p \mid p \in P\}$. In other words, if a point x is on the boundary of the Minkowski sum of two polyhedra P and Q, then the following condition must be true:

$$(-P^\circ + x) \cap Q^\circ = \emptyset \,,$$

where Q° is the open set of Q and $(P + x)$ denotes translating P to x.

Using this observation, we can determine if a simple region R is on the boundary by simply placing $-P$ at a point x sampled from a facet $f \in R$ and testing if $(-P+x)$ is in collision with Q. If $(-P+x)$ is collision free, then we can conclude that R is on the Minkowski sum boundary. Otherwise, we discard R.

3.6 Discussion and Implementation Details

The proposed method is simple and efficient, but it does not produce low dimensional (isolated) boundaries composed of only edges and vertices. In this section, we provide more detailed discussion regarding the implementation and the advantages and the limitations (and possible improvements) in some steps of the proposed method. The readers can also skip these details and go to Section 4 for experimental results.

Convolution. Our brute-force method does not compute exact 3-d convolutions, but a superset of the convolution. As far as we know, no practical method can compute the convolution of polyhedra exactly and robustly, even though methods exist to compute the convolution of polygons, such as the techniques in [10, 20]. Our method, unlike [20, 10], does not use the (mesh) connectivity of P and Q to construct the convolution, and, due to numerical errors, may generate 'isolated' facets in the final 'convolution' instead of a set of closed 2-manifolds. Note that all the isolated facets are inside the Minkowski sum boundary.

These two weaknesses of our brute-force method make the computations of the arrangement and the winding numbers even more difficult. However, because we intentionally avoid these two steps, our method does not suffer from the inaccuracy.

Given two polyhedra P and Q with size m and n, the brute-force method takes $O(mn)$ time. As we mentioned earlier, the convolution step is not the bottleneck of the entire computation. Even though computing the convolution from the nonplanar Gaussian maps using a strategy similar to the ideas in [10, 20] can definitely increase the efficiency, the improvement to the entire computation is limited.

Facet-facet intersection. We use bounding sphere hierarchy to detect the intersections. We use spheres because they are invariant under rotation. This step takes $O((N+k)\log N)$ time, where $N = mn$ is the size of the convolution and k is the intersection size.

Stitch sub-facets. The idea of stitching is to maintain a set of the largest 2-manifolds from the convolution. We claim that each of this 2-manifold is a simple region. The criteria proposed to construct the simple regions (in Section 3.4) also focus this goal. In Lemma 3.1, we show that these two criteria is indeed sufficient to generate simple regions.

Lemma 3.1. *A simple region is either on the Minkowski sum boundary or in the interior of the Minkowski sum if the simple region is constructed using the criteria in Section 3.4.*

Proof (Sketch). Let C be the convolution of two polyhedra and let $A(C)$ be the arrangement of C. Essentially, a simple region identified in Section 3.4 is a set of contiguous sub-facets that form or entirely reside on the boundary shared by two

(3-d) cells of $C(A)$. Since a cell must not cross the Minkowski sum boundary, the simple region will not cross the boundary. Thus, a simple region is either on the boundary or in the interior of the Minkowski sum.

Given the strong connection between the simple region and the arrangement cell, one might wonder if we can further stitch the simple regions into cells. There are several reasons that we do not go in this direction. First, given x cells in the arrangement of the convolution, there can be $O(x)$ simple regions, Therefore, further stitching regions into cells may not improve the efficiency (at least asymptotically). Second, this additional step greatly increases the difficulty of the implementation. Many degenerated cases, in particular with isolated regions, should be considered. In addition, from our preliminary results, little or no performance is gained by stitching further. Due to these reasons, we do not further stitch simple regions into arrangement cells.

Determine the boundary regions. We use collision detection calls to determine the type of a simple region. For detecting collisions, we use a modified version of RAPID [7]. An issue that we have to deal with when working with RAPID (and most collision detectors) is that RAPID cannot distinguish if two objects are in the contact configurations or are in fact in collision. To work around this problem, we perturb each point we sampled with an infinitesimally small vector pointing in the outward direction of the facet (from the convolution) where the point is sampled from. Note that the normal directions of all fv- and ee-facets are readily available from the convolution step.

After the perturbation, the point will most likely become collision free if it is indeed on the Minkowski sum boundary. The exceptions to the above case occur when the Minkowski sum boundary degenerates to an isolated vertex, edge or sliver (enclosing zero or a very small volume). This is the reason why our method provides only 'nearly' exact Minkowski sum.

Another concern of using collision detection to replace winding number is the efficiency. However, as our experiment shows, collision detection, although dominates the computation in some examples, does not significantly slow down our method.

4 Experimental Results

In this section, we show experimental results. All the experiments are performed on a PC with two Intel Core 2 CPUs at 2.13 GHz with 4 GB RAM. Our implementation is coded in C++. For detecting collisions, we use a modified RAPID [7]. Fig. 4 shows a set of models used in this section. All the models and the Minkowski sum boundaries in our experiments are in Wavefront OBJ format and can be downloaded from our project webpage.

4.1 Geometric Modeling

Our method can be used to perform operations such as offsetting, erosion, and sweeping on large geometric models. Fig. 5 shows an example of the offsetting

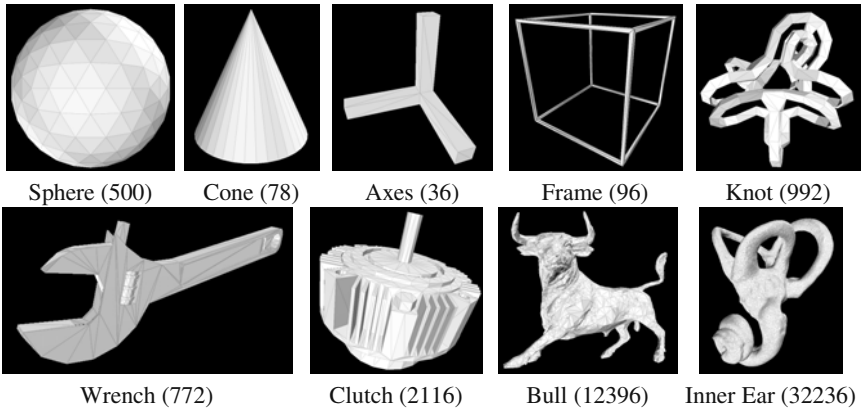


Fig. 4. Models used in this paper. The number following the model name is the number of facets of the model.

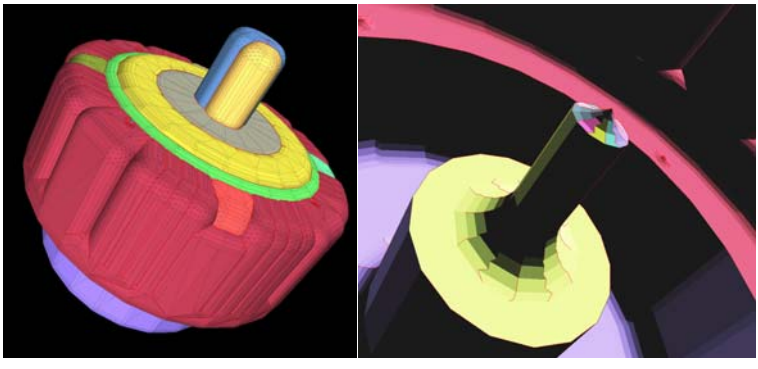
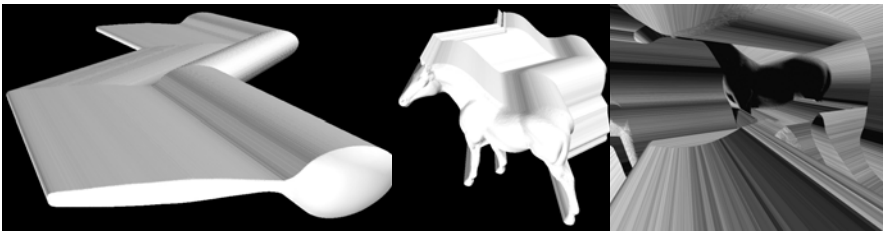


Fig. 5. Offsetting of the clutch model.

operation of the clutch model. Offsetting is done by computing its Minkowski sum with a sphere. The top figure of Fig. 5 shows the Minkowski sum boundary (13 974 facets) of the clutch model and the sphere model. Each colored patch (best viewed from the submitted *pdf* file) on the Minkowski sum boundary indicates a *simple region* bounded by red line segments. Interestingly, for some models, the red line segments that separate simple regions tend to go through the areas with high concavity. Therefore, the simple regions seem to represent visually meaningful segmentations of the model. The bottom figure of Fig. 5 shows an internal view of the Minkowski sum.

In Fig. 6, we show an example of the swept volumes of two large models: a spoon and a horse. The swept volume is generated by computing the Minkowski sum of the spoon and the horse models with a thin tube representing a trajectory. An internal view of the horse model's swept volume is also shown.



(a) (b) (c)

Fig. 6. (a) A swept volume of the spoon model (89 822 facets). The boundary is composed of 138 801 facets. (b) A swept volume of the horse model (39 694 facets). The boundary is composed of 73 912 facets. (c) An internal view of (b).

4.2 Computation Time

A step-by-step analysis. Fig. 7 shows our first experiment result using the models in Fig. 4, which include convex/non-convex models, zero and non-zero genus models, and CAD and digitized models. These models are selected carefully to test the

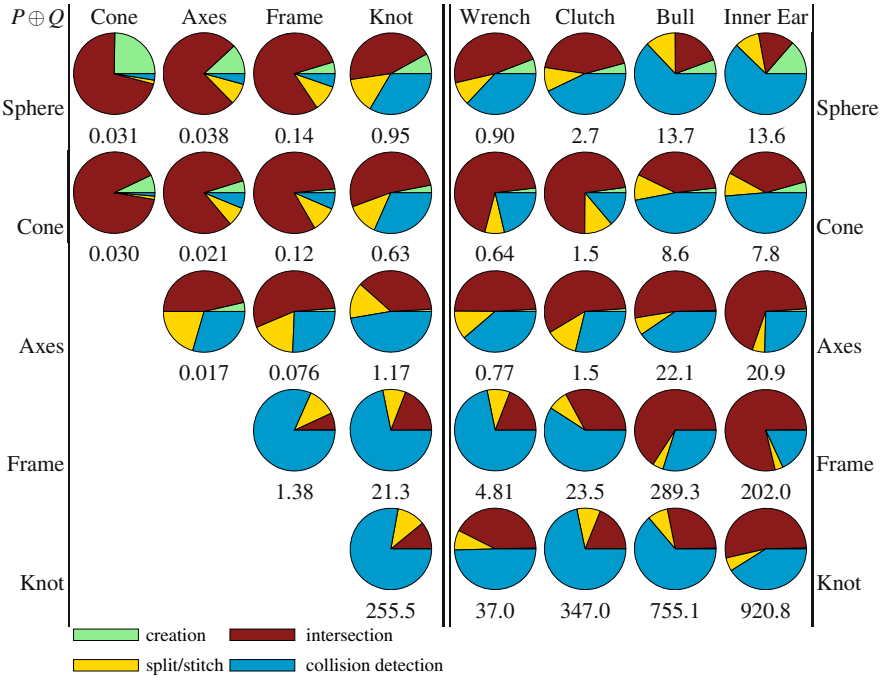


Fig. 7. Computation time of the proposed method. Each Minkowski sum computation is shown as a pie chart, representing the cost of each step, and a number below the chart, representing the total computation time (in seconds). Models used in this experiment can be found in Fig. 4.

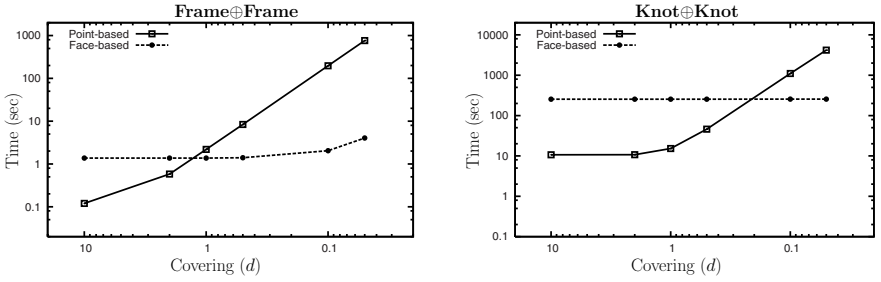


Fig. 8. Computation time for generating points covering the Minkowski sum boundaries. Notice that the x and y axes are both in logarithmic scale.

proposed method. In Fig. 7, we show the computation time of each Minkowski sum and the ratio of each step in an entire Minkowski sum computation. It is clear that the facet-facet intersection and collision detection steps dominate the computation. We observe that the ratio of the creation time decreases when the size of the model increases. When the size of the model increases, the intersection step becomes more dominating. When the models have handles, the ratio of the collision detection increases due to the increasing number of holes (e.g., Frame and Knot).

Point-based vs. Mesh-based Minkowski sum. We compare the proposed method (hereafter named mesh-based method) to the point-based Minkowski sum [14] since it is the only implementation available to the public that supports general polyhedra. In order to make fair comparisons, we sample points from the facets generated by the mesh-based method. Like point-based Minkowski sum, these points form a d-covering[1] of the Minkowski sum boundary. It is obvious that when d is large point-based method can outperform mesh-based method. In Fig. 8, we vary the value of d from 10 to 0.05. As we can see that, as the value of d decreases, the computation time of the mesh-based method is slightly elevated while the collision detection call number remain the same. On the other hand, the point-based method slows down significantly as d decreases due to rapidly increasing detection calls.

In addition to the benefit of being faster than the point-based method, the mesh-based method proposed in this paper does not suffer from the sampling density issues. In particular, when small features are present in the Minkowski sum boundary, high density points (i.e., small d) are needed to reveal these features. In Fig. 9, we show a 'classic' example of two grate-like shapes, from which a large number points will need to be sampled in order to capture the long and skinny columns of the Minkowski sum boundary. Our mesh-based method does not suffer from this problem and successfully generates the exact Minkowski sum boundary.

[1] We say that a set of points S is a d-covering of a surface M if, for every point m of M, there exists a point in S whose distance to m is less than d.

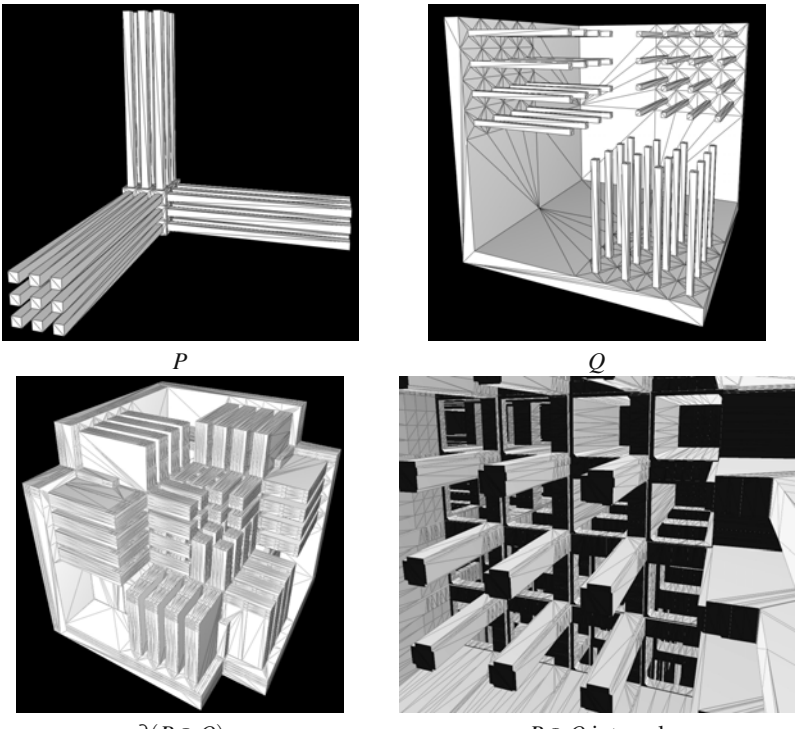


Fig. 9. Minkowski sum of two grate-like models. P has 27 teeth and 540 facets, and Q has 48 teeth and 942 facets, and $P \oplus Q$ has 71043 facets. The total computation time is 318.5 seconds using 1 thread. These models imitate the grate models created by Halperin [12] and from Varadhan and Manocha [19].

5 Conclusion

In this paper we proposed a simple 3-d Minkowski sum method. In essence, our idea is to avoid computing the exact convolution, 3-d arrangement and the winding numbers. Instead, we filter and trim facets using only 2-d arrangements and collision detector. Our method starts with an inaccurate convolution generated by a brute force method. For each facet in the convolution, we subdivide the facet into sub-facets induced by the arrangement of the facet-facet intersections within the convolution. Sub-facets are then grouped into simple regions, which are filtered by a collision detector. Our method does not solve the problem of 3-d Minkowski sum entirely. The simplicity of our method is gained by sacrificing the exactness. Although providing only *nearly* exact Minkowski sum, our method is more accurate than the approximate methods. In our experiment, we demonstrated the proposed method's ability of handling large geometric models. We also showed the efficiency of the proposed method comparing to the point-based Minkowski sum method. While we

are currently optimizing the performance of our implementation, we plan to release the software developed for this paper to the public.

Acknowledgements

The geometric models used in this paper are from several sources. The sphere model is from Efi Fogel and Dan Halperin. The knot model is from Ergun Akleman. The wrench and clutch models are from the GAMMA group at UNC-Chapel Hill. The "dancing children" model is provided courtesy of IMATI-GE by the AIMSHAPE Shape Repository.

References

1. Agarwal, P.K., Flato, E., Halperin, D.: Polygon decomposition for efficient construction of minkowski sums. In: Paterson, M. (ed.) ESA 2000. LNCS, vol. 1879, pp. 20–31. Springer, Heidelberg (2000)
2. Bajaj, C., Dey, T.K.: Convex decomposition of polyhedra and robustness. SIAM J. Comput. 21, 339–364 (1992)
3. Flato, E.: Robuts and efficient construction of planar minkowski sums. M.Sc. thesis, Dept. Comput. Sci., Tel-Aviv Univ., Israel (2000)
4. Fogel, E., Halperin, D.: Exact and efficient construction of Minkowski sums of convex polyhedra with applications. In: Proc. 8th Wrkshp. Alg. Eng. Exper. (Alenex 2006), pp. 3–15 (2006)
5. Fukuda, K.: From the zonotope construction to the minkowski addition of convex polytopes. Journal of Symbolic Computation 38(4), 1261–1272 (2004)
6. Ghosh, P.K.: A unified computational framework for Minkowski operations. Computers and Graphics 17(4), 357–378 (1993)
7. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree: A hierarchical structure for rapid interference detection. Computer Graphics 30(Annual Conference Series), 171–180 (1996)
8. Gritzmann, P., Sturmfels, B.: Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. SIAM J. Discret. Math. 6(2), 246–269 (1993)
9. Guibas, L.J., Ramshaw, L., Stolfi, J.: A kinetic framework for computational geometry. In: Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci., pp. 100–111 (1983)
10. Guibas, L.J., Seidel, R.: Computing convolutions by reciprocal search. Discrete Comput. Geom. 2, 175–193 (1987)
11. Hachenberger, P.: Exact Minkowksi Sums of Polyhedra and Exact and Efficient Decomposition of Polyhedra in Convex Pieces. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 669–680. Springer, Heidelberg (2007)
12. Halperin, D.: Robust geometric computing in motion. The International Journal of Robotics Research 21(3), 219–232 (2002)
13. Kaul, A., Rossignac, J.: Solid-interpolating deformations: construction and animation of PIPs. In: Proc. Eurographics 1991, pp. 493–505 (1991)
14. Lien, J.-M.: Point-based minkowski sum boundary. In: PG 2007: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, Washington, DC, USA, pp. 261–270. IEEE Computer Society, Los Alamitos (2007)
15. Lien, J.-M.: Hybrid motion planning using Minkowski sums. In: Proc. Robotics: Sci. Sys. (RSS), Zurich, Switzerland (2008)

16. Lozano-Pérez, T.: Spatial planning: A configuration space approach. IEEE Trans. Comput. C-32, 108–120 (1983)
17. Peternell, M., Pottmann, H., Steiner, T.: Minkowski sum boundary surfaces of 3d-objects. Technical report, Vienna Univ. of Technology (August 2005)
18. Raab, S.: Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes. In: SCG 1999: Proceedings of the fifteenth annual symposium on Computational geometry, pp. 163–172. ACM, New York (1999)
19. Varadhan, G., Manocha, D.: Accurate Minkowski sum approximation of polyhedral models. Graph. Models 68(4), 343–355 (2006)
20. Wein, R.: Exact and efficient construction of planar Minkowski sums using the convolution method. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 829–840. Springer, Heidelberg (2006)

Polyhedral Assembly Partitioning with Infinite Translations or *The Importance of Being Exact*⋆

Efi Fogel and Dan Halperin

Abstract. Assembly partitioning with an infinite translation is the application of an infinite translation to partition an assembled product into two complementing subsets of parts, referred to as a subassemblies, each treated as a rigid body. We present an exact implementation of an efficient algorithm to obtain such a motion and subassemblies given an assembly of polyhedra in $\mathbb{R}^3$. We do not assume general position. Namely, we handle degenerate input, and produce exact results. As often occurs, motions that partition a given assembly or subassembly might be isolated in the infinite space of motions. Any perturbation of the input or of intermediate results, caused by, for example, imprecision, might result with dismissal of valid partitioning-motions. In the extreme case, where there is only a finite number of valid partitioning-motions, no motion may be found, even though such exists. The implementation is based on software components that have been developed and introduced only recently. They paved the way to a complete, efficient, and concise implementation. Additional information is available at `http://acg.cs.tau.ac.il/projects/internal-projects/assembly-partitioning/project-page`.

1 Introduction

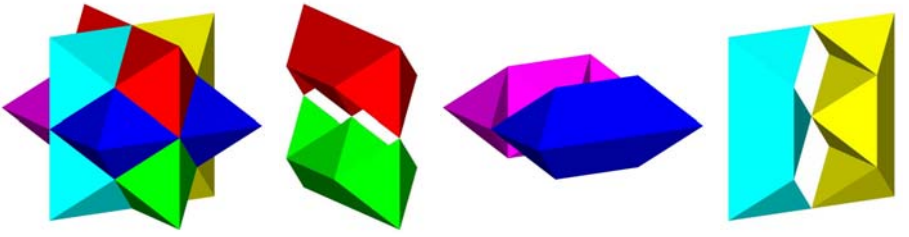
Assembly planning is the problem of finding a sequence of motions that move the initially separated parts of an assembly to form the assembled product. The reversed

Efi Fogel
Tel-Aviv University, Tel Aviv 69978, Israel
e-mail: efif@post.tau.ac.il

Dan Halperin
Tel-Aviv University, Tel Aviv 69978, Israel
e-mail: danha@post.tau.ac.il

⋆ This work has been supported in part by the Israel Science Foundation (grant no. 236/06), by the German-Israeli Foundation (grant no. 969/07), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 417–432.
springerlink.com



(a) (b) (c) (d)

Fig. 1. (a) The Split Star puzzle, and (b),(c), and (d) the Split Star six parts divided into three pairs of symmetric parts. The six parts are named according to their color *R*(ed), *G*(reen), *B*(lue), *P*(urple), *Y*(ellow), and *T*(urquoise).

order of sequenced motions separates an assembled product to its parts. Thus, for rigid parts, assembly planning and disassembly planning refer to the same problem, and used interchangeably. In this paper we concentrate on the case where the assembly consists of polyhedra in $\mathbb{R}^3$ and the motions are infinite translations.

The motion space is the space of possible motions that assembly parts may undergo. For each motion in a motion space, a subassembly of a given assembly may collide with a different subassembly, when transformed as a rigid body according to the motion. Pairs of subassemblies that collide constitute constraints. The motion space approach dictates the precomputation of a decomposition of a motion space into regions, such that the constraints among the parts in the assembly are the same for all the motions in the same region. All constraints over a region are represented by a graph, called the *directional blocking graph* (DBG) [23]. The collection of all regions in a motion space together with their associated DBGs can be used to obtain assembly (or disassembly) sequences.

Degenerate input is commonplace in computational-geometry applications, and numerical errors are inevitable when arithmetic based on machine number-type (e.g., floating-point) is used to carry out algebraic computation. Traditional implementations, which ignore these observations, may yield incorrect results, get into an infinite loop, or just crash, while running on a degenerate, or nearly degenerate, input (see [15] for examples). The problem intensifies in assembly planning, as motions might be isolated in the infinite space of motions. Any perturbation of the input or of intermediate results, caused by, for example, imprecision, might result with dismissal of valid partitioning-motions. In the extreme case, where there is only a finite number of valid partitioning-motions, as occurs in the assembly shown in Figure 1, no motion may be found, even though such exists.

The general framework and some of the techniques presented here have already been described in a series of papers and reports published in the past mainly during the late 90's. Halperin, Latombe, and Wilson made the connection between previously presented techniques that had used the motion space approach, and introduced a unified general framework [11] at the end of the previous millennium. Only few

publications related to this topic appeared ever since, to the best of our knowledge, which creates a long gap in the time line of the respective research. We certainly hope that the tools exposed in this paper will help revive the research on algorithmic assembly planing, a research subject of considerable importance. Moreover, we believe that the machinery presented here, together with other recent advances in the practice of computational-geometry algorithms, can more generally support the development of new and improved techniques in *algorithmic automation*.[1]

Solution to the assembly planning problem enables better feedback to designers. It helps them to create products that are more cost-effective to manufacture and maintain. This is emphasized in light of the strategy to "plan anywhere, build anywhere" many CAD/CAM companies are trying to adopt. Assembly sequences are also useful for selecting assembly tools and equipment, and for laying out manufacturing facilities.

We restrict ourselves to two-handed partitioning steps, meaning that we partition the given assembly into two complementing subsets each treated as a rigid body. Even for two-handed partitioning, if we allow arbitrary translational motions (and not restrict ourselves to infinite translations) the problem becomes NP-hard [12]. The assembly-sequencing general problem of planning a total ordering of assembly operations that merge the individual parts into the assembled product, is PSPACE-hard, even when each part is a polygon with a constant number of vertices [18].

Notice that the problem that we address in this paper, namely partitioning with *infinite translations*, is technically considerably more complex than partitioning with *infinitesimal motions*. Although the latter may sound more general, as it handles infinitesimal translations and *rotations*, it is far simpler to implement, since it deals only with constraints that can be described linearly. Thus, the problem can be reduced to solving linear programs. Indeed, there are several implementations for partitioning with infinitesimal motions (see, e.g., [9, 19]), but none that we are aware of, dealing robustly with infinite translations. The shortcoming of using infinitesimal motions only is that suggested disassembly moves may not be extendible to practical finite-length separation motions.

Infinite-translation partitioning was not fully robustly implemented until recently, in spite of being more useful than infinitesimal partitioning, most probably due to the hardship of accurately constructing the underlying geometric primitives. What enables the solution that we present here, is the significant headway in the development of computational-geometry software over the past decade, the availability of stable code in the form of the Computational Geometry Algorithms Library (CGAL)[2] in general and code for Minkowski sums of polytopes in $\mathbb{R}^3$ and arrangements in $\mathbb{R}^2$ in particular [22].

The implementation presented in this paper is based on a package of CGAL called `Arrangement_on_surface_2` [21]. It supports the robust construction and maintenance of arrangements of curves embedded on two-dimensional parametric

[1] `http://goldberg.berkeley.edu/algorithmic-automation`
[2] `http://www.cgal.org`

surfaces [2], and robust operations on them, e.g., overlay computation. The implementation uses in particular arrangements of geodesic arcs embedded on the sphere. It exploits supported operations, and requires additional operations, e.g., polyhedra central projection, which we implemented. We plan to make these new components available as part of a future public release of CGAL as well. The ability to robustly construct such arrangements, and carry out exact operations on them using only (exact) rational arithmetic is a key property that enables an efficient implementation.

1.1 Split Star Puzzle

We use the assembly depicted in Figure 1 as a running example throughout the paper. The name "Split Star" was given to this shape by Stewart Coffin in one of his Puzzle Craft booklet editions back in 1985. He uses the term *puzzle* to refer to any sort of geometric recreation having pieces that come apart and fit back together. We use it as an assembly. He describes how to produce the actual pieces out of wood [4], and suggests that they are made very accurately. He observes that finding the solution requires dexterity and patience, when the pieces are accurately made with a tight fit. Even though the assembly seems relatively simple, this should come as little surprise, since the first partitioning motion is one out of only eight possible translations of four symmetric pairs of motions in opposite directions associated with two complementing subassemblies of three parts each. Evidently, any automatic process that introduces even the slightest error along the way, will most likely fail to compute the correct first motion in the sequence, and falsely claim that the assembly is interlocked.

The Split Star assembly has the assembled shape of the first stellation of the rhombic dodecahedron [17], illustrated atop the right pedestal in M. C. Escher's Waterfall woodcut [3]. Its orthogonal projection along one of its fourfold axes of symmetry is a square, while the Star of David is obtained when it is projected along one of its threefold axes of symmetry, as seen on the left; for more details see [4]. The assembly is a space-filling solid when assembled. It consists of six identical concave parts. Each part can be decomposed into eight identical tetrahedra yielding 48 tetrahedra in total. As manufacturing the pieces requires extreme precision, it is suggested to produce the 48 identical pieces and glue them as necessary. Each part can also be decomposed into three convex polytopes — two square pyramids and one octahedron, yielding 18 polytopes in total. The partitioning described in this paper requires the decomposition of the parts into convex pieces. The choice of decomposition may have a drastic impact on the time consumption of the entire process, as observed in a different study in $\mathbb{R}^2$ [1], and shown by experiments in Section 5.

1.2 Outline

The rest of this paper is organized as follows. The partitioning algorithm is described in Section 2 along with the necessary terms and definitions. In Section 3 we provide implementation details. Section 4 presents optimizations that are not discussed in the preceding sections, some of which we have already implemented, and proved to be useful. We report on experimental results in Section 5.

2 The Partitioning Algorithm

The main problem we address in this paper, namely, *polyhedral assembly partitioning with infinite translations*, is formally defined as follows: Let $A = \{P_1, P_2, \ldots, P_n\}$ be a set of n pairwise interior disjoint polyhedra in $\mathbb{R}^3$. A denotes the assembly that we aim to partition. We look for a proper subset $S \subset A$ and a direction $\mathbf{d}$ in $\mathbb{R}^3$, such that S can be moved as a rigid body to infinity along $\mathbf{d}$ without colliding with the rest of the parts of the assembly $A \setminus S$. (We allow sliding motion of one part over the other. We disallow the intersection of the interior of two polyhedra.)

We follow the NDBG approach [23], and describe it here using the general formulation and notation of [11]. The *motion space* in our case, namely the space of all possible partitioning directions, is represented by the unit sphere $\mathbb{S}^2$. Every point p on $\mathbb{S}^2$ defines the direction from the center of $\mathbb{S}^2$ towards p. Every direction $\mathbf{d}$ is associated with the directed graph $DBG(\mathbf{d}) = (V, E)$ that encodes the blocking relations between the parts in A when moved along $\mathbf{d}$ as follows: The nodes in V correspond to polyhedra in A; we denote a node corresponding to the polyhedron P_i by $v(P_i) \in V$. There is an edge directed from $v(P_i)$ to $v(P_j)$, denoted $e(P_i, P_j) \in E$, if and only if the interior of the polyhedron P_i intersects the interior of the polyhedron P_j when P_i is moved to infinity along the direction $\mathbf{d}$, and P_j remains stationary.

The key idea behind the NDBG approach is that in problems such as ours, where the number of parts is finite, and any allowable partitioning motion can be described by a small number of parameters, there is only a relatively small (polynomial) number of distinct DBGs that need to be constructed in order to detect a possible partitioning direction. Stated differently, the motion space can be represented by an arrangement subdivided into a finite number of cells each assigned with a fixed DBG. Once this arrangement is constructed, we construct the DBG over each cell of the arrangement, and check it for strong connectivity. A DBG that is not strongly connected is associated with a direction, or a set of directions in case the cell is not a singular point, that partition the given assembly. The desired movable subset $S \subset A$ is a byproduct of the algorithm that checks for strong connectivity. If all the DBGs over all the cells of the arrangement are strongly connected, we conclude that the assembly is *interlocked*, as a subset of the parts in A that can be separated from the rest of the assembly by an infinite translation does not exist.

Next we show how to construct the motion-space arrangement and compute the DBG over each one of the arrangement cells. Each ordered pair of distinct polyhedra $< P_i, P_j >$ defines a region Q_{ij} on $\mathbb{S}^2$, which is the union of all the directions $\mathbf{d}$, such that when P_i is moved along $\mathbf{d}$ its interior will intersect the interior of P_j.

How can we effectively compute this region? Let M_{ij} denote the Minkowski sum $P_j \oplus (-P_i) = \{b-a \,|\, a \in P_i, b \in P_j\}$. We claim that the central projection of M_{ij} onto $\mathbb{S}^2$ is exactly Q_{ij}.

Lemma 2.1. *A direction* $\mathbf{d}$ *is in the interior of the central projection of* M_{ij} *onto* $\mathbb{S}^2$ *if and only if when* P_i *is moved along* $\mathbf{d}$ *its interior will intersect the interior of* P_j.

Proof. Let $\mathbf{d}$ be some direction in the central projection of M_{ij} onto $\mathbb{S}^2$. In other words, there exists a point $m \in M_{ij}$, such that $m = s \cdot \mathbf{d}$, for some positive scalar s. As m is in M_{ij}, there exist two points $p_i \in P_i$ and $p_j \in P_j$, such that $m = p_j - p_i$. Thus, $p_j = p_i + s \cdot \mathbf{d}$, meaning that the point p_i intersects p_j when moved along $\mathbf{d}$. A similar argument can be used to show the other way around. □

Next, we describe how, given two polyhedra P_i and P_j, we compute the region Q_{ij}, using a robust and efficient hierarchy of building blocks, which we have developed in recent years. The existing tools that we use are (i) computing the arrangement of spherical polygons [2, 8, 7, 22], and (ii) construction of Minkowski sums of convex polytopes [2, 5, 8, 7]. We also need some extra machinery, as explained below.

Assume P_i is given as the union of a collection of (not necessarily disjoint) convex polytopes $P_1^i, P_2^i, \ldots, P_{m_i}^i$, and similarly P_j is given as the collection of convex polytopes $P_1^j, P_2^j, \ldots, P_{m_j}^j$. It is easily verified that $M_{ij} = \bigcup_{k=1,\ldots,m_i,\ell=1,\ldots,m_j} P_\ell^j \oplus (-P_k^i)$. So we compute the Minkowski sum of each pair $P_\ell^j \oplus (-P_k^i)$, and centrally project it onto $\mathbb{S}^2$. Finally, we take the union of all these projections to yield Q_{ij}.

There are several ways to effectively compute the central projection of a convex polyhedron C (one of the polytopes $M_{k\ell}^{ij} = P_\ell^j \oplus (-P_k^i)$) from the origin onto $\mathbb{S}^2$. We opted for the following. An edge e of C is called a *silhouette edge*, if the plane π through the origin and e is tangent to C at e. Namely, it intersects C in e only. We assume for now that there is no tangent plane that contains a facet of C; we relax this assumption in Section 3.5, where we provide a detailed description of the procedure. We traverse the edges of C till we find a silhouette edge e_1. One can verify that the silhouette edges from a cycle on C. We start with e_1, and search for a silhouette edge adjacent to e_1. We proceed in the same manner, till we end up discovering e_1 again. Projecting this cycle of edges onto $\mathbb{S}^2$ is straightforward.

All the boundaries of the regions Q_{ij} form an arrangement of geodesic arcs on the sphere. We traverse the motion-space arrangement in say a breadth-first fashion. For the first face we check which one of the regions Q_{ij} contain it. We construct the corresponding DBG and check it for strong connectivity. If it is not strongly connected, we stop and announce a solution as described above. Otherwise we move to an adjacent feature of the current face. During this move we may step out from a set of regions Q_{ij}, and may step into a new set of regions Q_{ij}. We update the current DBG according to the regions we left or entered, test the new DBG for strong connectivity, and so on till the traversal of all the arrangement cells is complete. Notice that it is important to visit also vertices and edges of the arrangement, since the solution may not lie in the interior of a face. Indeed, in our Split Star example, solutions are on vertices of the arrangement. Without careful exact constructions, such solutions could easily be missed.

3 Implementation Details

The implementation of the assembly-partitioning operation consists of eight phases listed below. They all exploit arrangements of geodesic arcs embedded on the sphere [2, 7] in various ways. The `Arrangement_on_surface_2` package of CGAL already supports the construction and maintenance of such arrangements, the computation of union of faces of such arrangements, the construction of Gaussian maps of polyhedra represented by such arrangements, and the computation of their Minkowski sums. It provides the ground for efficient implementation of the remaining required operations, such as central projection.

1. **Convex Decomposition**
2. **Sub-part Gaussian map construction**
3. **Sub-part Gaussian map reflection**
4. **Pairwise sub-part Minkowski sum construction**
5. **Pairwise sub-part Minkowski sum projection**
6. **Pairwise Minkowski sum projection**
7. **Motion-space construction**
8. **Motion-space processing**

The partitioning process is implemented as a free (general) function that accepts as input an ordered list of polyhedra in $\mathbb{R}^3$, which are the parts of the assembly. Each part is represented as a polyhedral mesh in $\mathbb{R}^3$. A polyhedral mesh representation consists of an array of vertices and a collection of facets, where each facet is described by an array of indices into the vertex array. We proceed with a detailed discussion of the implementation of each phase.

We deal below with various details that are typically ignored in reports on geometric algorithms (for example, under the general-position assumption). However, in assembly planning, or more generally in movable-separability problems in tight scenarios, much of the difficulty shifts exactly to these technical details and in particular to handling degeneracies. This is especially emphasized in Phases 5 and 6 (Subsections 3.5 and 3.6 respectively), but prevails throughout the entire section.

3.1 Convex Decomposition

We decompose each concave part into convex polyhedra referred to as sub-parts. The output of this phase is an ordered list of parts, where each part is an ordered list of convex sub-parts represented as polyhedral surfaces. Each polyhedral surface is maintained as a CGAL `Polyhedron_3` [14] data-structure, which consists of vertices, edges, and facets and incidence relations on them [13]. A part that is convex to start with is simply converted into an object of type `Polyhedron_3`.

A new package of CGAL that supports convex decomposition of polyhedra has been recently introduced [10], but has not become publicly available yet. As we aim for a fully automatic process, we intend to exploit such components, once they become available, and study their impact. For the time being we resorted to a

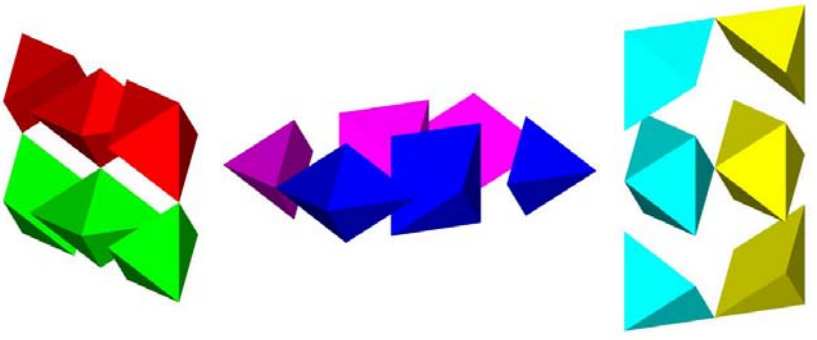


Fig. 2. The six parts of the Split Star decomposed into three convex sub-parts each.

manual procedure. A simple decomposition of the Split Star parts used in the running example is illustrated in Figure 2.

3.2 Sub-part Gaussian Map Construction

The *Gaussian Map* $G = G(P)$ of a compact convex polyhedron P in Euclidean three-dimensional space $\mathbb{R}^3$ is a set-valued function from P to the unit sphere $\mathbb{S}^2$, which assigns to each point p on the boundary of P the set of outward unit normals to support planes to P at p. A vertex v of P is mapped by G to a spherical polygon $G(v)$ [6]. Likewise, the inverse Gaussian Map, denoted by G^{-1}, maps the spherical features to the polytope boundary.

We convert each sub-part represented as a polyhedral surface into a Gaussian map, represented as an arrangement of geodesic arcs embedded on the sphere, where each face f of the arrangement is extended with the coordinates of its associated primal vertex $v = G^{-1}(f)$, resulting with a unique representation. The construction of an arrangement from the polytope features and their accessible incident relations provided by the `Polyhedron_3` data-structure amounts to the insertion of geodesic segments that are pairwise disjoint in their interior into the arrangements, an operation that can be carried out efficiently.

The output of this phase is an ordered list of parts, where each part is an ordered list of the Gaussian maps of the convex sub-parts. Figure 3 depicts the Gaussian maps of six of the 18 polytopes that comprise the set of sub-parts of our Split Star assembly.

3.3 Sub-part Gaussian Map Reflection

We reflect each sub-part P_k^i through the origin to obtain $-P_k^i$. This operation can be performed directly on of the output of the previous phase, reflecting the underlying arrangements of geodesic arcs embedded on the sphere, which represent the Gaussian maps, while handling the additional data attached to the arrangement faces. As

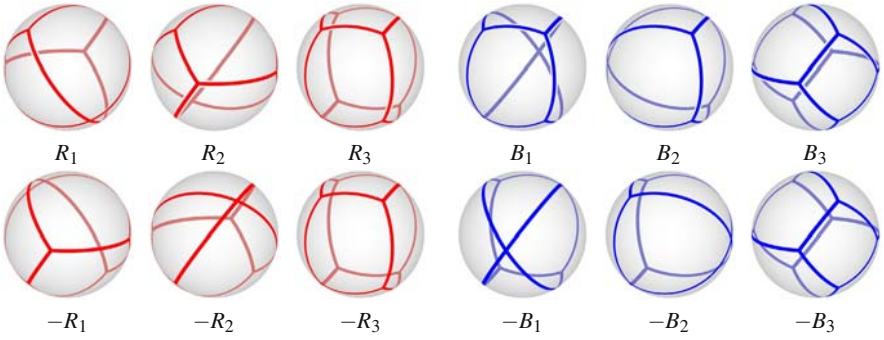


Fig. 3. Samples of the Gaussian maps of sub-parts of the Split Star assembly. The bottom row contains the reflections of the Gaussian maps at the top row.

a matter of fact, this phase can be reduced as part of an optimization discussed in Section 4.

The output of this phase is an ordered list of parts, where each part is an ordered list of Gaussian maps of the reflected convex sub-parts. Figure 3 depicts the Gaussian maps of six of the 18 polytopes that comprise the set of reflected sub-parts of the Split Star example.

3.4 Pairwise Sub-part Minkowski Sum Construction

Construct Pairwise Sub-part Minkowski Sums
for i **in** $\{1,2,\ldots,n\}$
for j **in** $\{1,2,\ldots,n\}$
if $i == j$ **continue**
for k **in** $\{1,2,\ldots,m_i\}$
for ℓ **in** $\{1,2,\ldots,m_j\}$
$M_{k\ell}^{ij} = P_\ell^j \oplus (-P_k^i)$

We compute the Minkowski sums of the pairwise sub-parts and reflected sub-parts. Aiming for an efficient output-sensitive algorithm, the construction of an individual Minkowski sum from two Gaussian maps represented as two arrangements respectively is performed by overlaying the two arrangements. When the overlay operation progresses, new vertices, edges, and faces of the resulting arrangement are created based on features of the two operands. When a new feature is created its attributes are updated. There are ten cases that must be handled [5]. For example, a new face f is induced by the overlap of two faces f_1 and f_2 of the two summands respectively. The primal vertex associated with f is set to be the sum of the primal vertices associated with f_1 and f_2 respectively.

The `Arrangement_on_surface_2` package conveniently supports the overlay operation allowing users to provide their own version of these ten operations. The overlay operation is exploited below on several different variants of arrangements of geodesic arcs embedded on the sphere. Each application requires the provision of a different set of those ten operations.

The output of this phase is a map from ordered pairs of distinct indices into lists of Minkowski sums represented as Gaussian maps. Each ordered pair $< i, j >, i \neq j$ is

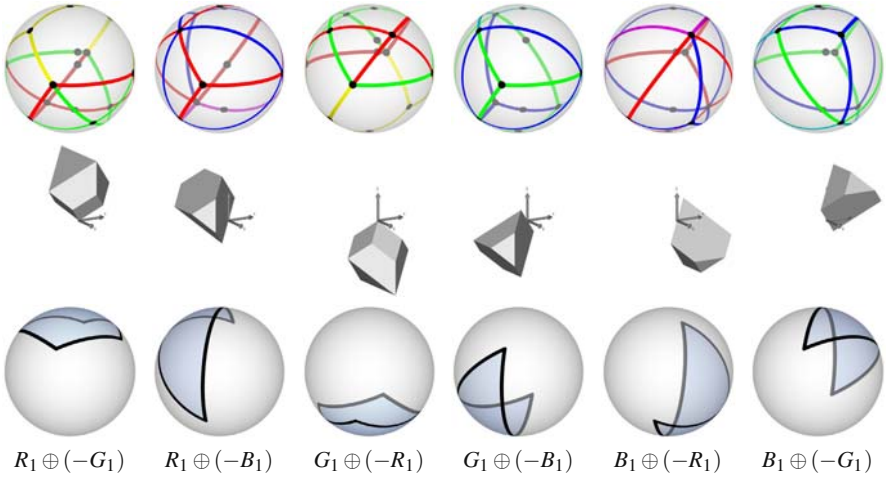


Fig. 4. Samples of the pairwise Minkowski sums of sub-parts of the Split Star assembly. The middle row contains six Minkowski sums. The top row contains the corresponding Gaussian maps. The bottom row contains the corresponding central projections of the Minkowski sums on $\mathbb{S}^2$.

associated with the list of Minkowski sums $\{M_{k\ell}^{ij} \mid k = 1,2,\ldots,m_i,\ \ell = 1,2,\ldots,m_j\}$. In case of our Split Star the map consists of 30 entries that correspond to all configurations of ordered distinct pairs of parts. Each entry consists of a list of nine Minkowski sums, that is, 270 Minkowski sums in total.

3.5 Pairwise Sub-part Minkowski Sum Projection

Project Pairwise Sub-part Minkowski sums

for i **in** $\{1,2,\ldots,n\}$
  for j **in** $\{1,2,\ldots,n\}$
    if $i == j$ **continue**
    for k **in** $\{1,2,\ldots,m_i\}$
      for ℓ **in** $\{1,2,\ldots,m_j\}$
        $Q_{k\ell}^{ij} = \text{project}(M_{k\ell}^{ij})$

We centrally project all pairwise sub-part Minkowski sums computed in the previous phase onto the sphere. Each projection is represented as an arrangement of geodesic arcs on the sphere, where each cell c of the arrangement is extended with a Boolean flag that indicates whether all infinite rays emanating from the origin in all directions $\mathbf{d} \in c$ pierce the corresponding Minkowski sum. As the Minkowski sums are convex, their spherical projections are spherically convex.

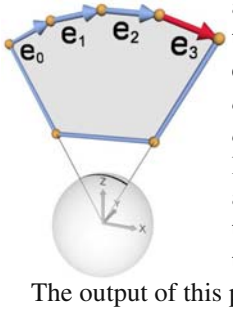
Given a convex Minkowski sum C, we distinguish between four different cases as follows:

1. The origin is contained in the interior of a facet of C.
2. The origin lies in the interior of an edge of C.
3. The origin coincides with a vertex of C.
4. The origin is separated from C.

Computing the projection of a convex polytope C can be done efficiently using dedicated procedures that handle the four cases respectively. Recall that C is represented as a Gaussian map, which is internally represented as an arrangement of geodesic arcs embedded on the sphere. We traverse the vertices of the arrangement. For each vertex v we extract its associated primal facet $f = G^{-1}(v)$. We dispatch the appropriate computation based on the relative position of the origin with respect to the supporting plane to f, and the supporting plane to adjacent facets of f.

If the origin is contained in the interior of a facet f of C, the projection of the silhouette of C is a great (full) circle that divides the sphere into two hemispheres. The normal to the plane that contains the great circle is identical to the normal to the supporting plane to f, easily extracted from the arrangement representing the Gaussian map of C. The `Arrangement_on_surface_2` package conveniently supports the insertion of a great circle, provided by the normal to the plane that contains it, into an arrangement of geodesic arcs embedded on the sphere.

We omit the implementation details of the succeeding two cases, and proceed to the general case. **If the origin is separated from** C, we traverse all edges of C until we find a silhouette edge characterized as follows: Let v_s and v_t be the source and target vertices of some edge e in the arrangement representing the Gaussian map of C, and let $f_s = G^{-1}(v_s)$ and $f_t = G^{-1}(v_t)$ be their associated primal facets respectively. e is a silhouette edge, if and only if, the origin is not in the negative side of the supporting plane to f_s and not in the positive side of the supporting plane to f_t. We start with the first silhouette edge we find, and search for an adjacent silhouette edge in a loop, until we rediscover the first one. We project only the target vertices of significant silhouette edges, and connect consecutive projections using arcs of great circle. Let e and e' be adjacent silhouette edges. We skip e, if the projections of e and e' lie on the same great circle. For example, all but the last adjacent silhouette edges incident to a facet supported by a plane that contains the origin are redundant, as illustrated in the figure on the left. Here we skip e_0, e_1, and e_2, and project the target vertex of e_3.



The output of this phase is a map from ordered pairs of distinct indices into lists of arrangements as described above. Each ordered pair $< i, j >, i \neq j$ is associated with the list of central projections of the pairwise Minkowski sums of P_j's sub-parts and the reflection through the origin of P_i's sub-parts.

3.6 Pairwise Minkowski Sum Projection

For each pair of distinct parts P_i and P_j we compute the union of projections of the pairwise Minkowski sums of all sub-parts of part P_j and reflections of all sub-parts of part P_i.

The output of this phase is a map from ordered pairs of distinct indices into arrangements. Each ordered pair $< i, j >, i \neq j$ is associated with a single

```
Unite Pairwise Sub-part
  Minkowski sums Projections
for i in {1,2,...,n}
  for j in {1,2,...,n}
    if i == j continue
    Q_ij = ∅
    for k in {1,2,...,m_i}
      for ℓ in {1,2,...,m_j}
        Q_ij = Q_ij ∪ Q_kℓ^ij
```

arrangement extended as described above, that represents the central projection Q_{ij} of $M_{ij} = P_j \oplus (-P_i)$.

We exploit the overlay operation in this phase the second time throughout this process, this time in a loop. Given two distinct parts P_i and P_j we traverse all projections in the set $\{Q_{k\ell}^{ij} | k = 1,2,\ldots,m_i, \ell = 1,2,\ldots,m_j\}$, and accumulate the result in the arrangement Q_{ij}. As mentioned in Section 3.4, when the overlay operation progresses, new vertices, edges, and faces of the resulting arrangement are created. When a new face f is created as a result of the overlay of a face g in some projection $Q_{k\ell}^{ij}$, and a face in the accumulating arrangement, the Boolean flag associated with f, which indicates whether all directions $\mathbf{d} \in f$ pierce M_{ij}, is turned on, if $\mathbf{d}$ pierces $M_{k\ell}^{ij}$, that is, if the flag associated with the face of g is on.

The intermediate result of this step are arrangements with potentially redundant edges and vertices. It is desired, (but not necessary,) to remove these cells, as this will reduce the time consumption of the succeeding operations, which is directly related to the complexity of the arrangements. It has even a larger impact when the optimization described in Section 4 is applied, as the optimization decreases the number of preceding operation at the account of slightly increasing the number of succeeding operations. We remove all edges and vertices that are in the interior of the projection, that is all marked edges and vertices. We also remove spherically collinear vertices on the boundary of the projection, the degree of which decreased below three, as a result of the redundant-edge removal.

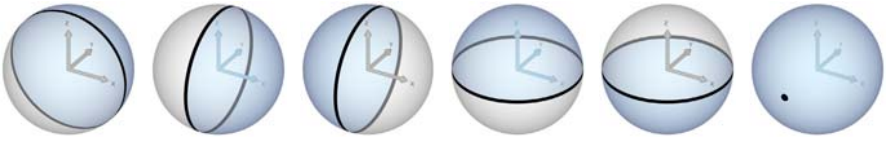
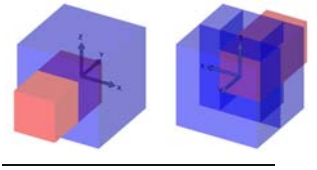


Fig. 5. Peg-in-the-hole Minkowski sum projections. (a), (b), (c), (d), and (e) are the sub-part projection. (f) is the union of all the former.

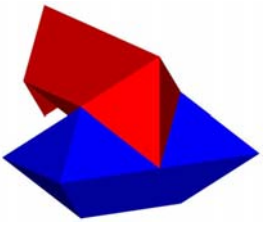
CGAL also supports the union operation among other Boolean operations applied to general polygons.[3] However, it consumes and produces *regularized* general polygons. This regularization operation is harmful in the realm of assembly planning. Therefore, we work directly on the cells of the arrangements Q_{ij}. Quite often the projection contains isolated vertices and edges, as occurs in the example depicted on the left, referred to as "peg-in-the-hole". Here the assembled product is translucently



[3] The code supports point sets bounded by algebraic curves embedded on parametric surfaces referred to as general polygons.

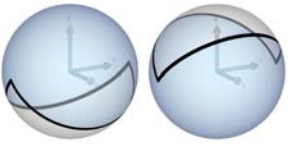
viewed from two opposite directions. The blue part is stationary and is decomposed into five sub-parts. Figure 5 illustrates the corresponding five pairwise Minkowski sum projections, and their union. The complement of the union consists of a single isolated vertex.

Recalling our Split Star assembly, the projection of the Minkowski sum of the red part and the reflection of the blue part, and its reflection, that is, the projection of the Minkowski sum of the blue part and the reflection of the red part are depicted on the right.



3.7 Motion-Space Construction

We compute a single arrangement that represents the motion space, where each cell c of the arrangement is extended with a DBG. We use the adjacency-matrix storage format provided by BOOST[4] to represent each DBG. Recall that for a graph with n vertices such as ours, an $n \times n$ matrix is used, where each element a_{ij}^c of a DBG associated with cell c is a Boolean flag that indicates whether part P_i collides with part P_j when moved along any direction $\mathbf{d} \in c$. Handling large assemblies with sparse blocking relations may require different representations of DBGs to reduce memory consumption.



$B \oplus (-R)$ $R \oplus (-B)$

We exploit the overlay operation in this phase the third time similar to its application in Section 3.6. We iterate over all central projections in the set Q_{ij}, and accumulate the result in the final motion-space arrangement. As mentioned above in Section 3.4, when the overlay operation progresses, new vertices, edges, and faces of the resulting arrangement are created. When a new cell c is created as a result of the overlay of a face g in some projection Q_{ij}, and a cell in the accumulating arrangement, the DBG associated with c is updated. That is, if the flag associated with g is turned on, we insert an edge between vertex i and vertex j into the DBG associated with c.

Depicted on the right is the motion-space arrangement computed by our program for the Split Star assembly.

3.8 Motion-Space Processing

We traverse all vertices, edges, and faces of the motion-space arrangement in this order, and test the DBG associated with each cell for strong connectivity using the BOOST global function `strong_components()`. This function computes the strongly connected components of a directed graph using Tarjan's algorithm based on DFS [20]. The set of constraints associated with a vertex v is a proper subset of the constraints associated with edges incident to v. Similarly, the set of constraints associated with an edge e is a proper subset of the constraints associated

[4] http://www.boost.org

with the two faces incident to e. Therefore, if the DBGs of all vertices are strongly connected, we terminate with the conclusion that the assembly is interlocked. Similarly, if we are interested in finding all solutions, and the DBGs of all edges are strongly connected, we terminate, as no further solutions on faces exist.

For the Split Star assembly, our program successfully identifies all the eight partitioning directions depicted above along with the corresponding subset of parts listed on the right.

	Direction	Subset
1.	$-1,-1,-1$	*GBT*
2.	$-1,-1,\ 1$	*RBT*
3.	$-1,\ 1,-1$	*GPT*
4.	$-1,\ 1,\ 1$	*RPT*
5.	$\ 1,-1,-1$	*GBY*
6.	$\ 1,-1,\ 1$	*RBY*
7.	$\ 1,\ 1,-1$	*GPY*
8.	$\ 1,\ 1,\ 1$	*RPY*

4 Additional Optimization

The reflection of the sub-parts through the origin as described in Section 3.3 has been naively implemented. The computation is applied to the polyhedral-mesh representation of each sub-part. An immediate optimization calls for an application of the reflection operation directly on the arrangements that represent the Gaussian map. We are planning to implementat the reflection operation, which operates on any applicable arrangement. This operation alters incidence relations between the arrangement features and their geometric embeddings. For each vertex, it reflects its associated point about the origin, and inverts the order of the halfedges incident to it. For each edge, it reflects its associated curve about the origin. For each face, it inverts the order of the halfedges along its outer boundary. Similar to the overlay operation (see Section 3.4), where the user can provide a set of ten functions, which are invoked when new vertices, edges, and faces of the resulting arrangement are created, while the overlay operation progresses, the user can provide a set of three functions that are invoked when a new vertex, halfedge, and face are created, while the reflection operation progresses. Extended data associated with these types, such as a primal vertex associated with an arrangement face as in the case of an arrangement representing a Gaussian map, can easily be updated with the provision of an appropriate function.

The trivial observation that $P \oplus (-Q) = -((-P) \oplus Q)$ leads to another optimization. Instead of reflecting all sub-parts in the set $\{P_k^i \,|\, i = 1,2,\ldots,n, k = 1,2,\ldots,m_i\}$, we reflect only the sub-parts in the set $\{P_k^i \,|\, i = 2,\ldots,n, k = 1,2,\ldots,m_i\}$, and compute only the pairwise sub-parts Minkowski sums in the set $\{M_{k\ell}^{ij} \,|\, 1 \le i < j \le n, k = 1,2,\ldots,m_i, \ell = 1,2,\ldots,m_j\}$, their central projection, and the union of the appropriate projections to yield the set $\{Q_{ij} \,|\, 1 \le i < j \le n\}$. Then, we apply the reflection operation described above on each member of this set, and obtain the full set of projections $\{Q_{ij} \,|\, i = 1,2,\ldots,n, j = 1,2,\ldots,n\}$. The Boolean flag associated with a face of an arrangement that represents a central projection is equal to the flag associated with its reflection. In other words, a face of Q_{ij} consists of directions that pierce M_{ij}, if and only if, its reflection in Q_{ji} consists of directions that pierce M_{ji}.

Phase 8 is purely topological. Thus, we do not expect the time consumption of this phase to dominate the time consumption of the entire process for any input.

Table 1. Time consumption (in seconds) of the execution of the eight phases (see Section 3) using the Split Star assembly as input. **A** — number of convex sub-parts per part. **B** — number of sub-part vertices per part. **C** — total number of convex sub-parts. **D** — total number of Minkowski sums. **E** — total number of arrangements of geodesic arcs embedded on the sphere constructed throughout the process.

A	B	C	D	E	1	2	3	4	5	6	7	8
3	16	18	270	607	NA	0.01	0.04	2.38	0.41	2.05		
5	22	30	750	1591	NA	0.01	0.05	5.03	1.09	7.07	0.36	0.01
8	32	48	1920	3967	NA	0.01	0.06	11.12	2.41	27.99		

Nevertheless, it might be possible to reduce its contribution to the total time consumption through efficient testing for strong connectivity applied to all the DBGs [16], exploiting the similarity between DBGs associated with incident cells. Recall, that the set of arcs in a DBG associated with a vertex v is a subset of the set of arcs associated with an edge incident to v. Similarly, the set of arcs in a DBG associated with an edge e is a subset of the set of arcs associated with a face incident to e. The proposed technique reduces the cost from $O(n^2)$ per DBG to an amortized cost of $O(n^{1.376})$, where n is the maximum number of arcs in any blocking graph.

5 Experimental Results

Our program can handle all inputs. However, due to lack of space, we limit ourselves to a small set of test cases, where we compare the impact of different decompositions on the process time-consumption. The results listed in Table 1 were produced by experiments conducted on a Pentium PC clocked at 1.7 GHz. In all three test cases we use the Split Star assembly as input. Naturally, in all three cases identical projections are obtained as the intermediate results of Phase 6, hence the identical time consumption of the succeeding last two phases. Evidently, it is desired to decompose each part into as few as possible sub-parts with as small as possible number of features. However, an automatic decomposition operation may require large amount of resources to arrive at optimal or near optimal decompositions. Notice that Phases 4 and 6 dominate the time complexity. This is due to the large number of geometric predicates that must be evaluated during the execution of the overlay operation.

References

1. Agarwal, P.K., Flato, E., Halperin, D.: Polygon decomposition for efficient construction of Minkowski sums. Comput. Geom. Theory Appl. 21, 39–61 (2002)
2. Berberich, E., Fogel, E., Halperin, D., Melhorn, K., Wein, R.: Sweeping and maintaining two-dimensional arrangements on surfaces: A first step. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 645–656. Springer, Heidelberg (2007)
3. Bool, F.H., et al.: M. C. Escher: His Life and Complete Graphic Work. Harry N. Abrams, Inc. (1982)

4. Coffin, S.T.: Geometric Puzzle Design, 2nd edn. A.K. Peters, Ltd. (2006)
5. Fogel, E., Halperin, D.: Exact and efficient construction of Minkowski sums of convex polyhedra with applications. CAD 39(11), 929–940 (2007)
6. Fogel, E., Halperin, D., Weibel, C.: On the exact maximum complexity of Minkowski sums of convex polyhedra. In: Proc. 23rd Annu. ACM Symp. Comput. Geom., pp. 319–326 (2007)
7. Fogel, E., Setter, O., Halperin, D.: Exact implementation of arrangements of geodesic arcs on the sphere with applications. In: Abstracts of 24th Eur. Workshop Comput. Geom., pp. 83–86 (2008)
8. Fogel, E., Setter, O., Halperin, D.: Movie: Arrangements of geodesic arcs on the sphere. In: Proc. 24th Annu. ACM Symp. Comput. Geom., pp. 218–219 (2008)
9. Guibas, L.J., Halperin, D., Hirukawa, H., Latombe, J.-C., Wilson, R.H.: Polyhedral assembly partitioning using maximally covered cells in arrangements of convex polytopes. Int. J. Comput. Geom. Appl. 8, 179–200 (1998)
10. Hachenberger, P.: Exact minkowksi sums of polyhedra and exact and efficient decomposition of polyhedra in convex pieces. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 669–680. Springer, Heidelberg (2007)
11. Halperin, D., Latombe, J.-C., Wilson, R.H.: A general framework for assembly planning: The motion space approach. Algorithmica 26, 577–601 (2000)
12. Kavraki, L., Kolountzakis, M.: Partitioning a planar assembly into two connected parts is NP-complete. Inf. Process. Lett. 55, 159–165 (1995)
13. Kettner, L.: Using generic programming for designing a data structure for polyhedral surfaces. Comput. Geom. Theory Appl. 13(1), 65–90 (1999)
14. Kettner, L.: 3D polyhedral surfaces. In: Cgal Editorial Board (ed.) Cgal User and Reference Manual, 3.3rd edn. (2007)
15. Kettner, L., Mehlhorn, K., Pion, S., Schirra, S., Yap, C.K.: Classroom examples of robustness problems in geometric computations. In: Albers, S., Radzik, T. (eds.) ESA 2004. LNCS, vol. 3221, pp. 702–713. Springer, Heidelberg (2004)
16. Khanna, S., Motwani, R., Wilson, R.H.: On certificates and lookahead in dynamic graph problems. Algorithmica 21(4), 377–394 (1998)
17. Luke, D.: Stellations of the rhombic dodecahedron. The Math. Gazette 41(337), 189–194 (1957)
18. Natarajan, B.K.: On planning assemblies. In: Proc. 4th ACM Symp. Comput. Geom., pp. 299–308 (1988)
19. Snoeyink, J., Stolfi, J.: Objects that cannot be taken apart with two hands. Disc. Comput. Geom. 12, 367–384 (1994)
20. Tarjan, R.E.: Depth first search and linear graph algorithms. SIAM J. on Computing 1(2), 146–160 (1972)
21. Wein, R., Fogel, E., Zukerman, B., Halperin, D.: 2D arrangements. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.3 edn. (2007)
22. Wein, R., Fogel, E., Zukerman, B., Halperin, D.: Advanced programming techniques applied to CGAL's arrangement package. Comput. Geom. Theory Appl. 38(1-2), 37–63 (2007); Special issue on CGAL
23. Wilson, R.H., Latombe, J.-C.: Geometric reasoning about mechanical assembly. Artificial Intelligence 71(2), 371–396 (1994)

Realistic Reconfiguration of Crystalline (and Telecube) Robots*

Greg Aloupis[1], Sébastien Collette[1,**], Mirela Damian[2], Erik D. Demaine[3,***], Dania El-Khechen[4], Robin Flatland[5], Stefan Langerman[1,†], Joseph O'Rourke[6], Val Pinciu[7], Suneeta Ramaswami[8], Vera Sacristán[9,‡], and Stefanie Wuhrer[10]

Abstract. In this paper we propose novel algorithms for reconfiguring modular robots that are composed of n atoms. Each atom has the shape of a unit cube and can expand/contract each face by half a unit, as well as attach to or detach from faces of neighboring atoms. For universal reconfiguration, atoms must be arranged in $2 \times 2 \times 2$ modules. We respect certain physical constraints: each atom reaches

[1]Université Libre de Bruxelles, Belgique
e-mail: {galoupis,secollet,slanger}@ulb.ac.be

[2]Villanova University, Villanova, USA
e-mail: mirela.damian@villanova.edu

[3]Massachusetts Institute of Technology, Cambridge, USA
e-mail: edemaine@mit.edu

[4]Concordia University, Montreal, Canada
e-mail: d_elkhec@cs.concordia.ca

[5]Siena College, Loudonville, N.Y., USA
e-mail: flatland@siena.edu

[6]Smith College, Northampton, USA
e-mail: orourke@cs.smith.edu

[7]Southern Connecticut State University, USA
e-mail: pinciu@scsu.ctstateu.edu

[8]Rutgers University, Camden, NJ, USA
e-mail: rsuneeta@camden.rutgers.edu

[9]Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: vera.sacristan@upc.edu

[10]Carleton University, Ottawa, Canada
e-mail: swuhrer@scs.carleton.ca

* This work was initiated at the Bellairs Research Institute of McGill University.

** Chargé de Recherches du FNRS.

*** Partially supported by NSF CAREER award CCF-0347776, DOE grant DE-FG02-04ER25647, and AFOSR grant FA9550-07-1-0538.

† Maître de Recherches du FNRS.

‡ Partially supported by projects MEC MTM2006-01267 and DURSI 2005SGR00692.

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 433–447.
springerlink.com

at most unit velocity and (via expansion) can displace at most one other atom. We require that one of the atoms can store a map of the target configuration. Our algorithms involve a total of $O(n^2)$ such atom operations, which are performed in $O(n)$ parallel steps. This improves on previous reconfiguration algorithms, which either use $O(n^2)$ parallel steps [8, 10, 4] or do not respect the constraints mentioned above [1]. In fact, in the setting considered, our algorithms are optimal, in the sense that certain reconfigurations require $\Omega(n)$ parallel steps. A further advantage of our algorithms is that reconfiguration can take place within the union of the source and target configurations.

1 Introduction

Crystalline and Telecube robots. In this paper, we present new algorithms for the reconfiguration of robots composed of *crystalline atoms* [3, 4, 8] or *telecube atoms* [9, 10], both of which have been prototyped.

The atoms of these robots are cubic in shape, and are arranged in a grid configuration. Each atom is equipped with mechanisms allowing it to extend each face out one unit and later retract it back. Furthermore, the faces can attach to or detach from faces of adjacent atoms; at all times, the atoms should form a connected unit. The default configuration for a Crystalline atom has expanded faces, while the default for a Telecube atom has contracted faces.

When groups of atoms perform the four basic atom operations (expand, contract, attach, detach) in a coordinated way, the atoms move relative to one another, resulting in a reconfiguration of the robot. Figure 1 shows an example of a reconfiguration. To ensure that all reconfigurations are possible, atoms must be arranged in $k \times k \times k$ modules, where $k \geq 2$ [1, 10]. In the $2D$ setting that we focus on, we assume that modules consist of 2×2 atoms. Our algorithms can easily be extended to $3D$.

We refer the reader to [8, 10, 1] for a more detailed and basic introduction to these robots. Various types of self-reconfiguring robots, as well as related algorithmic issues, are surveyed in [6, 11].

The model. The problem we solve is to reconfigure a given connected source configuration of n modules to a specified, arbitrary, connected target configuration T in $O(n)$ parallel steps. We allow modules to be able to exert only a constant amount of force, independent of n. In other words each module has the ability to push/pull one other module by a unit distance (the length of one module) within a unit of time. Simply bounding the force may still lead to arbitrarily high velocities and thus rather unrealistic motions. On the other hand, in some situations where maximal control is desired (e.g., treacherous conditions, dynamic obstacle environment, minimally stable static configuration of the robot itself) it may be desirable to strictly limit

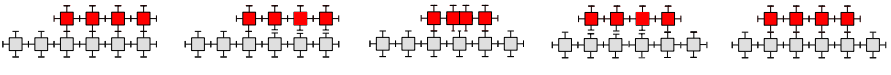


Fig. 1. Example of reconfiguring crystalline atoms.

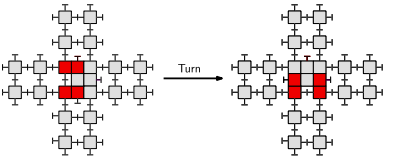


Fig. 2. The middle cell contains two modules. The red *guest* module is capable of turning orientation. Only initial and final configurations are shown.

velocity. Thus we also bound maximum velocity (and so the momentum) by a constant (module length/unit time). Our algorithms are designed for Crystalline robots. For a discussion of the main differences for Telecube robots, see Section 5.

We restrict our descriptions to a 2D lattice. None of our techniques depend on dimension, so it is straightforward to extend to 3D robots. Given the 2×2 module size, a cell of the lattice can contain up to two modules (see Fig. 2). Cells are marked with an integer in $\{0,1,2\}$: a 0-cell corresponds to a node in T that has no module yet, a 1-cell contains one module, and a 2-cell contains two (compressed) modules. In a 2-cell, we sometimes distinguish between the *host* module and the *guest* module.

Let r_0 be a specialized module that has access to a map of the target configuration, T. We compute a spanning tree S of the source configuration, rooted at r_0, and instruct cells to attach/detach so that the attachments model the tree connections in S. The spanning tree can be computed in linear time and constructed via local communication. The tree structure between cells is maintained throughout the algorithm by physical connections between host modules. These modules are also responsible for the parent-child pointer structure of the tree. For each node $u \in S$, let $P(u)$ denote the parent of u in S. A child of a cell u is adjacent either on the east, north, west, or south side of u. Let the *highest priority child* of u be the first child in counterclockwise order starting with the east direction.

Related results. Algorithms for reconfiguring Crystalline and Telecube robots in $O(n^2)$ parallel steps have been given in [8, 10, 4]. The same bound is implied in [5], which deals with reconfigurations of a specific class of modular robots (more restrictive than Crystalline). A linear-time parallel algorithm for reconfiguring within the bounding box of source and target is given in [1]. The total number of individual moves is also linear. However, no restrictions are made concerning physical properties of the robots. For example, $O(n)$ strength is required, since modules can carry towers and push large masses during certain operations. An $O(\log n)$ time algorithm for 2D robots that uses $O(n \log n)$ parallel moves and also stays within the bounding box is given in [2] (and it seems that the algorithm can be extended to 3D). However, not only do modules have $O(n)$ physical strength, they can also reach $O(n)$ velocity. An $O(\sqrt{n})$ time algorithm for 2D robots, using the third dimension as an intermediate, is given in [7]. This is optimal in the model considered, which permits linear velocities, but only constant acceleration. If applied within the model used in [2], this algorithm would run in constant time. We remind the reader that, unlike [7, 1, 2], we limit force and velocity to a constant level.

Contributions of this paper. We present two algorithms to reconfigure Crystalline robots in $O(n)$ time steps, using $O(n)$ parallel moves per time step. Our first algorithm (Section 3) is slightly simpler to describe, and relatively easily adaptable to Telecube robots. It also forms the basis of our second algorithm (Section 4), which is exactly *in-place*, i.e., it uses only the cells of the union of the source and target configurations. This is particularly interesting if there are obstacles in the environment. Both algorithms consider the given robot as a spanning tree, and push leaves towards the root with "parallel tunneling". No global communication is required. This means that constant-size memory suffices for each non-root module, which can decide how to move at each step based solely on the states of its neighbors. In the realistic model considered in this paper, our algorithms are optimal, in the sense that certain reconfigurations require a linear number of parallel moves.

2 Primitive Operations

Let m and q be adjacent cells. We define the following primitive operations:

1. PUSHINLEAF(m,q) – applies when $q = P(m)$, m is a leaf, and both are uncompressed. Here, m becomes empty and q becomes compressed (i.e., q takes the module of m as a guest).
2. POPOUTLEAF(m,q) – applies when q is compressed and m is empty. This is the inverse of the PUSHINLEAF operation.
3. TRANSFER(m,q) – applies when m is compressed and q is non-empty; if q is compressed, the guests of both cells physically exchange positions. Otherwise, the guest of m moves into (and becomes a guest of) q.
4. ATTACH(m,q) – host modules in m and q form a physical connection.
5. DETACH(m,q) – the inverse of ATTACH.
6. SWITCH(m) – applies when m is compressed. Its two modules physically switch positions (and roles of host and guest).

PUSHINLEAF, POPOUTLEAF, and TRANSFER are illustrated in Fig. 3.

Note that any permutation of atoms within a module can be realized in linear time with respect to the size of the module (i.e., $O(1)$ time for our modules). Thus a compressed cell may transfer or push one module to any direction, and two modules within a cell can switch roles. Details are omitted due to space restrictions.

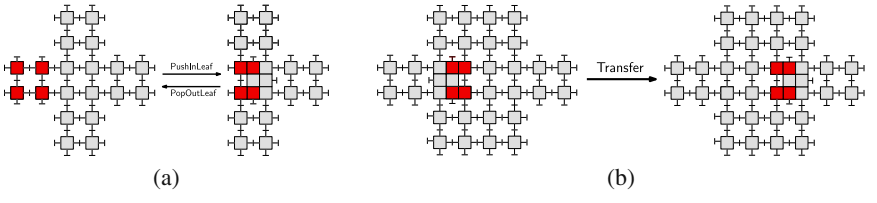


Fig. 3. (a) PUSHINLEAF and POPOUTLEAF. (b) TRANSFER. Only initial and final configurations are shown.

In the remainder of this paper, we assume that all parallel motions are synchronized. However, due to the simple hierarchical tree structure of our robots, we find it plausible that our algorithms could be implemented so that modules may operate asynchronously. Details remain to be verified.

Lemma 2.1. *Operations* PUSHINLEAF*,* POPOUTLEAF*,* SWITCH *and* TRANSFER *maintain the tree structure of a robot.*

The proof is omitted for lack of space.

In our basic motions, modules move by one unit length per time step. The moving modules do not carry other modules. Thus our reconfiguration algorithms place no additional force constraints beyond those required by *any* reconfiguration algorithm.

3 Reconfiguration via Canonical Form

This section describes an algorithm to reconfigure S into T via an intermediate canonical configuration. Modules follow a path directly to the root r_0, and into a canonical "storage configuration". We focus on the construction of one type of canonical form, a vertical line V. In fact V could be any path that avoids the source configuration. Thus the entire reconfiguration can take place relatively close to the bounding box of S. Reconfiguring from V to T is relatively straightforward and not discussed here, due to space restrictions.

We first move r_0 to a maximum possible y-coordinate within S: This involves pushing in a leaf and iteratively transferring it to r_0, so that r_0 becomes part of a 2-cell and then is able to iteratively transfer to its target. Note that this might not be necessary in implementations in which all modules are capable of playing the role of r_0 (for example, if all modules have a map of T, or if all are capable of communicating to an external processor). This initial step is followed by two main phases, during which r_0 does not move.

In the first phase, we repeatedly apply procedure CLUSTERSTEP to move modules closer to r_0, by compressing in at the leaves and moving up S in parallel. The shape of S shrinks, as PUSHINLEAF operations in CLUSTERSTEP compress leaf modules into their parent cells. It is not critical that all cells become compressed. In fact this phase mainly helps to analyze the total number of parallel steps in our algorithm. At the end of this phase, all *non-leaf* cells will become 2-cells. In this state we refer to S as being *fully compressed*.

CLUSTERSTEP(S)

For all cells u in S except for that containing r_0, execute the following in parallel:
  If $P(u)$ is a 1-cell
    If u is the highest priority child of $P(u)$ and
    all siblings of u are leaves or 2-cells,
      If u is a 1-cell leaf then PUSHINLEAF$(u, P(u))$.
      If u is a 2-cell, then TRANSFER$(u, P(u))$.

SOURCECLUSTER(S)

Repeat until S is fully compressed
CLUSTERSTEP(S).

SOURCECLUSTER is illustrated in Figure 4. The task of compressing a parent cell $P(u)$ falls onto its highest priority child, u. Note that $P(u)$ first becomes compressed only when all its subtrees are *essentially* compressed. That is, even if u is ready to supply a module to $P(u)$, it waits until all other children are also ready. This rule could be altered, and in fact the whole process would then run slightly faster. Here, we ensure that once the root of a subtree becomes compressed, it will supply a steady stream of guest modules to its ancestors.

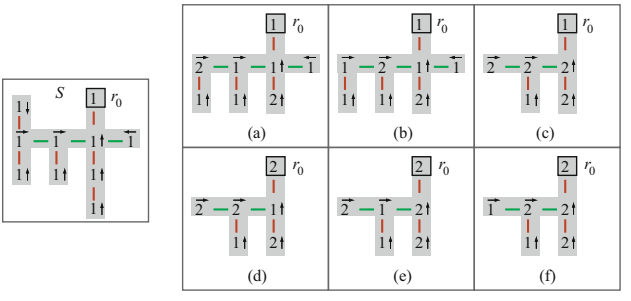


Fig. 4. An example of SOURCECLUSTER.

In the second phase, we construct V while emptying S, one module at a time. This is described in the second step of algorithm TREETOPATH, and is illustrated in Figure 5.

Algorithm TREETOPATH(S, V)

1. SOURCECLUSTER(S)
2. Let d be the cell containing r_0 as a *host*. Let $V = d$.
   Repeat until V contains all modules:
   a) For all 2-cells u in V, execute in parallel:
   Let c be the cell vertically above u.
   If c is empty, POPOUTLEAF(u, c);
   Otherwise, TRANSFER(u, c).
   b) CLUSTERSTEP(S)

Lemma 3.1. *If S is a set of modules physically connected in a tree of cells, then* CLUSTERSTEP*(S) returns a tree containing the same set of modules, while maintaining connectivity. So does* SOURCECLUSTER*(S).*

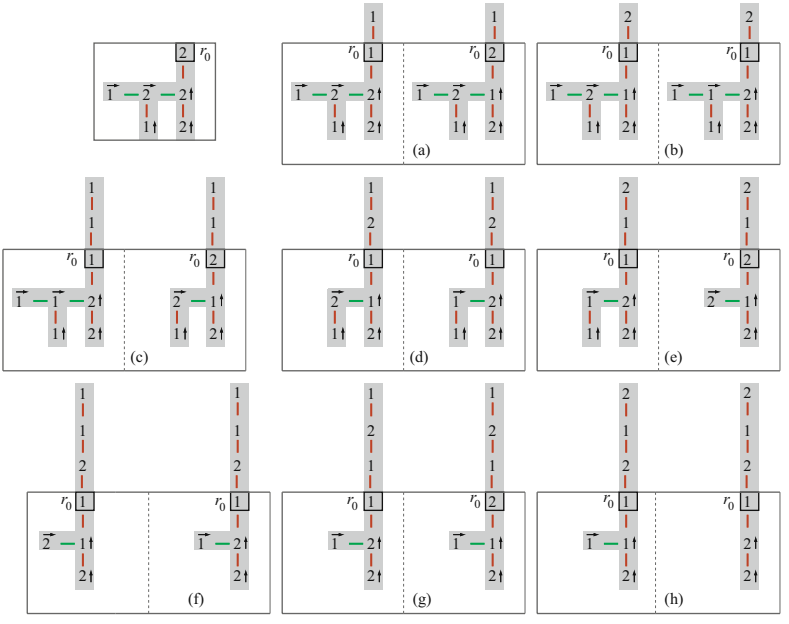


Fig. 5. An example of TREETOPATH.

Proof. CLUSTERSTEP invokes two basic operations, PUSHINLEAF and TRANSFER. By Lemma 2.1, these operations maintain a tree. The claim follows immediately for SOURCECLUSTER. □

The height of a cell in S is the height of its subtree in S.

Lemma 3.2. *Let r be a cell in S with height $h \geq 2$. In iteration $h-1$ of* SOURCECLUSTER*(S), r becomes a 2-cell for the first time.*

Proof. Prior to the first iteration, S contains only 1-cells. The proof is by induction on h. For the base case when $h = 2$, the children of r are leaves. Therefore, in the first iteration, during CLUSTERSTEP, the highest priority leaf compresses into r.

Now assume inductively that the lemma is true for all subtrees of height smaller than h. Cell r must have at least one child c with height $h-1$. By the inductive hypothesis, c becomes a 2-cell in iteration $h-2$, and all its other non-leaf children are 2-cells by the end of iteration $h-2$. Therefore, at iteration $h-1$, for the first time the conditions are satisfied for r to receive a module from its highest priority child during CLUSTERSTEP. □

Lemma 3.3. *Let r be a 2-cell with height h that transfers its guest module to $P(r)$ in iteration i of* SOURCECLUSTER*. Then at the end of iteration $i+1$, r is either a leaf or a 2-cell again.*

Proof. First note that at the beginning of iteration $i+1$, r is a 1-cell and $P(r)$ is a 2-cell. Thus if r is a leaf after iteration i, it remains so. On the other hand if r has

children ($h \geq 2$), it will become a 2-cell. We prove this by assuming inductively that our claim holds for all heights less than h. Consider the base case when $h = 2$. At the end of iteration i, all children of r are leaves and thus one will compress into r (note that r might *also* become a leaf in this particular case).

For $h > 2$, consider the iteration $j < i$ in which r received the guest module that it later transfers to $P(r)$ in iteration i. At the beginning of iteration j, all of r's children were leaves or 2-cells, since that is a requirement for r to receive a guest. Let c be the child that passed the module to r. If c used the PUSHINLEAF operation, then at the end of iteration j, r has one fewer children (but at least one). The other children remain leaves or 2-cells until iteration $i+1$, when r becomes a 1-cell again. Thus in iteration $i+1$, conditions are set for r to receive a module.

On the other hand, if c used the TRANSFER operation, we apply the inductive hypothesis: at the end of iteration $j+1 \leq i$, c is either a leaf or a 2-cell. During iterations j and $j+1$ in which r is busy receiving or transferring a module, all other children of r (if any) remain leaves or 2-cells. Therefore in iteration $j+2 \leq i+1$, the conditions are set for r to receive a module. □

Let the depth of a cell in a tree be its distance from the root. Hence, the root has depth zero.

Lemma 3.4. SOURCECLUSTER *terminates after at most* $2h-1$ *iterations of* CLUSTERSTEP.

Proof. We claim that at the completion of iteration $h-1+d$ of SOURCECLUSTER, all non-leaf modules at depth less than or equal to d in S_{h-1+d} are 2-cells. The proof is by induction on d. The base case is the root of S_{h-1} at depth $d = 0$. By Lemma 3.2, the root becomes a 2-cell in iteration $h-1$. Assume inductively that our claim is true for all values d', where $0 \leq d' < d$.

Now consider a cell p at depth $d-1$ that has children. By the inductive hypothesis, p and all its ancestors are 2-cells by the end of iteration $i = h-1+(d-1)$, and p is the last of this group to become a 2-cell. Thus at the beginning of iteration i, all children of p are either leaves or 2-cells. During iteration i, only p's highest priority child c changes, either by transferring a guest module into p (if c is a 2-cell), or by pushing into p (if c is a 1-cell leaf). In the first case, by Lemma 3.3, c will be a 2-cell or a leaf by the end of iteration $i+1$. In the second case, c is not part of S anymore.

Since p will not accept new guest modules after iteration i, all siblings of c remain leaves or 2-cells during iteration $i+1$. Thus at the end of this iteration, our claim holds for depth d. By setting $d = h$, our result follows. □

Let a *long gap* consist of two adjacent 1-cells that are not leaves. A tree is *root-clustered* if it has no long gaps. Observe that a fully compressed tree is a special case of a root-clustered tree.

Lemma 3.5. *Let S be a root-clustered tree. Then after one application of* CLUSTERSTEP*(S), S remains root-clustered.*

Proof. This follows from claims in the proof of Lemma 3.3. Specifically, consider any 2-cell u. If CLUSTERSTEP keeps u as a 2-cell, then u is not part of a long gap.

Otherwise, if u sends a module to $P(u)$, none of the children of u attempt to transfer a module to u. Now consider any 1-cell non-leaf child y of u. Since there was no long gap in S, all children of y were either 2-cells or leaves. Thus y will become a 2-cell during this iteration of ClusterStep. Again we conclude that u cannot be part of a long gap. □

Theorem 3.1. *Algorithm* TreeToPath *terminates in linear time.*

Proof. By Lemma 3.4, SourceCluster terminates in linear time. In fact by treating the final top position of V as an implicit root, our claim follows.

More specifically, however, we analyze the transition from S into V. When SourceCluster terminates, S is fully compressed (i.e., root-clustered), and we set r_0 to be the host in cell d.

In step 2a, d sends a module to the empty position c vertically above, if c is not a 2-cell. We may treat the position c as $P(d)$, and consider step 2a to be synchronous to step 2b. In other words, d is the only child of c, and thus d follows the same rules as ClusterStep. In fact, since S is fully compressed, after the first iteration of phase 2, the tree rooted at c will be root-clustered (only c and the highest-priority child of d will not be 2-cells). Therefore, by Lemma 3.5, in every iteration of phase 2, S remains root-clustered. Thus in every even iteration, d supplies a module to c, and in every odd iteration d is given a module from one of its children. Informally, when d sends a module up into V, the gap (in the sense of lack of guest module) that is created in S travels down the highest priority path of S until it disappears at a leaf. In general, a guest module on the priority path will never be more than two steps away from d, following the analysis of Lemma 3.3. Within V, a stream of guest modules, two units apart, will move upward. One module will pop up into an empty cell, every three iterations. Thus compressed modules in V can always progress. □

Again, we remind the reader that our first phase need not terminate before the second commences. By compressing leaves and sending them towards the root, while simultaneously constructing V from the root whenever it becomes compressed, the target configuration will be constructed even sooner. Splitting into two distinct phases simply helps for the analysis.

4 In-Place Reconfiguration

This section describes an algorithm that reconfigures S into T by restricting the movement of all modules to the space occupied by $S \cup T$, as long as they intersect. If S and T do not intersect, then we also use the cells on the shortest path between them. Our description assumes intersection. We call such an algorithm *in-place*. If all modules were to know which direction to take in each time unit (for example, by having an external source synchronously transmit instructions to each module individually), then it would not be difficult to design an in-place algorithm similar to the one in Section 3. However, in this section we impose the restriction

that all modules are only capable of communicating locally. It is up to r_0 to direct all action.

Our algorithm consists of two phases. The first phase is identical to phase 1 of the TREETOPATH algorithm from Section 3 (i.e., clustering around the root).

In the second phase, r_0 carries out a DFS (depth-first search) walk on T, dynamically constructing portions that are not already in place. Apart from modules in cells adjacent to r_0 that receive its instructions, all other modules simply try to keep up with r_0 (i.e., they follow CLUSTERSTEP). Note that if r_0 is not initially inside T, it first must travel to such a position. At any time, this "moving root" will either be traveling through modules that belong to the partially constructed tree T, or will be expanding T beyond the current tree structure, using compressed modules that are tagging along close to r_0.

The INPLACERECONFIGURATION algorithm maintains a dynamically changing tree S, each of whose cells u maintains two links: a *physical* link corresponding to the physical connection between u and $P(u)$, and a *logical* link that could either be NULL, or identical to the physical link. We call the tree S_ℓ induced by the logical links the *logical tree*. S always contains all occupied cells. S_ℓ is the smallest tree containing the modules that are not in their final position in T. Thus at the end of the algorithm, $S = T$ and $S_\ell = \emptyset$.

We now describe the heart of the algorithm, which is the operation of phase 2.

TARGETGROW(S,T)

{**1. DFS Root Update** }
$d \leftarrow$ next cell in the DFS visit of T.
$c \leftarrow$ current 2-cell in which r_0 is a guest module.

Mechanical/Physical Operations
1.1 If d is a 0-cell,
POPOUTLEAF(c,d)
1.2 If d is not a 0-cell,
If $c \neq P(d)$,
ATTACH(c,d) and DETACH$(d,P(d))$.
TRANSFER(c,d).

Tree Structure Update
1.3 Set $P(c)$ to be d. Set $P(d)$ to NULL
1.4 Mark c as "visited".
1.5 Include d in S_ℓ.

{**2. Root Clustering** }
Until c and d become 2-cells, repeat:
(a) *Detach Leaf*: For all 1-cell leaves $u \in S_\ell$, execute in parallel:
If u is marked "visited", remove u from S_ℓ.
(b) CLUSTERSTEP(S_ℓ)
If r_0 is not the guest in c, SWITCH(c).

The full algorithm is summarized in the following:

Algorithm INPLACERECONFIGURATION(S,T)	
Phase 1.	$S \leftarrow$ SOURCECLUSTER(S).
Phase 2.	Repeat until r_0 reaches the final position in its DFS traversal: $S \leftarrow$ TARGETGROW(S,T).

In Phase 2, we start with $S_\ell = S$. Throughout this phase, the logical tree is maintained as a subtree of the physical tree: $S_\ell \subseteq S$. The host module in each cell uses a bit to determine if the cell is also part of S_ℓ. Pointers between cells and their parents apply for both trees.

The main idea of the target growing phase is to move r_0 through the cells of T in a DFS order. A caravan of modules will follow r_0, providing a steady stream of modules to fill in empty target cells that r_0 encounters. The algorithm repeats the following main steps:

1. DFS Root Update: r_0 is the guest of 2-cell c and is ready to depart. It marks c as "visited" (i.e., c now belongs to T). Then r_0 moves to the next cell d encountered in a DFS walk of T. This is accomplished either by uncompressing (popping) r_0 into d (see Fig. 6(a $\rightarrow$ b)), or by transferring r_0 to d (see Fig. 6(e $\rightarrow$ f)). Cell d is added to S_ℓ, if not already included.
2. Root Clustering: Modules in S_ℓ attempt to move closer to r_0, to ensure that they are readily available when r_0 needs them. However, host modules in their final target position should never be displaced from that position, so we must carefully prevent such modules from compressing towards r_0. To achieve this, we alternate between the following two steps, until c and d both become 2-cells:
   a. Logical Leaf Detach: remove any 1-cell leaf of S_ℓ that has been visited (i.e., is in T). Note that a detached 1-cell may end up back in S_ℓ one more time, during *Root Update*.
   b. Cluster Step: this step is applied to S_ℓ. Thus, only modules that are guests or unvisited leaves will try to move towards r_0.

Fig. 6 illustrates the INPLACERECONFIGURATION algorithm with the help of a simple example. The top row of the figure shows the source configuration S after phase 1 completes (left), and the target robot configuration (right). Links in S_ℓ, which is shaded, are depicted as arrows.

Lemma 4.1. *Algorithm* INPLACERECONFIGURATION *maintains a physically connected tree that contains all modules.*

Proof is omitted for lack of space. We just mention briefly an interesting case, in which the root attaches to a module in a way that a cycle is created. This happens if d is already part of S, but $c \neq P(d)$. The cycle is created because c becomes attached to d, but is always broken by detaching d from its parent.

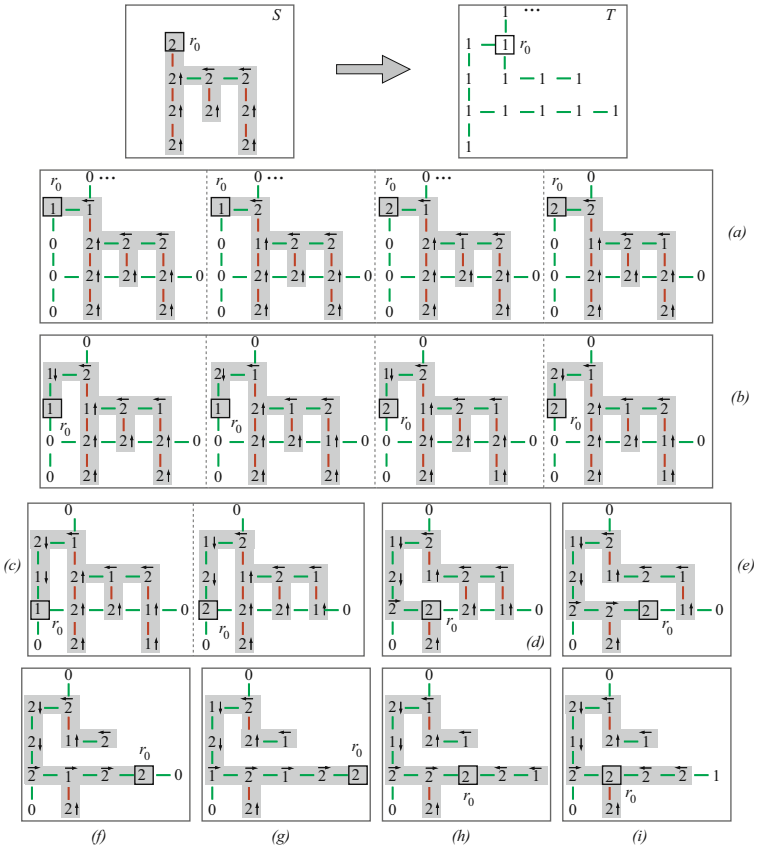


Fig. 6. Reconfiguring S into T: the top row shows S (after SOURCECLUSTER) and T. Subsequent figures show S (with its logical subtree S_ℓ shaded) *(a)* after TARGETGROW, with each intermediate step illustrated (DFS root update on the left and the subsequent 3 clustering steps on the right); *(b)* after TARGETGROW, with each intermediate step illustrated; *(c)* after TARGETGROW, with its two main steps (root update and root clustering) illustrated; *(d,e,f,g)* show the next 4 TARGETGROW steps; *(h)* after the next 2 TARGETGROW steps; *(i)* after the next TARGETGROW (note the rightmost 1-cell leaf getting disconnected from S_ℓ); the process continues.

In phase 1 of INPLACERECONFIGURATION SOURCECLUSTER produces a root-clustered tree containing r_0 in a 2-cell. We now show that phase 2 maintains this property in constant time, regardless of how r_0 moves.

Lemma 4.2. TARGETGROW *maintains S_ℓ as a root-clustered tree containing r_0 in a 2-cell. Furthermore, the procedure uses $O(1)$ parallel steps.*

Proof. The proof is rather similar to that of Lemma 3.5. Since $S_\ell \subseteq S$, it follows from Lemma 4.1 that S_ℓ is physically connected.

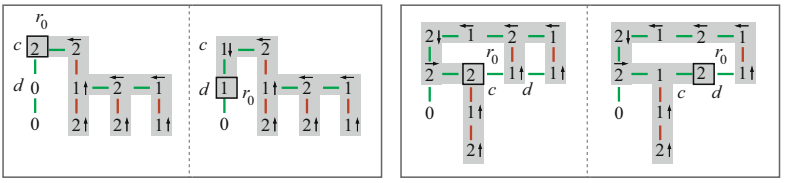


Fig. 7. DFS Root update (a) POPOUTLEAF(c,d) (b) TRANSFER(c,d).

Let S_ℓ^i denote the root-clustered tree that is input for TARGETGROW. In step 1 (DFS root update), S_ℓ^i will be modified according to any physical operations carried out (POPOUTLEAF and TRANSFER). By Lemma 4.1, these changes result in a tree, which we call S_ℓ^{i+1}.

Since step 1 only affects c and d, it follows that at the beginning of step 2, a long gap in S_ℓ^{i+1} must contain c, which becomes a 1-cell via POPOUTLEAF (see Fig. 7a), or via TRANSFER (see Fig. 7b).

We now show that the loop in step 2 of TARGETGROW iterates at most four times before our claim holds. Recall that, since S_ℓ^{i+1} was root-clustered, children of c are either leaves, 2-cells, or their children have that property.

Any *DetachLeaf* operation only trims visited 1-cell leaves from the tree and thus does not affect the root-clustered property of the tree. There are two cases for the number of CLUSTERSTEP applications required to terminate the loop:

1. S_ℓ^{i+1} *was obtained via* POPOUTLEAF *(step 1.1):* In this case c and d are 1-cells at the beginning of step 2. If all children of c are leaves or 2-cells, then in the first iteration of CLUSTERSTEP, c will become a 2-cell again. Otherwise, since S_ℓ^i was root-clustered, any non-leaf 1-cell child will become a 2-cell in the first iteration. Thus in the second iteration at the latest, c will become a 2-cell. Furthermore, just as described in Lemma 3.5, the subtree rooted at any child of c remains root-clustered after the first application of CLUSTERSTEP (in particular, for the highest-priority child which is the only one that changes). Similarly, by the time c becomes a 2-cell, the subtree rooted at c also becomes root-clustered. The third CLUSTERSTEP makes d a 2-cell root of a root-clustered tree, since all children of c must have been leaves or 2-cells to supply a module to c. The fourth CLUSTERSTEP makes c a 2-cell, which terminates the loop.
2. S_ℓ^{i+1} *was obtained via* TRANSFER *(step 1.2):* In this case d is already a 2-cell at the beginning of step 2 because of the TRANSFER operation in step 1.2. If c remains a 2-cell during the transfer, then S_ℓ^{i+1} is already root-clustered and the loop condition is satisfied. If c is a 1-cell, arguments similar to case 1 imply that after one application of CLUSTERSTEP, S_ℓ^{i+1} is root-clustered. A second application of CLUSTERSTEP makes c a 2-cell, which terminates the loop. □

Theorem 4.1. *The* INPLACERECONFIGURATION *algorithm can be implemented in* $O(n)$ *parallel steps.*

Proof. By Lemma 3.4, phase 1 uses $O(n)$ steps. Step 2 of InPlaceReconfiguration has $O(n)$ iterations, since DFS has $O(n)$ complexity. By Lemma 4.2, each iteration takes constant time. □

5 Observations

Matching lower bound: Transforming a horizontal line of modules to a vertical line requires a linear number of parallel steps, if each module can only displace one other and maximum velocity is constant.

3D: All of our techniques apply directly to 3D robots, once the top and bottom sides of cells are incorporated into our highest priority rule.

Labeled robots: Our algorithms are essentially unaffected if labels are assigned to modules. In TreeToPath, assume that the partially constructed canonical path is sorted. Then a new module m pushed through can bubble/tunnel to its position. When it gets there, the tail of the path must shift over, but this is straightforward involving propagation of one compressed unit, and does not interfere with other modules following m. For the in-place algorithm, T can first be constructed disregarding labels. A similar type of bubble-sort can then be applied, within T.

Telecube robots: The natural state of a telecube robot has atom arms contracted. There is no room to compress two modules into one cell. Thus an algorithm cannot commence with PushInLeaf operations, and it is not possible to physically exchange modules in adjacent cells while remaining in place. However, consider our first algorithm. We do not even need a SourceCluster phase, since all atoms are packed together. The root can transmit an instruction to a cell at maximum y-coordinate to act as root and immediately push out two of its atoms. For the construction of V, all analysis follows. It seems that labeled atoms within a module might become separated (for example, if the module is at a junction in a tree). Thus an extra step is used, to collect the root atoms at the bottom of V.

Exact in-place reconfiguration is impossible for labeled telecube robots. Thus the root cannot travel to any position within S. It might be possible to deal with this issue by requiring larger modules and designing a "reduced module shape" for the root (e.g., fewer atoms, using naturally expanded links). Instead, we could require that all modules have access to the map of T, which means any module can begin to expand T by filling adjacent 0-cells. Instead of advancing through nonempty cells of T physically, the root can just tell its neighbors to take over. Eventually a new root module would expand T at a different connected component of 0-cells.

Acknowledgements. We thank the other participants of the 2008 *Workshop on Reconfiguration* at the Bellairs Research Institute of McGill University for providing a stimulating research environment.

References

1. Aloupis, G., Collette, S., Damian, M., Demaine, E.D., Flatland, R., Langerman, S., O'Rourke, J., Ramaswami, S., Sacristán, V., Wuhrer, S.: Linear reconfiguration of cube-style modular robots. Computational Geometry: Theory and Applications (to appear)
2. Aloupis, G., Collette, S., Demaine, E.D., Langerman, S., Sacristán, V., Wuhrer, S.: Reconfiguration of cube-style modular robots using $O(\log n)$ parallel moves. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 342–353. Springer, Heidelberg (2008)
3. Butler, Z., Fitch, R., Rus, D.: Distributed control for unit-compressible robots: Goal-recognition, locomotion and splitting. IEEE/ASME Trans. on Mechatronics 7(4), 418–430 (2002)
4. Butler, Z., Rus, D.: Distributed planning and control for modular robots with unit-compressible modules. Intl. Journal of Robotics Research 22(9), 699–715 (2003)
5. Chirikjian, G., Pamecha, A., Ebert-Uphoff, I.: Evaluating efficiency of self-reconfiguration in a class of modular robots. Journal of Robotic Systems 13(5), 317–338 (1996)
6. Murata, S., Kurokawa, H.: Self-reconfigurable robots: Shape-changing cellular robots can exceed conventional robot flexibility. IEEE Robotics & Automation Magazine 14(1), 43–52 (2007)
7. Reif, J.H., Slee, S.: Optimal kinodynamic motion planning for self-reconfigurable robots between arbitrary 2D configurations. In: Robotics: Science and Systems Conference, Georgia Institute of Technology (2007)
8. Rus, D., Vona, M.: Crystalline robots: Self-reconfiguration with compressible unit modules. Autonomous Robots 10(1), 107–124 (2001)
9. Suh, J.W., Homans, S.B., Yim, M.: Telecubes: Mechanical design of a module for self-reconfigurable robotics. In: Proc. of the IEEE Intl. Conf. on Robotics and Automation, pp. 4095–4101 (2002)
10. Vassilvitskii, S., Yim, M., Suh, J.: A complete, local and parallel reconfiguration algorithm for cube style modular robots. In: Proc. of the IEEE Intl. Conf. on Robotics and Automation, pp. 117–122 (2002)
11. Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robots systems: Challenges and opportunities for the future. IEEE Robotics & Automation Magazine 14(1), 43–52 (2007)

Kinodynamic Motion Planning by Interior-Exterior Cell Exploration

Ioan A. Şucan and Lydia E. Kavraki

1 Introduction

Over the last two decades, motion planning [4, 15, 17] has grown from a field that considered basic geometric problems to a field that addresses planning for complex robots with kinematic and dynamic constraints [5]. Applications of motion planning have also expanded to fields such as graphics and computational biology [16].

Much of the recent progress in motion planning is attributed to the development of sampling-based algorithms [4, 17]. A sampling-based motion planning algorithm can only be probabilistically complete [9, 12], which means if a solution exists, it will be eventually found. One of the first successful sampling-based motion planners was the Probabilistic Roadmap Method (PRM) [10]. This method provided a coherent framework for many earlier works that used sampling and opened new directions for research [2]. In the case of realistic robots, taking dynamic constraints into account (kinodynamic motion planning) is a necessity. Sampling-based tree planners such as Rapidly-exploring Random Trees (RRT) [11, 19], Expansive Space Trees (EST) [6, 7] have been successfully used to solve such problems. These planners build a tree of motions in the state space of the robot and attempt to reach the goal state. Many variations of these planners exist as well (*e.g.*, [8, 18, 22]). More recent planners have been designed specifically for planning with complex dynamic constraints [14, 21]. The Path-Directed Subdivision Tree (PDST) planner [13, 14] has been used in the context of physics-based simulation as well.

This work presents a new motion planner designed specifically for handling systems with complex dynamics. This planning algorithm will be referred to as Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE). While there are other planners for systems with complex dynamics, KPIECE was designed with additional goals in mind. One such design goal is the ease of use for systems where only a forward propagation routine is available (that is, the simulation of the system

Ioan A. Suçan and Lydia E. Kavraki
Rice University, 6100 Main St., Houston TX, 77005, USA
e-mail: {isucan,kavraki}@rice.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 449–464.
springerlink.com

can be done forward in time). Another goal is that no state sampling and no distance metric are required. These limitations make KPIECE particularly well suited for complex systems described by physical models instead of equations of motion, since in such cases only forward propagation is available and sampling of states is in general expensive. Since KPIECE does not need to evaluate distance between states, it is also well suited for systems where a distance metric is hard to define or when the goal is not known until it is actually reached. It is typical for sampling-based tree planners to spend more than 90% of their computation extending the trees they build using forward propagation. Since physics simulation is considerably more expensive than integration of motion models, it is essential to use as few propagation steps as possible. This was a major motivation behind this work and as will be shown later, KPIECE provides significant computational improvements over previous methods (up to two orders of magnitude), which allows tackling more complex problems that could not be previously addressed. Since motion planning is usually a subproblem of a more complex task, it is generally desirable to have fast methods for the computation of motion plans. To this end, KPIECE was also designed with shared memory parallelism in mind and the developed implementation can take advantage of the emerging multi-core technology. The implementation can use a variable number of processors and shows super-linear speedup in some cases. The combination of obtained speedup and physics-based simulation, could make KPIECE fast and accurate enough to be applicable in real-time motion planning for complex reactive robotic systems.

The rest of content is organized as follows: Section 2 presents the motion planning problem in more detail, Section 3 contains a description of the proposed algorithm, and Section 4 presents experiments using KPIECE. The parallel implementation is discussed in Section 5. Conclusions and future work are in Section 6.

2 Problem Definition

An instance of the motion planning problem addressed here can be formally defined by the tuple $S = (Q, U, I, F, f)$ where Q is the state space, U is the control space, $I \subset Q$ is the set of initial states, and $F \subset Q$ is the set of final states. The dynamics are described by a forward propagation routine $f : Q \times U \rightarrow TgQ$, where TgQ is the tangent space of Q (f does not need to be explicit). A solution to a motion planning problem instance consists of a sequence of controls $u_1, \ldots, u_n \in U$ and times $t_1, \ldots, t_n \in \mathbb{R}^{\geq 0}$ such that $q_0 \in I$, $q_n \in F$ and $q_k, k = 1, \ldots, n$, can be obtained sequentially by integration of f.

For the purposes of this work, the function f is computed by a physics simulator. In particular, an open source library called Open Dynamics Engine (ODE) [24], is used. Instead of equations of motion to be integrated, a model of a robot and its environment needs to be specified. Although simulation incurs more computational costs than simple integration, the benefits outweigh the costs: increased accuracy is available since physics simulators take into account more dynamic properties of the robot (such as gravity, friction) and constructing models of systems is easier and less

error prone than deriving equations of motion. Limited numerical precision will still be a problem regardless of how f is computed. However, as robotic systems become more complex, physics-based simulation becomes a necessity.

It is sometimes the case that due to the high dimensionality of the state space Q, a projection space $\mathscr{E}(Q)$ is used for various computations the motion planning algorithm performs ($\mathscr{E}$ can be the identity transform). Finding such a projection $\mathscr{E}$ is a research problem in itself. In this work it is assumed that such a projection $\mathscr{E}$ is available, when needed. Section 4.1 presents simple cases of $\mathscr{E}$ used in this work.

3 Algorithm

A high-level description of the algorithm is provided before the details are presented. KPIECE iteratively constructs a tree of motions in the state space of the robot. Each motion $\mu = (s,u,t)$ is identified by a state $s \in Q$, a control $u \in U$ and a duration $t \in \mathbb{R}^{\geq 0}$. The control u is applied for duration t from s to produce a motion. It is possible to split a motion $\mu = (s,u,t)$ into $\mu_1 = (s,u,t_a)$ followed by $\mu_2 = (\int_{t_0}^{t_0+t_a} f(s(\tau),u)d\tau, u, t_b)$, where $s(\tau)$ identifies the state at time τ and $t_a + t_b = t$. In this exploration process, it is important to cover as much of the state space as possible, as quickly as possible. For this to be achieved, estimates of the coverage of Q are needed. To this end, the discretization described in Section 3.1 is employed. When a less covered area of the state space is discovered, the tree of motions is extended in that area. This process is iteratively executed until a stopping condition is satisfied.

3.1 Discretization

During the course of its run, the motion planner must decide which areas of the state space merit further exploration. As the size of the tree of motions increases, making this decision becomes more complex. There are various strategies to tackle this problem (*e.g.*, [7, 14, 19, 21, 22]). The approach taken in this work is to construct a discretization that allows the evaluation of the coverage of the state space. This discretization consists of k levels $\mathscr{L}_1, ..., \mathscr{L}_k$, as shown in Fig. 1. Each of these levels is a grid where cells are polytopes of fixed size. The number of levels and cell sizes are predefined, however, cells are instantiated only when they are needed. The purpose of these grids is to cover the area of the space that corresponds to the area spanned by the tree of motions. Each of the levels provides a different resolution for evaluating the coverage. Coarser resolution (higher levels) can be used initially to find out roughly which area is less explored. Within this area, finer resolutions (lower levels) can then be employed to more accurately detect less explored areas. The following is a formal definition of a k-level discretization:

- for $i \in \{1,...,k\} : \mathscr{L}_i = \{p_i | p_i \text{ is a cell in the grid at level } i\}$
- for $i \in \{2,...,k\} : \forall p \in \mathscr{L}_i, \mathscr{D}_p = \{q \in \mathscr{L}_{i-1} | q \subset p\}$, such that
  - $\forall p \in \mathscr{L}_i, \mathscr{D}_p \neq \emptyset$

- $\bigcup_{p \in \mathscr{L}_i} \mathscr{D}_p = \mathscr{L}_{i-1}$
- $\forall p, q \in \mathscr{L}_i, p \neq q \rightarrow \mathscr{D}_p \cap \mathscr{D}_q = \emptyset$

The tree of motions exists in the state space Q, but since the dimension of this space may be too large, the discretization is typically imposed on a projection of the state space, $\mathscr{E}(Q)$. The use of such a projection $\mathscr{E}(Q)$ was also discussed in [13, 21]. An important result we show in this work is that simple projections work for complex problems. For any motion μ, each level of discretization contains a cell that μ is part of. A motion μ is considered to be part of a grid cell p if there exists a state s along μ such that the projection $\mathscr{E}(s)$ is inside the bounding box of cell p. If a motion spans more than one cell at the same level of discretization, it is split into smaller motions such that no motions cross cell boundaries. This invariant is maintained to make sure each motion is accounted for only once. For every motion μ, there will be exactly one cell at every level of discretization that μ is part of. This set of cells forms a tuple $\mathbf{c} = (p_1, ..., p_k), p_i \subset p_{i+1}, p_i \in \mathscr{L}_i$ and will be referred to as the "cell chain" for μ. Since cells in $\mathscr{L}_1$ will determine whether a motion is split, we augment the definition of the discretization:

- $\forall p \in \mathscr{L}_1, \mathscr{M}_p = \{m_i | m_i \text{ is a motion contained in } p\}$

For all $p \in \mathscr{L}_1$ we say p contains $\mathscr{M}_p$ and for all $p \in \mathscr{L}_i, i > 1$ we say p contains $\mathscr{D}_p$. While the discretization spans the potentially very large projection space $\mathscr{E}(Q)$, cells are instantiated only when a motion that is part of them is found, hence the grids are not fully instantiated. This allows the motion planner to limit its use of memory to reasonable amounts. The size of the grid cells is discussed in Section 3.4.

A distinguishing feature of `KPIECE` is the notion of interior and exterior cells. A cell is considered exterior if it has less than $2n$ instantiated neighboring cells (diagonal neighboring cells are ignored) at the same level of discretization, where n is the dimension of $\mathscr{E}(Q)$. Cells with $2n$ neighboring cells are considered interior (there can be no more than $2n$ non-diagonal neighboring cells in an n-dimensional

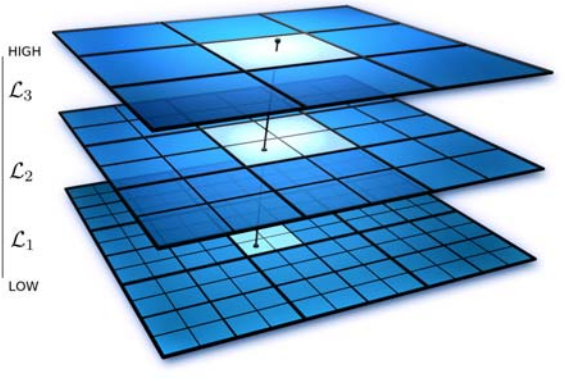


Fig. 1. An example discretization with three levels. The line intersecting the three levels defines a cell chain. Cell sizes at lower levels of discretization are integer multiples of the cell sizes at the level above.

space). As the algorithm progresses and new cells are created, some exterior cells will become interior. When larger parts of the state space are explored, most cells will be interior. However, for very high dimensional spaces, to avoid having only exterior cells, the definition of interior cells can be relaxed and cells can be considered interior before all $2n$ neighboring cells are instantiated. For the purposes of this work, this relaxation was not necessary.

With these notions in place, a measure of coverage of the state space can be defined. For a cell $p \in \mathscr{L}_1$, the coverage is simply the sum of the durations of the motions in $\mathscr{M}_p$. For higher levels of discretization, the coverage of a cell $p \in \mathscr{L}_i, i > 1$ is the number of instantiated cells in $\mathscr{D}_p$.

3.2 Algorithm Execution

A run of the KPIECE algorithm proceeds as described in Algorithm 1. The tree of motions is initialized to a motion defined by the initial state q_{start}, a null control and duration 0 [line 1]. Adding this motion to the discretization will create exactly one exterior cell for every level of discretization [lines 2,3].

At every iteration, a cell chain $\mathbf{c} = (p_1, ..., p_k)$ is sampled. This means $p_i \in \mathscr{L}_i$ will have to be selected, from p_k to p_1, as will be shown later. It is important to note here that "samples" in the case of KPIECE are chains of cells. This can be regarded as a natural progression (selection of "volumes") from the selection of states ("points") as in the case of RRT and EST, and selection of motions ("curves") as in the case of PDST. Our experiments show that selecting chains of cells benefits from the better estimates of coverage that can be maintained for cells at each level, as opposed to estimates for single motions or states. Sampling a cell chain $\mathbf{c} = (p_1, ..., p_k)$ is a k-step process that proceeds as follows: the decision to expand from an interior or exterior cell is made [line 5], with a bias towards exterior cells. An instantiated cell, either interior or exterior, is then deterministically selected from $\mathscr{L}_k$, according to the cell importance (higher importance first). The idea of deterministic selection was inspired by [14], where it has been successfully used. The importance of a cell p, regardless of the level of discretization it is part of, is computed as:

$$Importance(p) = \frac{\log(\mathscr{I}) \cdot \texttt{score}}{\mathscr{S} \cdot \mathscr{N} \cdot \mathscr{C}}$$

where $\mathscr{I}$ stands for the number of the iteration at which p was created, score is initialized to 1 but may later be updated to reflect the exploration progress achieved when expanding from p, $\mathscr{S}$ is the number of times p was selected for expansion (initialized to 1), $\mathscr{N}$ is the number of instantiated neighboring cells at the same level of discretization, and $\mathscr{C}$ is a positive measure of coverage for p, as described at the end of Section 3.1.

Once a cell p is selected, if $p \notin \mathscr{L}_1$, it means that further levels of discretization can be used to better identify the more important areas within p. The selection process continues recursively: an instantiated cell from $\mathscr{D}_p$ is subsequently selected using the method described above until the last level of discretization is reached and

the sampling of the cell chain is complete. At the last level, a motion μ from $\mathscr{M}_p$ is picked according to a half-normal distribution [line 6]. The half-normal distribution is used because order is preserved when adding motions to a cell and motions added more recently are preferred for expansion. A state s along μ is then chosen uniformly at random [line 7]. Expanding the tree of motions continues from s [line 9].

The controls applied from s are selected uniformly at random from U [line 8]. The random selection of controls is what is typically done if other means of control selection are not available. This choice is not part of the proposed algorithm, and can be replaced by other methods, if available.

If the tree expansion was successful, the newly obtained motion is added to the tree of motions and the discretization is updated [lines 11,13]. An estimate of the achieved progress is then computed. For every level of discretization j, the coverage of some cells may have increased:

$$\Delta\mathscr{C}_j = \Sigma_{p\in\mathscr{L}_j}\Delta p, \text{ where } \Delta p = \text{increase in coverage of } p$$

$$P_j = \alpha + \beta \cdot (\text{ratio of } \Delta\mathscr{C}_j \text{ to time spent computing simulations}).$$

P_j is considered the progress at level j [line 16]. The values α and β are implementation specific and should be chosen such that $P_j > 0$, and $P_j \geq 1$ implies good progress. The offset α needs to be strictly positive since the increase in coverage can be 0 (*e.g.*, in case of an immediate collision). The value of P_j is also used as a penalty if not enough progress has been made ($P_j < 1$): the cell at level j in the selected cell chain has its `score` multiplied by P_j [line 17]. If good progress has been made ($P_j \geq 1$), the value of P_j is ignored, since we do not want to over-commit to specific areas of the space.

Algorithm 1. KPIECE(q_{start}, $N_{iterations}$)

1: Let μ_0 be the motion of duration 0 containing solely q_{start}
2: Create an empty `Grid` data-structure G
3: G.ADDMOTION(μ_0)
4: **for** $i \leftarrow 1...N_{iterations}$ **do**
5: Select a cell chain **c** from G, with a bias on exterior cells (70% - 80%)
6: Select μ from **c** according to a half normal distribution
7: Select s along μ
8: Sample random control $u \in U$ and simulation time $t \in \mathbb{R}^+$
9: Check if any motion $(s,u,t_\circ)$, $t_\circ \in (0,t]$ is valid (forward propagation)
10: **if** a motion is found **then**
11: Construct the valid motion $\mu_\circ = (s,u,t_\circ)$ with $t_\circ$ maximal
12: If $\mu_\circ$ reaches the goal region, **return** path to $\mu_\circ$
13: G.ADDMOTION($\mu_\circ$)
14: **end if**
15: **for** every level $\mathscr{L}_j$ **do**
16: $P_j = \alpha + \beta \cdot$ (ratio of increase in coverage of $\mathscr{L}_j$ to simulated time)
17: Multiply the `score` of cell p_j in **c** by P_j if and only if $P_j < 1$
18: **end for**
19: **end for**

3.3 Implementation Details

To aid in the implementation of the KPIECE algorithm, an efficient grid data-structure (`Grid`) was defined. `Grid` maintains the list of cells it contains, grouped into interior and exterior, sorted according to their importance. To maintain the lists of interior and exterior cells sorted, binary heaps are used. For every cell p, `Grid` also maintains some additional data: another `Grid` instance (stands for $\mathscr{D}_p$), for all but the lowest level of discretization, and for the lowest level of discretization, an array of motions (stands for $\mathscr{M}_p$). Algorithm 2 shows the steps for adding motions to `Grid`.

Algorithm 2. ADDMOTION(s,u,t)

1: Split (s,u,t) into motions $\mu_1,...,\mu_k$ such that $\mu_i, i \in \{1,...,k\}$ does not cross the boundary of any cell at the lowest level of discretization
2: **for** $\mu_\circ \in \{\mu_1,...,\mu_k\}$ **do**
3: Find the cell chain corresponding to $\mu_\circ$
4: Instantiate cells in the chain, if needed
5: Add $\mu_\circ$ to the cell at the lowest level in the chain
6: Update coverage measures and lists of interior and exterior cells, if needed
7: **end for**

3.4 Computing the Discretization

An important issue not discussed so far is the selection of number of levels in the discretization and the grid cell sizes. This section presents a method to compute these cell sizes if the discretization is assumed to consist of only $\mathscr{L}_1$ (a one-level discretization).

While `KPIECE` is running, we can keep track of averages of how many motions per cell there are, how many parts a motion is split into before it is added to the discretization, and the ratio of interior to exterior cells. While we do not know how to compute optimal values for these statistics (if they exist), there are certain ranges that may work better than others. In particular, the authors have observed that for good performance the following should hold:

- Less than 10% of the motions cover more than 2 cells in one simulation time-step. This value should be in general less than 1% as the event occurs only when the velocity of the robotic system is very high.
- At least 50% of the motions need to be 3 simulation time-steps or longer.
- Average number of parts in which a motion is split should be larger than 1 but not higher than 4.
- As the algorithm progresses, at least some interior cells need to be created.
- The average number of samples per cell should be in the range of tens to hundreds.

Based on collected statistics and these observations, it can be automatically decided whether the cell sizes used for $\mathscr{L}_1$ are good, too large or too small. This

information is reported for each dimension of the space. If the used cell size is too small or too large in some dimension, the size in that dimension is increased or decreased, respectively, by a factor larger than 1 and the algorithm is restarted. This process usually converges in 2 or 3 iterations.

These statistics do not offer any information about higher levels of discretization, nor do they provide information about how many levels of discretization should be used. The presented constants are implementation specific, but they seem not to vary across the examined robotics systems.

4 Experiments

The presented algorithm was benchmarked against well-known efficient algorithms (RRT, EST, PDST) with three different robotic systems, in different environments. For modeling the robots, the ODE [24] physics-based simulator was used. For the implementations of RRT [19] and EST [6], the OOPSMP framework was used [20]. A plugin for linking OOPSMP with the ODE simulator was developed by the authors. The authors did their best to tune the parameters of both RRT and EST. For RRT, a number of different metrics were tested for each robot and experiments are presented with the metric that performed best. In addition, random controls were selected instead of attempting to find controls that take the robotic system toward a desired state, as this strategy seemed to provide better results. For EST, the nodes to expand from were selected both based on their degree [7] and based on a grid subdivision of the state space [22]. Experiments are shown for the selection strategy that performed best. PDST and KPIECE were implemented by the authors. A projection was defined for each robot and the same projection was used for both PDST and KPIECE. In addition to the projection, KPIECE needs a discretization to be defined for each robot. When comparing with other algorithms, only discretizations computed as shown in Section 3.4 were used. Separate experiments are shown when using empirically chosen discretizations with multiple levels. Explanations on how these multiple levels were chosen are given later in this section. No goal biasing was used for any of the algorithms. However, separate experiments are shown for RRT with biasing (RRT_b). All implementations are in C++ and were tested on the Rice Cray XD1 Cluster, where each machine runs at 2.2 Ghz and has 8 GB RAM. For each system and each of its environments, each algorithm was executed 50 times. The best two and worse two results in terms of runtime were discarded and the results of the remaining 46 runs were averaged. The time limit was set to one hour and the memory limit was set to 2 GB. If an execution exceeded the time or memory limit, it was considered successful with execution time equal to the time limit.

4.1 Robots

Three different robots were used in benchmarking the planner, to show its generality: a modular robot, a car, and a blimp. These robots have been chosen to be different in terms of the difficulties they pose to a motion planner. Details on what

these difficulties are follow in the next paragraphs. ODE version 0.9 was used to model the robots. The used simulation step size was 0.05s.

Modular Robot

The model for this robot was implemented in collaboration with Mark Yim, and characterizes the CKBot modules [23]. Each CKBot module contains one motor. An ODE model for serially linked CKBot modules has been created [5]. The task is to compute the controls for lifting the robot from a vertical down position to a vertical up position for varying number of modules, as shown in Fig. 2. Each module adds one degree of freedom. The controls represent torques that are applied by the motors inside the modules. The difficulty of the problem lies in the high dimensionality of the control and state spaces as the number of modules increases, and in the fact that at maximum torque, the motors in the modules are only able to statically lift approximately 5 modules. This is why the planner has to find swinging motions to solve the problem. The employed projection $\mathscr{E}$ was a 3-dimensional one, the first two dimensions being the (x,z) coordinates of the last module (x,z is the plane observed in Fig. 2) and the third dimension, the square root of the sum of squares of the rotational velocities of all the modules. The environments the system was tested in are shown in Fig. 2.

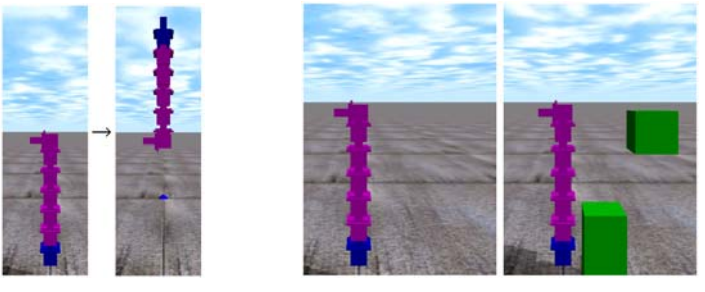


Fig. 2. Left: start and goal configurations. Right: environments used for the chain robot (7 modules). Experiments were conducted for 2 to 10 modules. In the case without obstacles, the environments are named ch1-x where x stands for the number of modules used in the chain. In the case with obstacles, the environments are named ch2-x.

Car Robot

A model of a car [4] was created as well. The model is fairly simple and consists of five parts: the car body and four wheels. Since ODE does not allow for direct control of accelerations, desired velocities are given as controls for the forward velocity and steering velocity (as recommended by the developers of the library). These desired velocities go together with a maximum allowed force. The end result is that the car will not be able to achieve the desired velocities instantly, due to the limited force. In effect, this makes the system a second order one. The employed projection $\mathscr{E}$ was the (x,y) coordinates of the center of the car body. The environments the system was tested in are shown in Fig. 3.

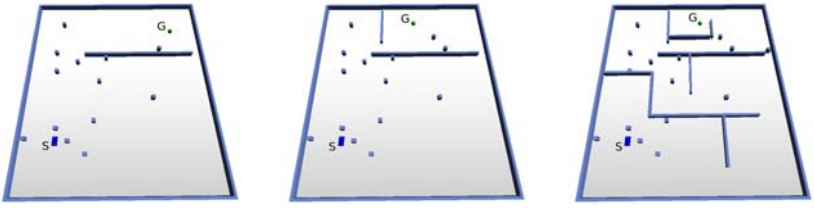


Fig. 3. Environments used for the car robot (cr-1, cr-2, cr-3). Start and goal configurations are marked by "S" and "G".

Blimp Robot

The third robot that was tested was a blimp robot [14]. The motion in this case is executed in a 3D environment. This robot is particularly constrained in its motion: the blimp must always apply a positive force to move forward (slowing down is caused by friction), it must always apply an upward force to lift itself vertically (descending is caused by gravity) and it can turn left or right along the direction of forward motion. Since `ODE` does not include air friction, a Stokes model of drag was implemented for the blimp. The employed projection $\mathscr{E}$ was the (x, y, z) coordinates of the blimp's center. The environments the system was tested in are shown in Fig. 4.

4.2 Results

In terms of runtime, when compared to other algorithms such as `RRT`, `EST`, and `PDST`, Table 1 shows significant computational gains for `KPIECE`. In particular, as the dimensionality of the problem increases, `KPIECE` does better. For simple problems however, other algorithms can be faster (*e.g.*, `RRT` for ch1-3). The presented speedup values are consistent with the time spent performing simulations, which serves to prove that the computational improvements are obtained by minimizing the usage of the physics-based simulator. Since physics simulation takes up around 90% of the execution time, computational gain will be observed with purely geometric planning as well, where forward integration is replaced by collision detection.

While the results shown in Table 1 are computed with a one-level discretization, for some problems, better results can be obtained using multiple levels of

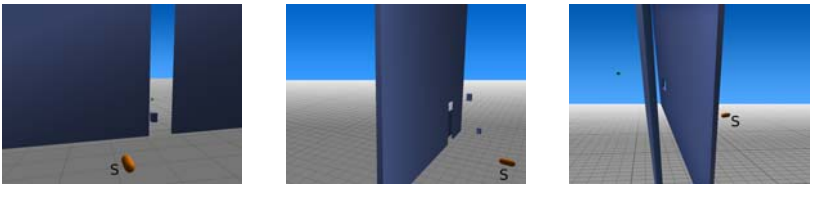


Fig. 4. Environments used for the blimp robot (bl-1, bl-2, bl-3). Start configurations are marked by "S". The blimp has to pass between the walls and through the hole(s), respectively.

Table 1. Speedup achieved by KPIECE over other algorithms for four different problems. If one of the other algorithms was unable to solve the problem in at least 10% of the cases, "—" is reported. KPIECE was configured with an automatically computed one-level discretization, as described in Section 3.4.

	RRT	RRT$_b$	EST	PDST
ch1-2	1.1	3.5	2.2	2.5
ch1-3	0.8	2.1	1.0	3.6
ch1-4	1.5	3.9	1.8	9.6
ch1-5	4.1	3.7	14.4	15.6
ch1-6	13.4	9.6	946.8	42.5
ch1-7	58.5	196.3	—	238.1
ch1-8	—	—	—	—

	RRT	RRT$_b$	EST	PDST
ch2-5	18.3	23.4	—	13.0
ch2-6	35.0	255.7	—	23.0
ch2-7	45.7	124.7	—	81.3
ch2-8	—	—	—	5.9
ch2-9	—	—	—	—

	RRT	RRT$_b$	EST	PDST
cr-1	3.2	3.1	27.7	7.9
cr-2	5.0	3.5	16.1	9.7
cr-3	8.7	14.8	15.5	13.1
bl-1	1.6	2.2	3.1	3.3
bl-2	6.4	7.2	8.7	9.4
bl-3	4.5	7.3	5.7	7.5

discretization. To show this, for each robot, twelve discretizations are defined. First, a one-level discretization (consists only of $\mathscr{L}_1$) is computed as discussed in Section 3.4. Two more one-level discretizations with half and double the cell volume of the computed discretization's cells are then constructed (cell sides shortened and lengthened proportionally, in each dimension). For each of these three one-level discretizations, three more two-level discretizations (consist of $\mathscr{L}_1$, $\mathscr{L}_2$) are defined: ones that have the same $\mathscr{L}_1$, but $\mathscr{L}_2$ consists of cells with sizes of 10, 15, and 20 times the cell sizes of $\mathscr{L}_1$. Table 2 shows the speedup obtained when employing the best of the nine defined two-level discretizations. As we can see, in most cases there are benefits to using two discretization levels. Experiments with more than two levels of discretization were conducted as well, but the performance started to decrease and the results are not presented here. The defined discretizations can also be used to evaluate the sensitivity of KPIECE to the defined grid sizes. As shown in Fig. 5, the runtimes of the algorithm for the different discretizations are relatively close to one another (within a factor of 2.3). This implies that the algorithm is not overly sensitive to the defined discretization and thus approximating good cell sizes is sufficient. Nevertheless, finding good discretizations remains an open problem.

4.3 Discussion of Experimental Results

In the previous section we have shown the computational benefits of using KPIECE over other algorithms. There are a few key details that make KPIECE distinct: the

Table 2. Speedup achieved by KPIECE when using a two-level discretization relative to the automatically computed one-level discretization. For ch1-10 and ch2-10, a solution was found only with the two-level discretization so no speedup is reported.

ch1-2: 0.9	ch1-7: 2.2	ch2-5: 0.9	cr-1: 1.0	bl-1: 1.3
ch1-3: 1.1	ch1-8: 2.0	ch2-6: 1.1	cr-2: 1.0	bl-2: 1.1
ch1-4: 0.9	ch1-9: 1.3	ch2-7: 1.7	cr-3: 0.7	bl-3: 1.8
ch1-5: 0.7		ch2-8: 0.5		
ch1-6: 2.5		ch2-9: 1.2		

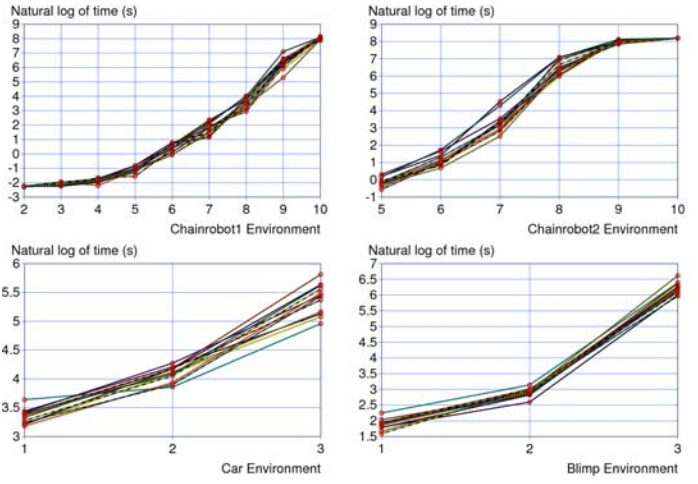


Fig. 5. Logarithmic runtimes with twelve different discretizations for the ch1, ch2, cr, and bl.

sampling of a chain of cells, the grouping of cells into interior and exterior, and the progress evaluation, based on increase in coverage. While the sampling of cell chains is an inherent part of the algorithm, the other two features can be easily disabled. This allows us to evaluate the contribution of these components individually.

Fig. 6 shows that both progress evaluation and cell distinction contribute to reducing the runtime of `KPIECE`. While these components do not seem to help for easier problems (bl-1), their contribution is important for harder problems (cr-3, bl-3). In particular, the cell distinction seems to be the more important component as the problems get harder. This is to be expected, since the distinction allows the algorithm to focus exploration on the boundary of the explored space, while ignoring the larger, already explored interior volume.

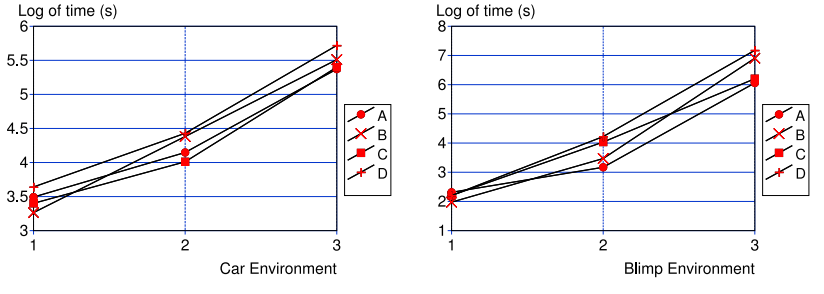


Fig. 6. Logarithmic runtime for `KPIECE` with various components disabled, on 2-dimensional and 3-dimensional projections (cr and bl) with the automatically computed one-level discretization. `A` = no components disabled, `B` = no cell distinction, `C` = no progress evaluation, `D` = no cell distinction and no progress evaluation.

5 Parallel Implementation

The presented algorithm was also implemented in a shared memory parallel framework. While previous work has shown significant improvements with embarrassingly parallel setups [1, 3], this work attempts to take the emerging multi-core technology into account and use it as an advantage. Instead of running the algorithm multiple times and stopping when one of the active instances found a solution as in [1, 3], KPIECE uses multiple threads to build the same tree of motions (threads can continue expanding from cells instantiated by other threads). Synchronization points are used to ensure correct order of execution. This execution format will become more important in the next few years as the number of computing cores and memory bandwidth increase. Since each computing thread starts from a different random seed, the chances of all seeds being unfavourable decrease. If a single thread finds a path through a narrow passage, the rest of the threads will immediately use this information as well. This setup also reduces the variance in the average runtime of the algorithm. It is important to note this proposed parallelization scheme can be applied to other sampling-based algorithms as well.

All experiments presented in previous sections were conducted when using the planner in single-threaded mode. Table 3 shows the speedup achieved by the motion planner when using one to four threads on a four-core machine. The achieved speedup is super-linear in some cases, a known characteristic of sampling-based motion planners. When comparing to the speedup obtained with an embarrassingly parallel setup, shown in Table 4, we notice that better runtimes are obtained with our suggested setup. In addition, total memory requirements in our suggested setup do not increase significantly as the number of processors is increased.

Table 3. Speedup achieved by KPIECE with multiple threads for 2-dimensional and 3-dimensional projections (cr and bl). KPIECE was configured with an automatically computed one-level discretization, as described in Section 3.4.

Threads	cr-1	cr-2	cr-3	bl-1	bl-2	bl-3
2	1.7	2.0	2.6	2.3	1.9	1.4
3	2.8	2.7	3.0	2.9	3.0	2.2
4	3.9	3.6	4.4	3.5	3.2	3.1

Table 4. Speedup achieved by KPIECE in embarrassingly parallel mode.

Threads	cr-1	cr-2	cr-3	bl-1	bl-2	bl-3
2	1.3	1.5	1.6	1.5	1.6	1.3
3	1.5	1.8	1.8	1.8	1.9	1.4
4	1.7	2.1	2.0	2.2	3.0	1.5

6 Conclusions and Future Work

We have presented KPIECE, a sampling-based motion planning algorithm designed for complex systems where physics-based simulation is needed. This algorithm does

not need a distance metric or a way to sample states. It does however require a projection of the state space and the specification of a discretization. At this point we recommend that the projection is defined by the user. As shown in our experiments, even simple intuitive projections work for complex problems. The discretization is an additional requirement when compared to other state-of-the-art algorithms. The algorithm's performance is not drastically affected by the discretization and a method to automatically compute one-level discretizations was presented. When using an automatically computed one-level discretization, KPIECE was compared to other popular algorithms, and shown to provide significant computational speedup. In addition, the provided shared memory parallel implementation seems to give better results than the embarrassingly parallel setup.

KPIECE is the result of a combination of ideas. Some of these ideas are new, some are inspired by previous work. In previous work, we have encountered state [7, 11] and motion sampling [14]; KPIECE takes this further and uses cell chain sampling. We have also seen progress evaluation [21], deterministic sample selection [14], use of physics-based simulation [13], and use of additional data-structures for estimation of coverage [14, 21, 22]. KPIECE implements variants of these ideas, combined with new ideas like distinction between interior and exterior cells, to obtain an algorithm that works well in a parallel framework. The result is a more accurate and efficient method that can solve problems previous methods could not.

It is conjectured that KPIECE is probabilistically complete: in a bounded state space, the number of cells is finite. Since with every selection, the importance of a cell can only decrease, every cell will be selected infinitely many times during the course of an infinite run. Every motion in a cell has positive probability of being selected, which makes the number of selections of each motion in the tree of motions be infinite as well. By the completeness of PDST [13], KPIECE is likely to be probabilistically complete. A formal proof is left for future work.

Further work is needed for better automatic computation of the employed discretization. Automatic computation of the used state space projection would be beneficial as well, not only for KPIECE, but for other algorithms that require such a projection. Furthermore, it would be interesting to push the limits of this method to harder problems.

Acknowledgements. This work was supported in part by NSF IIS 0713623 and Rice University funds. The experiments were run on equipment obtained by NSF CNS 0454333 and NSF CNS 0421109 in partnership with Rice University, AMD and Cray. The authors would like to thank Mark Yim and Jonathan Kruse for their help in defining the CKBot ODE model and Marius Şucan for drawing the example discretization.

References

1. Amato, N.M., Dale, L.K.: Probabilistic roadmap methods are embarrassingly parallel. In: IEEE Intl. Conf. on Robotics and Automation, Detroit, USA, May 1999, pp. 688–694 (1999)

2. Barraquand, J., Kavraki, L.E., Latombe, J.-C., Li, T.-Y., Motwani, R., Raghavan, P.: A random sampling scheme for robot path planning. Intl. Journal of Robotics Research 16(6), 759–774 (1997)
3. Caselli, S., Reggiani, M.: Randomized motion planning on parallel and distributed architectures. In: Rolim, J.D.P. (ed.) IPPS-WS 1999 and SPDP-WS 1999. LNCS, vol. 1586, pp. 297–304. Springer, Heidelberg (1999)
4. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
5. Şucan, I.A., Kruse, J.F., Yim, M., Kavraki, L.E.: Kinodynamic motion planning with hardware demonstrations. In: Intl. Conf. on Intelligent Robots and Systems, September 2008, pp. 1661–1666 (2008)
6. Hsu, D., Kindel, R., Latombe, J.-C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Intl. Journal of Robotics Research 21(3), 233–255 (2002)
7. Hsu, D., Latombe, J.-C., Motwani, R.: Path planning in expansive configuration spaces. In: IEEE Intl. Conf. on Robotics and Automation, vol. 3, April 1997, pp. 2719–2726 (1997)
8. Jaillet, L., Yershova, A., LaValle, S.M., Siméon, T.: Adaptive tuning of the sampling domain for dynamic-domain rrts. In: Intl. Conf. on Intelligent Robots and Systems (2005)
9. Kavraki, L.E., Latombe, J.-C., Motwani, R., Raghavan, P.: Randomized query processing in robot path planning. Journal of Computer and System Sciences 57(1), 50–60 (1998)
10. Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.: Probabilistic roadmaps for path planning in high dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
11. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. In: IEEE Intl. Conf. on Robotics and Automation (2000)
12. Ladd, A., Kavraki, L.: Measure theoretic analysis of probabilistic path planning. IEEE Transactions on Robotics and Automation 20(2), 229–242 (2004)
13. Ladd, A.M.: Direct Motion Planning over Simulation of Rigid Body Dynamics with Contact. PhD thesis, Rice University, Houston, Texas (December 2006)
14. Ladd, A.M., Kavraki, L.E.: Fast tree-based exploration of state space for robots with dynamics. In: Algorithmic Foundations of Robotics VI, pp. 297–312. Springer, Heidelberg (2005)
15. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers, Boston (1991)
16. Latombe, J.-C.: Motion planning: A journey of robots, molecules, digital actors, and other artifacts. Intl. Journal of Robotics Research 18(11), 1119–1128 (1999)
17. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006), http://planning.cs.uiuc.edu/
18. LaValle, S.M., Kuffner, J.: Rapidly-exploring random trees: Progress and prospects. New Directions in Algorithmic and Computational Robotics, 293–308 (2001)
19. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Intl. Journal of Robotics Research 20(5), 378–400 (2001)
20. Plaku, E., Bekris, K.E., Kavraki, L.E.: OOPS for Motion Planning: An Online Open-source Programming System. In: IEEE Intl. Conf. on Robotics and Automation, Rome, Italy, pp. 3711–3716 (2007)
21. Plaku, E., Vardi, M.Y., Kavraki, L.E.: Discrete search leading continuous exploration for kinodynamic motion planning. In: Robotics: Science and Systems, Atlanta, Georgia, pp. 313–320 (2007)

22. Sánchez, G., Latombe, J.-C.: A single-query bi-directional probabilistic roadmap planner with lazy collision checking. Intl. Journal of Robotics Research, 403–407 (2003)
23. Sastra, J., Chitta, S., Yim, M.: Dynamic rolling for a modular loop robot. Intl. Journal of Robotics Research 39, 421–430 (2008)
24. Smith, R.: Open dynamics engine, http://www.ode.org

Part VIII
Stochastic Methods in Planning

Control of Probabilistic Diffusion in Motion Planning

Sébastien Dalibard and Jean-Paul Laumond

Abstract. The paper presents a method to control probabilistic diffusion in motion planning algorithms. The principle of the method is to use on line the results of a diffusion algorithm to describe the free space in which the planning takes place. Given that description, it makes the diffusion go faster in favoured directions. That way, if the free space appears as a small volume around a submanifold of a highly dimensioned configuration space, the method overcomes the usual limitations of diffusion algorithms and finds a solution quickly. The presented method is theoretically analyzed and experimentally compared to known motion planning algorithms.

1 Problem Statement, Related Work and Contribution

1.1 General Framework

Motion planning problems have been intensively studied in the last decades, with applications in many diverse areas, such as robot locomotion, part disassembly problems, digital actors in computer animation, or even protein folding and drug design. For comprehensive overviews of motion planning problems and methods, one can refer to [10], [1] and [13].

In the past fifteen years, two kinds of configuration space (*CS*) search paradigms have been investigated with success.

- The *sampling* approach, first introduced in [8] as probabilistic roadmaps (PRM), consists in computing a graph, or a roadmap, whose vertices are collision free configurations, sampled at random in the free space and whose edges reflect the existence of a collision free elementary path between two configurations. It aims

Sébastien Dalibard and Jean-Paul Laumond
LAAS-CNRS, University of Toulouse, 7 avenue du Colonel Roche,
31077 Toulouse Cedex 4, France
e-mail: {sdalibar,jpl}@laas.fr

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 467–481.
springerlink.com

at capturing the topology of the free space in a learning phase in order to handle multiple queries in a solving phase.
- The *diffusion* approach, introduced in both [5] and [9], which includes RRT planners, consists in solving single queries by growing a tree rooted at the start configuration towards the goal configuration to be reached.

These methods have been proven to be efficient and suitable for a large class of motion planning problems. Work has been done to analyze and validate theoretically these algorithms. Probabilistic completeness has been studied and proved for RRT [9], as well as for PRM [7].

1.2 Narrow Passages

In some environments, however, passing through so-called *narrow passages* is a difficult task for probabilistic motion planners. A lot of work has been done to evaluate this difficulty, as well as to overcome it.

The formalism of expensive spaces was first presented in [5]. It quantifies the complexity of a configuration space from a randomized planning point of view. This formalism helps understanding what makes a motion planning problem difficult. Narrow passages and the configuration space dimension are identified as the main sources of complexity.

It is believed that probabilistic planning algorithms have an exponential complexity with respect to the dimension of the configuration space, as the number of nodes needed to describe the free space has a combinatorial explosion. The presence of narrow passages makes the search more difficult because it is unlikely to sample configurations that lie in the passage, while passing through it could be necessary to solve the planning problem. The quality of the work of Hsu and Latombe comes from the precise quantification of these natural ideas. However, it does not provide a way of estimating the complexity of a given configuration space, which makes these results non effective, when one would wish to use them to accelerate probabilistic planning. Some work deals with controlling probabilistic planning algorithms.Visibility based PRM is presented in [17]. A visibility criteria is used to improve the quality of the sampling and accelerate the algorithm. Dynamic-domain RRTs, presented in [20], control the sampling domain of a RRT algorithm to overcome some bias of classical RRT algorithm.

More recently, work has been done to use workspace information within *CS* probabilistic algorithms. In [15] for instance, diffusion uses workspace hints, such as real world obstacle directions, or decouples rotations and translations to accelerate a RRT algorithm that diffuses through a narrow passage.

1.3 Motivations

Motivations for the work presented here come from motion planning for digital actors. In these high-dimensioned problems - we consider whole-body motion planning, diffusion algorithms behave rather well, but can be slowed down by narrow

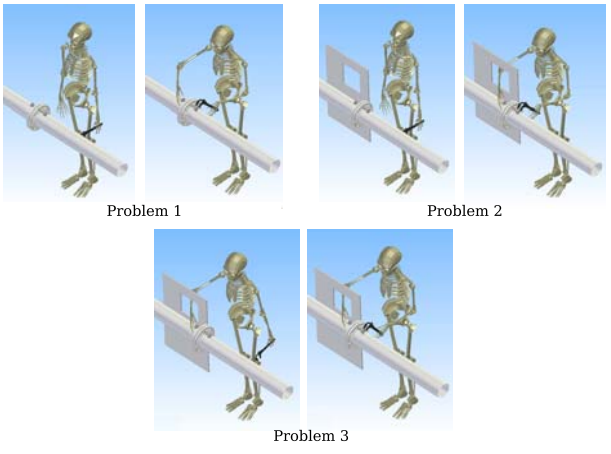


Fig. 1. Three animation motion planning problems. The mannequin has to get into position to fix the pipe. The first problem is unconstrained, we then add an obstacle for problems 2 and 3. In problem 3 the right arm is already in position and the mannequin has to find a way to move its left arm.

passages. The existence of a narrow passage, however, does not reflect the intrinsic difficulty of a motion planning problem.

Fig. 1 shows the example of a mannequin getting into position to fix a pipe. In the first environment, the mannequin has to move both arms in an unconstrained environment. This problem should be easy to solve for current motion planners. In the second environment, the right hand has to go through a hole to get into position. This makes the problem more difficult, since the path to find in *CS* goes through a narrow passage. The third problem takes place in the same environment, but the right hand is already in position. The difficulty of this last problem, from a human point of view, is more or less the same as the first one: the mannequin has to find a way to move only one of its arms in a free environment. However, the fact that the right hand is in position forces the diffusion to take place in a small volume around a sub-manifold of *CS* - the sub-manifold corresponding to a still right hand. This slows down a RRT algorithm (we will quantify it in section 3) and motivates our work on how to identify on line when a diffusion process takes place within small volumes of *CS*. Note that we want this identification to be automatic, to keep the generic properties of *CS* motion planners. This means that in the third problem of fig. 1 we do not want an outside operator to input the information about which degrees of freedom the mannequin should use.

1.4 Contribution

This paper presents a study on how narrow passages slow down diffusion algorithms, and a new diffusion planner based on this study. Our adaptation of diffusion algorithms analyzes the growing tree on line to estimate if it lies in a small volume

around a submanifold of *CS*. If it does, the search is restrained to that volume. That way, if the tree has to pass through a narrow passage, the algorithm does not loose time in trying to expend in the directions orthogonal to the passage. Thanks to the method used to describe the free space from the growing tree, the identified submanifold can be of dimension more than one, while rest of the contributions dealing with narrow passages consider tubes, i.e. small volumes around a one-dimensional manifold. This problem is different from the one of sampling for closed kinematic chains -presented for instance in [2], since in our case, the subspace in which the search should take place is unknown. The new algorithm presented in this paper only uses *CS*-information, which makes it more generic than algorithms that use workspace information.

Our work uses a long known and classical statistical method: the principal component analysis (PCA). It is used in many fields where subspace identification is needed, and in particular in [19], in a motion planning context. The use of PCA there is different than ours though, since it is only applied to motion description, while we use it to control and generate motion. We found some recent mathematical publications [21] that helped us understanding some convergence properties of PCA. They are shown and explained in this paper. Our algorithm has been tried in various examples, and the experimental results are presented. We chose to use it on different motion planning classes of problems: two motion planing benchmarks, an industrial disassembly problem and the animation problems of fig. 1.

2 Preliminaries: Rapidly Exploring Random Trees (RRT)

The method presented in this article can be applied to any diffusion algorithm. To understand what our method changes in a typical diffusion algorithm, let us remind briefly the structure of the RRT algorithm [12],[9]. It takes as input an initial configuration q_0 and grows a tree $\mathscr{T}$ in *CS*. At each step of diffusion, it samples a random configuration q_{rand}, finds the nearest configuration to q_{rand} in $\mathscr{T}$: q_{near}, and extends $\mathscr{T}$ from q_{near} in the direction of q_{rand} as far as possible. The new configuration q_{new} is then added to $\mathscr{T}$.

The version of RRT we consider is RRT-Connect rather than classical RRT. It often works better and is more generic, as there is no need to tune any step parameter.

3 Analyzing Probabilistic Diffusion in Narrow Passages

3.1 Local Description of a Narrow Passage

Most motion planners encounter difficulties when solving problems containing *narrow passages*. In [5], Hsu *et. al* defined the notion of expansive spaces. This formalism describes configuration spaces containing difficult areas, or narrow passages. These difficulties occur when connected areas of CS_{free} do not *see* large regions outside themselves. Probabilistic planners then have to sample configurations in these small volumes to solve the problem, which make the search longer. The expansive

space formalism describes the global difficulty of a configuration space. What we studied here is the local properties of CS_{free} that slow down a diffusing algorithm. In other terms, how can we describe locally a narrow passage?

Configuration space probabilistic planners try to find a path between a start and a goal configurations with as few samples as possible. Basically, this means that one would want new edges to be as long as possible - the accurate notion is that one would want to add nodes that expand the roadmap visibility region as much as possible.

Let q be a configuration in CS_{free}. To describe how a diffusing tree will grow from q, one should look at its visibility region $V(q)$. If $V(q)$ is equally constrained in all directions, then diffusing algorithms will behave nicely. A RRT-Connect planner will add edges from q of length the radius of $V(q)$, which is the best one could expect. On the other hand, if $V(q)$ has an elongated shape, we can say that q lies in a narrow passage. The volume of $V(q)$ is small because of constraints in some directions. The tree will grow slowly, while it could expand easily in some other directions.

Thus, a good description of the local difficulty to expand from q is the variances and covariances of $V(q)$ along a basis of CS. Directions accounting for most of the volume of $V(q)$ are the ones along which the tree could be expanded the furthest, while directions accounting for low volume are highly constrained and slow down a diffusing algorithm.

Some remarks should be made at this point. We use two key notions in motion planning: visibility and $CS-$metrics. The visibility region depends on the local method the planner uses to connect two configurations. That local method is part of the motion planning problem.

The metric used to compute distances in CS is a critical point in diffusion algorithms, since it should capture the likeliness of configurations to be connected. As we describe narrow passages in terms of distance, variance and covariance, our description, as well as the algorithm we will present, are also very dependant on the metric. However, we will not discuss the choice of the metric, which is a deep motion planning algorithmic issue, and just assume CS has a metric well suited for the problem.

3.2 One Step of Diffusion in a Narrow Passage

What characterizes a narrow passage is the elongated shape of a visibility region, i.e. the region has a much higher variance along some directions than along others. To understand how it slows down the expansion of a random tree, it is sufficient to look at a 2D example, where one dimension is more constrained than the other.

Fig. 2 shows a 2D example of a narrow space. Here, the visibility region of q has a rectangular shape. One of the dimension of the rectangle is fixed, and the other approaches 0, which means the space around q gets narrower. The configuration towards which the tree is extended is sampled at random on a circle centered in q of

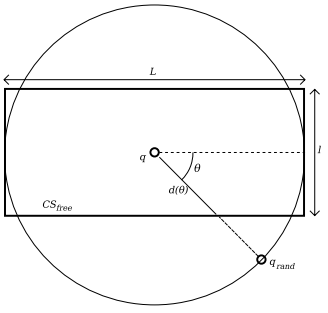


Fig. 2. One step of RRT expansion in a 2D narrow passage. L is fixed and l approaches 0. We are interested in the expected length $\overline{d}$ of the new vertex of the tree. For $|\theta[\pi]| \leq \arcsin(l/L), d(\theta) = L/2$, otherwise $d(\theta) = l/2\sin(\theta)$

radius half the length of the rectangle. Thus, the diffusion from q is isotropic, and the longest progress the tree could make starting from q is $L/2$.

In this example, we can compute the mean length $\overline{d}$ of the new edge returned by one step of diffusion. The exact value of $d(\theta)$ is indicated in 2.

$$\begin{array}{rcl} \overline{d} & = & \frac{1}{2\pi}\int_{\theta=0}^{2\pi} d(\theta)d\theta \\ & = & \frac{1}{\pi}\left(L\arcsin(l/L) - l\log(\tan(\arcsin(l/L)/2))\right) \\ & \simeq_{l\to 0} & -\frac{l}{\pi}\log(l) \end{array}$$

As the width of the rectangle approaches 0, the mean distance the tree is extended converges to 0, even though q sees configurations far away in one of the two dimensions. This example shows how diffusion is slowed down in a narrow passage: if some directions are more constrained than others, the diffusion process does not go as fast as possible in the free directions. We used a 2D example to describe the effect of one dimension getting narrower, but the result is still valid in higher dimension. If more dimensions are constrained the diffusion gets even slower.

4 A New Planner Using Dimensional Reduction Tools

According to the previous description of narrow passages, a good way to quantify the difficulty of a region of CS_{free} in terms of probabilistic motion planning is to describe the shape of its points visibility region. Following the sampling motion planning paradigm, we do not want to describe effectively CS_{free}. The only information that we have and will use are the vertices of the random tree already obtained.

Given a node we are about to extend the tree from, we will assume its nearest neighbours in the tree capture the local shape of the free space. We will then describe that shape by computing its variances and covariances along the canonical basis of CS. A statistical tool, the Principal Component Analysis, is then used to determine directions in CS accounting for most of the variance of this region of CS_{free}. Given

that description, we change the random direction in which the tree is about to be extended to follow the directions of the passage the node lies in.

4.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical dimensionality reduction technique. Its goal is to determine the underlying dimensionality of a set of p points in an $n-$dimensional space. PCA is one of the most commonly used technique for dimensionality reduction. It was first proposed by Pearson [14] and further developed by Hotelling[4]. A good description can be found in Jolliffe [6].

This method involves a mathematical procedure that, given p points, computes an orthogonal change of basis such that the first vectors of the new basis are the directions that account for most of the variance of the input set. The underlying structure found by PCA is therefore linear.

The new basis and the corresponding variances are obtained by diagonalizing the covariance matrix of the input points. They correspond respectively to its eigenvectors and eigenvalues. The PCA Algorithm takes a set of p points in an $n-$dimensional space, computes its covariance matrix C and returns the matrices Λ and U, containing respectively the eigenvalues and the eigenvectors of C.

4.2 Applying PCA to Diffusing Algorithms

The way we use PCA for diffusion algorithms does not change their structure, we only reimplemented the extension step. At each step of diffusion, we find the p nearest neighbours of q_{near} by performing a Breadth-First Search in the tree. We then compute a PCA on those points. The random direction towards which the tree should be extended (q_{rand}) is changed according to the results of the PCA, in order to favour the directions in which the variance of the growing tree is high.

More precisely, let us note $\lambda_1 > \cdots > \lambda_n > 0$ the eigenvalues of the covariance matrix, and $\mathbf{u_1}, \ldots, \mathbf{u_n}$ the corresponding eigenvectors. Placing the origin on q_{near}, the coordinates of the new random direction q'_{rand} , in the eigen basis, are given by:

$$\forall i \in [1,n], q'^{(i)}_{rand} = \frac{\lambda_i}{\lambda_1} q^{(i)}_{rand} \tag{1}$$

Thus, U being the eigen change of basis matrix, we get in CS canonical basis :

$$q'_{rand} = q_{near} + \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1} (q_{rand} - q_{near}).U_i \right) U_i \tag{2}$$

By projecting directions with low variances, the algorithm makes the diffusion process go faster in narrow spaces. Recalling the example of fig. 2, we can show the difference between classical diffusion and PCA-controlled diffusion. Fig.3 shows how an isotropic distribution around q is transformed into an ellipsoidal distribution. We indicated in red where half of the probability stands, before and after

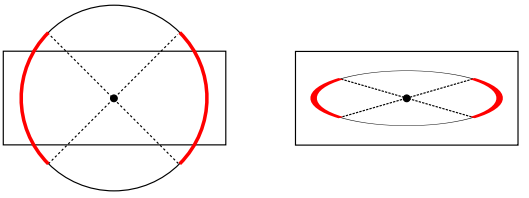


Fig. 3. One step of diffusion in a narrow passage. One the left, RRT isotropic diffusion and on the right PCA-RRT diffusion, concentrated along the direction of the passage. The bold red part shows where half of the measure is concentrated.

transforming q_{rand}. One can see that with PCA-Extend, the measure is concentrated on the ends of the ellipse. That way, when the space gets narrower, the expected length of one step of diffusion does not converge to 0, but to a positive limit.

4.3 Accuracy of the PCA Description

The previous sections explained how knowledge about the local shape of CS_{free} could help improving the performance of a diffusion algorithm. However, a key point is the accuracy of this knowledge. We said that we assume the nearest neighbours of a configuration catch the shape of the free space. Two questions remain: how true is this assumption and how many neighbours are necessary to describe locally CS_{free}?

The following result answer these questions. It is a simplified version of mathematical results found in [21]. The accuracy of PCA description is expressed as the distance δ between the high variance subspace found by PCA and the real high variance subspace CS_{free} lies around. Let D be the dimension of that subspace and $\lambda_1 > \cdots > \lambda_n > 0$ the eigen values of the local covariance matrix of CS_{free}.

Theorem 4.1. *For p points lying in a ball of radius r, for $\xi \in [0,1]$, the following bound holds with probability at least $1 - e^{-\xi}$:*

$$\delta \leq \frac{4r^2}{\sqrt{p}(\lambda_D - \lambda_{D+1})}\left(1 + \sqrt{\frac{\xi}{2}}\right)$$

This theorem is a direct consequence of Theorem 4 in [21]. It illustrates two points: the speed of convergence of the PCA analysis with respect to the number of points is $1/\sqrt{p}$, and the narrower the free space is - the higher $(\lambda_D - \lambda_{D+1})$ is, the more accurate the PCA description will be.

Note that if the space is unconstrained (all the λ_i are close to each other), this bound is not interesting. In that case, PCA may return imprecise results, but then again, traditional diffusion algorithms behave properly. We will discuss in a next section how not to suffer from PCA computation in a locally unconstrained space.

5 Implementation Details

5.1 PCA Bias in Unconstrained Spaces

As shown earlier, the computation of PCA will give accurate results in narrow passages. However, if the free space is locally unconstrained - or equally constrained in all directions - the description returned by PCA may not be accurate enough. This could induce a bias in the algorithm: if the space is free, the algorithm may favour expansion in directions that have previously been sampled at random. This is a main drawback since the new diffusion algorithm loses one of the key properties of the RRT algorithm: the convergence of the RRT vertices distribution towards the sampling distribution.

To overcome this bias, our implementation of PCA-RRT uses, at each step of expansion, PCA-Extend with probability 0.5 and classical RRT Extend with probability 0.5. That way, the tree still covers densely the free space, while the PCA-Extend function accelerates the search in narrow passages.

5.2 Recursive PCA

Previous section presented ideas about how many points are necessary to correctly describe the space. It depends on each region of the space, narrow passages needing less points than free regions. As we do not want that tuning to be done by a user, but *automatically* by the algorithm, we need a version of the algorithm where the number p of nearest neighbours used to compute the PCA is no longer a parameter. It is done by evaluating the PCA result each time we add a new neighbour. When the dimension of the subspace identified by PCA converges, the number of neighbours is sufficient and we can use the results of the PCA. This idea comes from a similar method used in [11] to determine when to stop a visibility PRM algorithm: when the size of the region visible from the milestones stops growing, we assume that the set of milestones correctly describes the connectivity of the space.

This needs an incremental version of the PCA, in order not to recompute the diagonalization of the covariance matrix at each step. A good description of incremental PCA can be found in [3]. It uses matrix perturbation theory.

6 Experimental Results

In this section, we compare our PCA-RRT algorithm with classical RRT. The algorithm has been implemented in KineoWorks^{TM}, we used its implementation of RRT as a reference. The default expansion method used in KineoWorks is RRT-Connect. Since our expansion method only uses *CS* information, it can be usefull in any motion planing problem, as long as the configuration space contains narrow passages that slow down classical diffusion methods. To show the range of uses of our method, we present results in various motion planning problems: first two

motion planning benchmarks, then an industrial disassembly problem, and finally three computer animation problems, where a virtual character has to use a wrench.

All the experiments presented here were performed on a 2.13 GHz Inter Core 2 Duo PC with 2 GB RAM. Each method was tested ten times on each environment and we show the average results of all test runs. The results report the time taken by each method, the number of expansion steps and the number of nodes needed. We have also indicated the average length of each expansion step, in an arbitrary unit, to show the benefit of PCA-RRT in narrow passages.

6.1 Motion Planning Benchmarks

The two following environments are known motion planning benchmarks that contain narrow passages. The random tree the algorithm will grow is rooted inside the narrow passage, so the difficulty is to follow the passage rather than find the entrance of it. Results of OBRRT on these environments were published in [15]. We will recall these results, even if direct comparisons are difficult. Computation times are irrelevant since the computers on which the experiments have been made are different. We will show the number of iterations taken by each algorithm, including both implementations of RRT, since they seem to behave differently.

6.1.1 Flange Environment

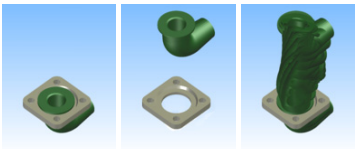
In this environment, the robot is a free flyer tube, that should slide out of the obstacle. We present the results of our algorithm on two versions of the environment, the original one and a easier one rescaled at 0.95.

6.1.2 S-Tunnel Environment

This environment consists in an s-shaped tunnel, through which the robot should pass. The initial configuration, where the random tree is rooted, is placed in the middle of the passage. We tested this environment with the same three robots as in [15].

In these two benchmarks, PCA-RRT give better results than classical RRT. However, we gain less with PCA-RRT than with OBRRT. One should notice that our method *does not need any tuning*, while the results of OBRRT we indicated correspond to a tuned version, adapted to each problem. Besides, the structure of our algorithm being the same as the OBRRT one, they can be used at the same time, to gain from both workspace hints from OBRRT and *CS* information from PCA-RRT.

An other important point is that PCA-RRT only uses *CS* information which makes it usable in the following problems, where workspace information is not relevant. Indeed, in the animations problems presented in fig. 1, decoupling rotations and translations, or following the obstacle directions does not help since there are only rotation degrees of freedom.



Flange 0.95				
	Iterations	Nodes	Time	Distance
RRT	6062	89.1	16.4 s	0.095
PCA-RRT	5008	97.7	15.3 s	0.11
RRT (Rodriguez *et al.*)	13260	109.2	328.8 s [1]	-
OBRRT (tuned)	3220	479.8	143.2 s [1]	-
Flange 1.0				
	Iterations	Nodes	Time	Distance
RRT	222786	117.4	343.1 s	0.075
PCA-RRT	147331	357	304.9 s	0.099
RRT (Rodriguez *et al.*)	-	-	-	-
OBRRT (tuned)	6100	1011	227.1 s [1]	-

Fig. 4. Flange environment. Start and goal configurations, solution path, amd experimental results.

6.2 Industrial Disassembly Problem

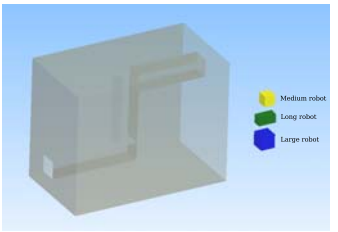
This environment is the back of a car. The robot, the exhaust, must be disassembled. The initial configuration is the mounted position, and the random tree must get out of the narrow passage to the unmounted position.

6.3 Animation Problems

This algorithm was first designed to overcome the limitations of traditional motion planing algorithms. Among others, highly dimensioned problems are hard to solve for diffusion algorithms, as it is believed that these algorithms have an exponential time complexity with respect to the number of degrees of freedom of the robot. The problems presented in fig. 1 deal with a rather highly dimensioned robot: the upper body of a virtual character. The robot can move its torso, head and both arms. These adds up to 20 degrees of freedom. It should get into position to fix a pipe. Its left hand holds a wrench and its right hand should hold a bolt that lies into a hole. The difficulties in that environment come from the narrow space the right hand has to go through, the dimension of *CS* and the collisions caused by the arms and wrench.

Problem 1 is an easy problem for current motion planners. There is no real need for sophisticated techniques such as PCA-RRT here.

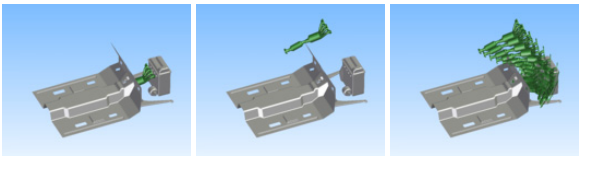
[1] Those figures come from [15]. Notice that while number of iterations and nodes are relevant for a comparison between various methods, time comparison is irrelevant because depending on the machine.



Medium Robot				
	Iterations	Nodes	Time	Distance
RRT	3380	227.8	2.1 s	0.27
PCA-RRT	1638	216.6	1.4 s	0.39
RRT (Rodriguez *et al.*)	3780	541.4	15.4 s [1]	-
OBRRT (tuned)	197	190.9	0.8 s [1]	-
Long Robot				
	Iterations	Nodes	Time	Distance
RRT	4769	167.4	2.6 s	0.19
PCA-RRT	2313	193.9	1.7 s	0.23
RRT (Rodriguez *et al.*)	11322	1311.7	212.6 s [1]	-
OBRRT (tuned)	699.3	589.1	2.8 s [1]	-
Large Robot				
	Iterations	Nodes	Time	Distance
RRT	16389	285	10.6 s	0.21
PCA-RRT	3320	234.8	2.5 s	0.36
RRT (Rodriguez *et al.*)	40513	2740.6	699.6 s [1]	-
OBRRT (tuned)	331	117.6	1.0 s [1]	-

Fig. 5. S-Tunnel environment with three different robots. The robots start in the middle of the tunnel and have to get out.

In problem 2, we used a bi-RRT -i.e. one tree is rooted at the initial configuration and an other one at the goal configuration, they are both randomly grown- since the feature of the algorithm we want to test is the diffusion in a narrow space rather than finding the entrance of narrow spaces.



	Iterations	Nodes	Time	Distance
RRT	38101	7693.8	1 063.0 s	33
PCA-RRT	11672	3592.6	452.8 s	53

Fig. 6. Exhaust disassembly problem. Start and goal configurations, solution path and experimental results.

	Iterations	Nodes	Time	Distance
RRT	3260	215.2	3.2 s	0.022
PCA-RRT	978	76.2	1.8 s	0.095

Fig. 7. Solution path and results for problem 1.

	Iterations	Nodes	Time	Distance
RRT	168171	657.5	526.7 s	0.21
PCA-RRT	5470	806.5	39.1 s	0.72

Fig. 8. Solution path and results for problem 2.

	Iterations	Nodes	Time	Distance
RRT	38990	212.6	109.5 s	0.017
PCA-RRT	4643	147.6	15.8 s	0.026

Fig. 9. Solution path for problem 3.

In problem 3, the right arm is already in position. The random tree is rooted inside the narrow passage - at the goal configuration.

Those last two animation problems show the benefit of using PCA-RRT. As foreseen in introduction, the difficulty of problem 3 is not much higher than the one of problem 1, thanks to PCA-RRT. The time taken by our algorithm is one order of magnitude below the time taken by classical RRT. This comes from the speed of diffusion within narrow passages as one can see by looking at the mean distance of step extension.

7 Conclusion

An adaptation of randomized diffusion motion planning algorithms is proposed. The statistical method we used, PCA, is well-known and widely used in literature, but not yet to control motion planning algorithms. Using dimensional reduction tool in diffusion planning, and analyzing the result of it is the contribution of this work.

Of course it could be worth trying other dimensionality reduction technique, for instance non-linear ones ([18],[16]). As expected, we observed that PCA could significantly improve diffusion algorithm performances in constrained environments. Note that it does not solve the problem of finding the entrance of a narrow passage, but only accelerate the diffusion within the passage. Our method has been used on several motion planning problems, coming from different domains of application, and has shown good results.

Acknowledgements. This work is partly supported by the French ANR-RNTL project PerfRV2.

References

1. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of Robot Motion: theory, algorithms, and implementation. MIT Press, Cambridge (2005)
2. Cortes, J., Simeon, T., Laumond, J.: A random loop generator for planning the motions of closed kinematic chains using PRM methods. In: IEEE International Conference on Robotics and Automation, ICRA 2002. Proceedings, vol. 2 (2002)
3. Erdogmus, D., Rao, Y.N., Peddaneni, H., Hegde, A., Principe, J.C.: Recursive principal components analysis using eigenvector matrix perturbation. EURASIP Journal on Applied Signal Processing 2004(13), 2034–2041 (2004)
4. Hotelling, H.: Analysis of a Complex of Statistical Variables into principal components. J. Ed Psychology 24, 417–441 (1933)
5. Hsu, D., Latombe, J., Motwani, R.: Path planning in expansive configuration spaces. Int. J. Computational Geometry & Applications 9(4-5), 495–512 (1999)
6. Jolliffe, I.T.: Principal Component Analysis. Springer, Heidelberg (2002)
7. Kavraki, L., Kolountzakis, M., Latombe, J.: Analysis of probabilistic roadmaps for path planning. IEEE Transactions on Robotics and Automation 14(1), 166–171 (1998)
8. Kavraki, L., Svestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
9. Kuffner, J., LaValle, S.: RRT-connect: An efficient approach to single-query path planning. In: Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA 2000), San Francisco, CA (April 2000)
10. Latombe, J.: Robot Motion Planning. Kluwer Academic Publishers, Dordrecht (1991)
11. Laumond, J., Simeon, T.: Notes on Visibility Roadmaps and Path Planning. In: Algorithmic and Computational Robotics: New Directions: the Fourth Workshop on the Algorithmic Foundations of Robotics (2001)
12. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University (October 1998)
13. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006), http://planning.cs.uiuc.edu/
14. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2(6), 559–572 (1901)
15. Rodriguez, S., Tang, X., Lien, J., Amato, N.: An obstacle-based rapidly-exploring random tree. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, pp. 895–900 (2006)

16. Roweis, S.T., Saul, L.K.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290(5500), 2323–2326 (2000)
17. Siméon, T., Laumond, J., Nissoux, C.: Visibility-based probabilistic roadmaps for motion planning. Advanced Robotics 14(6), 477–493 (2000)
18. Tenenbaum, J.B., Silva, V.d., Langford, J.C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290(5500), 2319–2323 (2000)
19. Teodoro, M.L., George, J., Phillips, N., Kavraki, L.E.: A dimensionality reduction approach to modeling protein flexibility. In: RECOMB 2002: Proceedings of the sixth annual international conference on Computational biology, pp. 299–308. ACM Press, New York (2002)
20. Yershova, A., Jaillet, L., Simeon, T., LaValle, S.: Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 3856–3861 (2005)
21. Zwald, L., Blanchard, G.: On the convergence of eigenspaces in kernel principal component analysis. In: Neural Information Processing Systems (2005)

Stochastic Motion Planning and Applications to Traffic

Sejoon Lim, Hari Balakrishnan, David Gifford, Samuel Madden, and Daniela Rus

Abstract. This paper presents a stochastic motion planning algorithm and its application to traffic navigation. The algorithm copes with the uncertainty of road traffic conditions by stochastic modeling of travel delay on road networks. The algorithm determines paths between two points that optimize a cost function of the delay probability distribution. It can be used to find paths that maximize the probability of reaching a destination within a particular travel deadline. For such problems, standard shortest-path algorithms don't work because the optimal substructure property doesn't hold. We evaluate our algorithm using both simulations and real-world drives, using delay data gathered from a set of taxis equipped with GPS sensors and a wireless network. Our algorithm can be integrated into on-board navigation systems as well as route-finding Web sites, providing drivers with good paths that meet their desired goals.

1 Introduction

This paper presents and evaluates an algorithm for planning the motion of vehicles (autonomous or human-driven) on roadways in the face of traffic delays. Rather than model road delays statically, as in current on-board navigation systems and Web-based mapping services, our algorithm uses past observations of actual delays on road segments to model these delays as probability distributions. The algorithm minimizes a user-specified cost function of the delay distribution. We investigate a few cost functions in detail, particularly one that is equivalent to maximizing the likelihood of reaching a destination within a specified travel deadline.

Our work provides a planning system that can be used by robots as well as human drivers. The system is a useful addition to on-board navigation systems using computer-aided automation to provide good paths that meet desired travel goals

Sejoon Lim, Hari Balakrishnan, David Gifford, Samuel Madden, and Daniela Rus
CSAIL, MIT
e-mail: {sjlim,hari,gifford,madden,rus}@csail.mit.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 483–500.
springerlink.com

(e.g., "when should you leave, and what path should you take, to reach the airport by 8am with high probability?"); it is also a worthwhile addition to Web-based mapping services. We view the incorporation of traffic-aware path computation as an important practical addition in the rapid trend toward computer-assisted driving and autonomous decision-making in vehicles.

Traffic congestion is clearly a serious problem: a recent survey [11] estimates that the annual nationwide cost of traffic congestion is $78 billion, including 4.2 billion hours in lost time and 2.9 billion gallons in wasted fuel. Drivers today have little knowledge of historic and real-time traffic congestion on the paths they drive, and even when they do (e.g., from "live" traffic updates), they generally don't know how to use that information to find good paths. As a result, they often tend to drive suboptimal routes and often leave well in advance when they need to make an important deadline.

In addition to helping individual cars avoid congested roads, we believe that our work, if deployed widely, can manage traffic flow, reduce congestion, and reduce the fuel consumed by cars on a macroscopic basis by using the under-utilized parts of the road network better than today (thereby reducing load on congested areas). Using our algorithm to investigate this global traffic management question is an area for future work. In this paper, we are concerned with finding good paths for a single car.

The main challenge in planning paths taking traffic delay into account is that these delays are not fixed. The delay on a road segment is best modeled as a probability distribution; in addition, this distribution typically depends on a number of factors, such as time-of-day, whether it's a working day or not, events such as concerts or sporting events, weather, etc. The shortest-distance path is often not the best path to use if one seeks to minimize the expected travel time or maximize the probability of reaching the destination by a certain time. Our algorithm uses historic observations of travel delays on road segments at different times of day to produce delay distributions (indexed by time-of-day). We posit that this information, together with real-time updates of extraneous conditions (such as accidents), is invaluable (and sufficient) to compute good paths that meet user-specified goals. Given the probability distributions of delays on segments, finding good paths requires more than a shortest-path computation, because the "optimal substructure" property does not hold as explained in [8] (i.e., if the best path from S to T goes through X, it does not follow that the sub-path of this path from S to X is itself the best S-X path).

We have implemented our algorithm and evaluated it by first modeling the historic delays using data from the CarTel vehicular testbed [5], a network of 28 taxis. The data consists of travel times organized by road segment and by time-of-day, yielding statistical profiles for all the road segments. We model the road network as a weighted graph where the nodes represent intersections and the edges represent road segments. An aggregation algorithm combines the road segments into groups to coalesce the important delay characteristics without losing information about alternate paths. Our algorithm has the flavor of searching and pruning the delay statistics

on the road network data structure. We evaluate the algorithm and its assumptions using simulation and actual test driving.

1.1 Related Work

Our work is closely related to stochastic planning and stochastic shortest-path algorithms. In [3], an efficient algorithm for a dynamic shortest path with time dependent deterministic edge cost is given. Prior work has considered stochastic shortest paths under the assumption that the travel time follows a known probability distribution [12, 1, 6, 13, 7, 8, 9]. In stochastic shortest path, edge costs are probability distributions rather than deterministic costs and the optimal path depends on drivers' diverse objectives. When a driver's objective is to minimize the expected travel time, the problem becomes the deterministic shortest-path problem. It is well known that Dijkstra's algorithm is optimal and applicable for deterministic problems. However, for some goals such as maximizing the probability of arriving within a given deadline, the optimal path cannot be found with the standard shortest-path algorithms since the optimal substructure property does not hold. Nikolova et al. [9] developed an algorithm and theoretical bounds by assuming that delays are both Gaussian and independent on different road segments. Inspired by this algorithm, we developed a method that improves performance by removing unnecessary invocations of shortest-path searches.

There have been several approaches to acquiring traffic data. The most prevalent one uses inductor loops installed beneath roads [14, 4]. This is adequate for counting the number of cars that pass a specific location, but it is not suitable for measuring travel time and measurements are possible only on instrumented roads. Recently, GPS sensors installed in probe vehicles have been used [10, 5, 15]. The travel time of vehicles can be measured and recorded for each route segment. In [5], the researchers developed a system called CarTel Network nodes that include GPS and wireless communication. This system was used to study routing and data delivery from cars.

1.2 Outline

This paper is organized as follows. Section 2 gives an overview of a route planning system for traffic. Section 3 gives the stochastic motion planning formulation, and Section 4 presents our algorithm and gives its correctness and performance bounds. Section 5 evaluates the algorithm in simulation and also using physical data from the deployment of GPS sensor nodes in taxis.

2 Transportation System Context

Our research objective is to provide an effective navigation system for autonomous or human-piloted cars that uses historical and real-time traffic data to determine

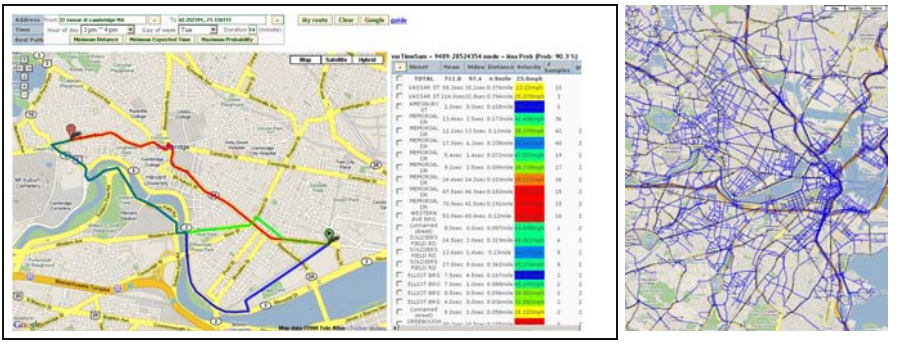


Fig. 1. The user interface to the traffic system with highlighted paths and travel times found by the algorithm described in this paper. Roads with traffic travel time data in our database are also shown in the right.

optimal driving directions and traffic estimates. Our intelligent navigation system consists of:

1. a data gathering system included in cars that move in traffic frequently;
2. a data analysis system to compile a historical database of traffic conditions;
3. an algorithm for route planning that uses both historical data and current information;
4. a traffic information system implementation with an appropriate interface.

In [5] a system called CarTel was developed that uses GPS and wireless communication to collect position and time data from cars. CarTel nodes deployed in 28 taxis since January 2007 collected travel data. This data was organized per road segment to create a historical database of traffic delays.

A stochastic route planning algorithm was developed and implemented. The best route depends on the drivers' goals and is a combination of speed and reliability. A Web-based interface (Fig. 1) allows users to query the system for traffic conditions and for optimal paths given historical data. The rest of this paper describes the motion planning component 3. of this intelligent traffic system.

3 Problem Formulation

3.1 Road Network Modeling

The road network is a graph (called the Geographic Map), where nodes represent intersections and edges represent road segments. We associate a road delay distribution with each road segment (Fig. 1). This per-intersection granularity road map leads to a large graph for small road segments with related travel statistics. We combine statistically related road segments into groups so that they can capture important delay characteristics without losing information about alternate path segments. This data structure is the Delay Statistics Map. The Geographic Map is used for

matching GPS traces onto real road segments, while the Delay Statistics Map is used for statistical delay sensitive routing. We assume that

1. the delay of each edge follows a Gaussian distribution;
2. the delay of each edge is independent of every other edge.

In Section 5, we provide evidence for these assumptions. We formulate stochastic motion planning as a graph search problem over a graph with an origin O and a destination D, where the travel time of each edge is an independent Gaussian random variable. Since the sum of independent Gaussian random variables is also a Gaussian random variable, we can denote the travel time for a path π consisting of edges e of mean m_e and variance v_e as follows:

$$t_\pi \sim \mathcal{N}(m_\pi, v_\pi), \text{ where } m_\pi = \sum_{e\in\pi} m_e \text{ and } v_\pi = \sum_{e\in\pi} v_e.$$

3.2 Cost Functions

Our objective is to find a path that minimizes an expected cost when the cost function models a user's goal. We call this the "optimal" path for the given cost function. We consider several cost functions including:

Linear cost: Here, the cost increases linearly with the travel time. When the cost of arriving at the destination in time t is $C(t) = t$ and the delay PDF of a path π is $f_\pi(t)$, the expected cost of traveling through π is $EC_\pi = \int_{-\infty}^{\infty} t f_\pi(t)dt = m_\pi$. Linear cost models the path with minimum expected time.

Exponential cost: Exponential cost models a cost function that increases sharply as the arrival time increases. When the cost function is $C(t) = e^{kt}$, where k is the steepness of the cost increase, the expected cost can be written as $EC_\pi = \int_{-\infty}^{\infty} e^{kt} f_\pi(t)dt = \{e^{k(m_\pi + \frac{kv_\pi}{2})}\}$. This exponential cost function minimizes a linear combination of mean and variance determined by k.

Step cost: Step cost models a cost that only penalizes the late arrival after a given deadline. The cost function is $C(t,d) = u(t-d)$, where $u(\cdot)$ is the unit step function and d is the deadline. The expected cost is $EC_\pi(d) = \int_{-\infty}^{\infty} u(t-d) f_\pi(t)dt = \int_d^{\infty} f_\pi(t)dt = \{1 - \Phi(\frac{d-m_\pi}{\sqrt{v_\pi}})\}$, where $\Phi(\cdot)$ is the CDF of the Standard Normal distribution. Thus, when Π is a set of all paths from O to D, the minimum expected cost path is $\text{argmax}_{\pi\in\Pi}\ \Phi(\frac{d-m_\pi}{\sqrt{v_\pi}})$, which turns out to be the path that maximizes arrival probability. Since $\Phi(\cdot)$ is monotonically increasing, maximizing $\Phi(\cdot)$ is equivalent to maximizing

$$\varphi_d(\pi) = \frac{d - m_\pi}{\sqrt{v_\pi}}. \tag{1}$$

The minimum expected cost path for the linear and exponential cost cases can be found by a deterministic shortest-path algorithm, such as Dijkstra's or A^* search

algorithm since the cost of a path can be expressed as the sum of the cost of each edge in the path. However, when the cost is a step function, these algorithms cannot be used since the objective, (1), is nonlinear. Our goal for the rest of the paper is to develop an efficient algorithm for finding the maximum probability path given a deadline.

4 Stochastic Path Planning by Parametric Optimization

In [9], an algorithm for the case of normally distributed edge costs was given based on quasi-convex maximization. It finds the path with the maximum arrival probability by standard shortest-path runs with different edge costs corresponding to varying parameters. We now give a graphical interpretation of the optimal path and show a connection to a parametric optimization problem, which will ultimately lead to a new algorithm that reduces unnecessary runs over [9].

4.1 Transforming the Cost Function into Parametric Form

Let path π be denoted by a point (m_π, v_π) in a rectangular coordinate system, called the $m-v$ plane, where the horizontal axis represents the mean and the vertical axis represents the variance. The objective of the optimization problem, (1), can be rewritten to show the relation between m_π and v_π as

$$v_\pi = \frac{1}{\varphi_d(\pi)^2}(m_\pi - d)^2, \tag{2}$$

which is a parabola in the $m-v$ plane with apex at d, where $\varphi_d(\pi)$ is determined by the curvature of the parabola. Thus, the optimal path is the path that lies on the parabola of the smallest curvature. Intuitively, the optimization problem is to find the first path that intersects the parabola while we lift up the parabola starting from the horizontal line (see Fig. 2 (Left)). This suggests finding the optimal path using linear optimization with various combinations of cost coefficients.[1]

Consider setting the cost of an edge to be linear combinations of mean and variance, $m_e + \lambda v_e$, for an arbitrary non-negative λ. We call the solution for this edge cost the λ-optimal solution. This edge cost follows the optimal substructure property and has the property described in Lemma 4.1, which was also stated in [9].

Lemma 4.1. *An optimal path occurs among the extreme points of the convex hull for all the O to D paths in the $m-v$ plane if there exists a path that has a mean travel time smaller than the deadline.*

Proof. Let point P on the $m-v$ plane represent the optimal path. Then, there is no path point that has the φ value larger than that of P. Therefore, every other

[1] Linear optimization finds a path that first intersects a straight line when the line is moved in a direction determined by cost parameters.

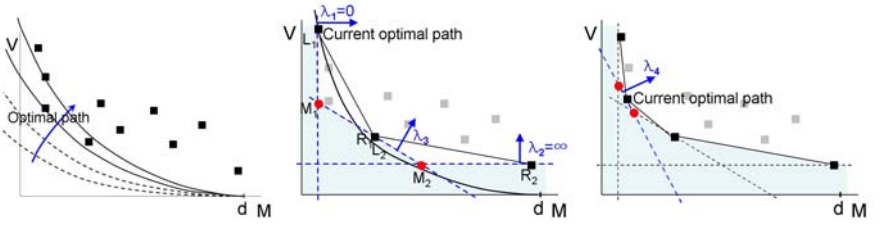


Fig. 2. (Left) Graphical interpretation of the optimal path in the $m-v$ plane. Each square represents a path from the origin to the destination. Equi-probability paths lie on a parabola with an apex at $(d,0)$ and a curvature of $\frac{1}{\varphi_d(\pi)^2}$. The optimal path is the first point that meets with a parabola as we increase the curvature. (Middle) The result after three executions of λ-optimal searches with $\lambda_1 = 0$, $\lambda_2 = \infty$, and $\lambda_3 = -\frac{m_0 - m_\infty}{v_0 - v_\infty}$. Each square point represents the λ-optimal path for each λ. The gray points represent the paths that are not found yet. The blue regions are guaranteed to contain no path. The white triangles indicate candidate regions for better paths. The round points are the probe points of the regions. (Right) λ-optimal search was done only for the left candidate region. The newly found path turns out to be the new current optimal path and the two round points are the probe points.

path point must be inside the parabola. Since the parabola is convex, P must be an extreme point.

With Lemma 4.1 we can find the optimal solution from λ-optimal solution for a given λ. Since λ-optimal cost satisfies the optimal substructure property, any deterministic shortest-path algorithm (e.g., Dijkstra's algorithm or A^* search algorithm) will find λ-optimal paths.

4.2 Exhaustive Enumeration

In [9], a method for stochastic motion planning was proposed that exhaustively enumerates all the extreme points of the path convex hull. A brief description of the algorithm is as follows: First, find the λ-optimal paths for $\lambda = 0$ and $\lambda = \infty$. If they are the same, it must be the optimal solution. Otherwise, find the λ-optimal path using $\lambda = -\frac{m_0 - m_\infty}{v_0 - v_\infty}$ since this λ value will cause the algorithm to search the entire region completely unless it finds a new path, as illustrated in Fig. 2 (Middle). If no new path is found, the algorithm terminates with the optimal solution being the one with the largest φ value. Otherwise, the newly found path divides the search region into two parts. Then, the λ-optimal search is executed for each region using λ values determined to search each region completely. In this approach, when the number of extreme points is N_e, there will be N_e searches to guarantee that all the extreme points are enumerated. In addition, $N_e - 1$ more searches are needed to conclude that no other paths exist between the extreme points. Thus, the total number of enumerations could be large. Next, we show how to reduce the number of required λ-optimal searches.

4.3 Examining Probe Points

If we know that a certain search region's best possible outcome is worse than the current optimal solution, we do not need to execute the costly λ-optimal search for that region. In this section we formalize this point.

Definition 4.1. *Let the triangular region where a better path can exist be called a* candidate region. *Let the vertex in the middle of the candidate region be called a* probe point. *Candidate regions are illustrated as white triangles $\triangle L_i M_i R_i$, and probes as round points M_i in Fig. 2 (Middle).*

Theorem 4.1. *If the φ value of the probe point as defined in (1) is smaller than the current optimal value, the candidate region does not contain the optimal path.*

Proof. Suppose that a path lies at the probe point. Then, no other point in the candidate region can be an extreme point. The interior points cannot be an optimal solution since the optimal solution occurs at one of the extreme points by Lemma 4.1. Suppose that a path does not lie at a probe point. Add an imaginary origin-to-destination path that lies on the probe point. The addition of an imaginary path will not make any difference for searching for the optimal solution since it is not better than the current optimal path. The same argument shows that interior points cannot be optimal solutions.

By Theorem 4.1, we can remove from consideration the candidate region if the region's probe point satisfies the condition in Theorem 4.1. Fig. 2 (Right) illustrates a case where the right candidate region $\triangle L_2 M_2 R_2$ was removed without any execution of λ-optimal search since the φ value of the probe point M_2 is smaller than that of the current optimal path. The left candidate region $\triangle L_1 M_1 R_1$ was searched since the left probe point gives a larger φ value than the current optimal value, and a new path was found as the λ-optimal path. The same procedure is applied to the new candidate regions built by the newly found λ-optimal path until there is no candidate region remaining.

4.4 Restricting λ by Upper and Lower Bounds

The λ values that should be searched are limited by upper and lower bounds.

Theorem 4.2. *The optimal path can be found by searching only with the λ values upper bounded by λ_u, the negative inverse of the tangent to the parabola at the intersection of the* 0*-optimal search line and the ∞-optimal search line.*

Proof. If the λ-optimal solution is the same as the λ_u-optimal solution for all $\lambda > \lambda_u$, we can trivially find the same path with λ_u instead of $\lambda > \lambda_u$. Suppose that there exists a certain $\lambda > \lambda_u$ for which λ-optimal path (m_λ, v_λ) is different from the λ_u-optimal path $(m_{\lambda_u}, v_{\lambda_u})$. Then, we can say that $m_{\lambda_u} \neq m_\lambda$ and $v_{\lambda_u} \neq v_\lambda$ since $0 < \lambda < \infty$. From the definition of λ-optimal path, $m_{\lambda_u} + \lambda_u v_{\lambda_u} < m_\lambda + \lambda_u v_\lambda$ and $m_{\lambda_u} + \lambda v_{\lambda_u} > m_\lambda + \lambda v_\lambda$. Rewriting these, we get

$$\lambda_u(v_{\lambda_u} - v_\lambda) < m_\lambda - m_{\lambda_u}, \;\; \lambda(v_{\lambda_u} - v_\lambda) > m_\lambda - m_{\lambda_u}. \tag{3}$$

From the two inequalities we get $(\lambda - \lambda_u)(v_{\lambda_u} - v_\lambda) > 0$ and since $\lambda > \lambda_u$, it follows that $v_{\lambda_u} > v_\lambda$ and $m_\lambda > m_{\lambda_u}$.We get an expression for λ_u by taking the derivitive of (2) and its negative inverse $\lambda_u = -1/\frac{\partial v}{\partial m}|_{m=m_0} = \frac{(d-m_0)^2}{2v_\infty(d-m_0)} = \frac{d-m_0}{2v_\infty}$. From this and (3), and since $d - m_0 > d - m_\lambda$ and $v_\lambda > v_\infty$,

$$\frac{m_\lambda - m_{\lambda_u}}{v_{\lambda_u} - v_\lambda} > \frac{d-m_0}{2v_\infty} > \frac{1}{2}\frac{d-m_\lambda}{v_\lambda}. \tag{4}$$

Since $m_\lambda > m_{\lambda_u}$ and $d - m_\lambda > 0$, $\frac{1}{\frac{m_\lambda - m_{\lambda_u}}{d-m_\lambda}+2} < \frac{1}{2}$. From this and (4), $\frac{1}{\frac{m_\lambda - m_{\lambda_u}}{d-m_\lambda}+2} < \frac{m_\lambda - m_{\lambda_u}}{v_{\lambda_u} - v_\lambda}\frac{v_\lambda}{d-m_\lambda}$. We can rewrite this as follows:

$$\frac{v_{\lambda_u} - v_\lambda}{v_\lambda} < \frac{m_\lambda - m_{\lambda_u}}{d-m_\lambda}\left(\frac{m_\lambda - m_{\lambda_u}}{d-m_\lambda} + 2\right) \rightarrow \;\; \frac{v_{\lambda_u}}{v_\lambda} < \left(\frac{m_\lambda - m_{\lambda_u}}{d-m_\lambda} + 1\right)^2$$
$$\rightarrow \frac{v_{\lambda_u}}{v_\lambda} < \left(\frac{d-m_{\lambda_u}}{d-m_\lambda}\right)^2 \rightarrow \;\; \frac{d-m_\lambda}{\sqrt{v_\lambda}} < \frac{d-m_{\lambda_u}}{\sqrt{v_{\lambda_u}}}.$$

Thus, for any $\lambda > \lambda_u$, the λ-optimal solution is worse than the λ_u-optimal solution. Thus, there is no need to search the area with the λ that is larger than λ_u. Therefore, whether λ-optimal solution is the same with the λ_u-optimal solution or not we can find the optimal solution by searching with $\lambda \leq \lambda_u$.

Theorem 4.3. *The optimal path can be found by searching only with the λ values lower bounded by λ_l, the negative inverse of the tangent to the current λ-optimal parabola at the intersection of the current λ-optimal parabola and 0-optimal search line.*

Proof. Similar to the proof of Theorem 4.2.

Theorems 4.1, 4.2, and 4.3 lead to the Parametric Search algorithm (see Algorithm 1) for finding the best route that maximizes the probability of arriving at the destination within a given deadline. In lines 3 and 4, the 0-optimal and ∞-optimal paths are searched with a shortest-path algorithm (e.g., Dijkstra's algorithm or A^* search algorithm). If the two found paths are the same, the algorithm terminates. If they are different, the first candidate region consisting of the three points denoted in line 7 is pushed into the queue. Candidate regions are evaluated for searching. The conditions in lines 10, 21, and 23 come from Theorem 4.1, and those in lines 12 and 14 from Theorems 4.2 and 4.3, respectively. If the candidate region does not need to be searched, the algorithm continues with the next region. Otherwise, the region is searched with the λ value determined by the left and right path of the region (line 11) and possibly modified by the upper and lower bounds (lines 13 and 15) in line 16.

Algorithm 1. PARAMETRIC-SEARCH

Data: Graph with mean and variance of each edge, origin, and destination
Result: The optimal path

1: $bestPath \leftarrow \varnothing$
2: $Regions$ = [] : FIFO queue containing candidate regions.
3: $path_0 \leftarrow$ SEARCH-λ-OPTIMAL-PATH(0)
4: $path_\infty \leftarrow$ SEARCH-λ-OPTIMAL-PATH(∞)
5: **if** $path_0 == path_\infty$ **then**
6: **return** $path_0$
7: $Regions.push(Region(l : path_0,\ r : path_\infty,\ p : (path_0.mean, path_\infty.var)))$
8: calculate λ_l and λ_u
9: **while** $(R \leftarrow Regions.pop())\ != \varnothing$ **do**
10: **if** $R.probe.\varphi < bestPath.\varphi$ **then** *continue*
11: $\lambda \leftarrow -\frac{R.l.mean - R.r.mean}{l.var - r.var}$
12: **if** $\lambda \geq \lambda_u$ **then**
13: **if** λ_u was not searched **then** $\lambda \leftarrow \lambda_u$ **else** *continue*
14: **if** $\lambda \leq \lambda_l$ **then**
15: **if** λ_l was not searched **then** $\lambda \leftarrow \lambda_l$ **else** *continue*
16: $path \leftarrow$SEARCH-λ-OPTIMAL-PATH(λ)
17: **if** $path\ != R.l\ and\ path\ != R.r$ **then**
18: **if** $path.\varphi > bestPath.\varphi$ **then**
19: $bestPath \leftarrow path$, update λ_l
20: locate $probe_l$ and $probe_r$
21: **if** $probe_l.\varphi > bestPath.\varphi$ **then**
22: $Regions.push(Region(l : R.l,\ r : path,\ p : probe_l))$
23: **if** $probe_r.\varphi > bestPath.\varphi$ **then**
24: $Regions.push(Region(l : path,\ r : R.r,\ p : probe_r))$
25: **return** $bestPath$

4.5 Correctness

Algorithm 1 finds the optimal solution in a finite number of λ-optimal searches. The paths in the region we exclude from the exhaustive enumeration using the extreme points cannot be optimal by Theorem 4.1. The paths in the region we excluded using the upper and lower bound of λ cannot be optimal due to Theorems 4.2 and 4.3. Since the number of required λ-optimal searches is upper bounded by $2N_e - 1$ as described in Section 4.2, the algorithm finds the optimal solution in a finite number of searches.

4.6 Running Time

As shown by Nikolova, et al. in [9] based on [2], there are total $N^{\Theta(logN)}$ extreme points in the $m - v$ plane, where N is the number of nodes in the network. Compared to their algorithm that searches every extreme point, our algorithm does not invoke unnecessary searches yielding an average running time of $O(N^2 log^4 N)$, where N^2 term is due to Dijkstra's runs for each λ-optimal search.

The intuition is as follows. Each new path point found with a λ-optimal search yields two candidate regions. Our algorithm only searches the candidate region if its probe point is outside the current optimal parabola. The parabola passing through the newly found path point will almost always divide the two probe points causing one of them to lie outside the parabola. The adverse case where both probe points are outside the parabola happens rarely when the current optimal parabola meets the current λ-optimal search line twice within the interval determined by the two probe points. The decision to remove candidate regions without searching them depends on the distribution of path points in the $m-v$ plane and the deadline. Thus, the running time of our algorithm can be described probabilistically.

Let the probability that both candidate regions are searched be p_2 and the probability that neither candidate region is searched be p_0. We give the running time bound of our algorithm taking the varying p_2 and p_0 into account in Theorem 4.4. We use two lemmas: the running time of our algorithm when p_2 and p_0 are given (Lemma 4.2) and how p_2 and p_0 vary as algorithm iterations proceed (Lemma 4.3).

Lemma 4.2. *Let the probability that both candidate regions are searched be p_2 and the probability that neither candidate region is searched be p_0. If $p_2 \leq p_0$, the average running time of our algorithm is $O(N^2 log^2 N)$, where N is the number of nodes in the graph.*

Proof. We use a binary tree to represent a hierarchy of candidate regions created by the λ-search sequence. Each node represents a candidate region and its children nodes represent the two candidate regions created by searching the current region. Let h be the height of the tree, and let the function $num(\mathscr{N})$ be defined as the total number of candidate regions to search for a tree with a root node $\mathscr{N}$ and left and right nodes $\mathscr{L}$ and $\mathscr{R}$. Then, $h = O(log(N^{logN})) = O(log^2 N)$ and

$$num(\mathscr{N}) = \begin{cases} 1 + num(\mathscr{L}) + num(\mathscr{R}) & \text{with probability } p_2, \\ 1 + num(\mathscr{L}) & \text{with probability } \frac{1-p_2-p_0}{2}, \\ 1 + num(\mathscr{R}) & \text{with probability } \frac{1-p_2-p_0}{2}, \\ 1 & \text{with probability } p_0. \end{cases}$$

The conditional expectation of $num(\mathscr{N})$ given $num(\mathscr{L})$ and $num(\mathscr{R})$ is described as follows:

$$E[num(\mathscr{N})|num(\mathscr{L}), num(\mathscr{R})] = 1 + \frac{1+p_2-p_0}{2}\,(num(\mathscr{L}) + num(\mathscr{R}))$$

Then, the expectation of $num(\mathscr{N})$ can be expressed as follows, taking expectation over $num(\mathscr{L})$ and $num(\mathscr{R})$ on the above conditional expectation.

$$\begin{aligned} E[num(\mathscr{N})] &= E[E[num(\mathscr{N})|num(\mathscr{L}), num(\mathscr{R})]] \\ &= 1 + \frac{1+p_2-p_0}{2}\,(E[num(\mathscr{L})] + E[num(\mathscr{R})]) \end{aligned} \tag{5}$$

Since the expectation of $num(\mathcal{N})$ depends only on the height of the tree, we can define a function $num_e(h)$ as the expected number of nodes given the height h. Then, for a node $\mathcal{N}$ whose height is h and whose two children are $\mathcal{L}$ and $\mathcal{R}$, we have $E[num(\mathcal{N})] = num_e(h)$ and $E[num(\mathcal{L})] = E[num(\mathcal{R})] = num_e(h-1)$. From (5), $num_e(h)$ can be written as:

$$num_e(h) = \begin{cases} 1 + (1 + p_2 - p_0)\, num_e(h-1) & \text{if } h \geq 1, \\ 0 & \text{if } h = 0. \end{cases}$$

Solving the above recursive equations, noting that $(1 + p_2 - p_0) \leq 1$, we get:

$$num_e(h) = 1 + (1 + p_2 - p_0) + \cdots + (1 + p_2 - p_0)^{h-1} \leq h = O(log^2 N).$$

Thus, the running time is $O(N^2 log^2 N)$ when we use Dijkstra's algorithm for each λ-optimal search.

Lemma 4.3. *$p_2(i) \leq p_0(i), \forall i > n, n = O(log^2 N)$, where i is the index of iterations. In other words, p_0 becomes larger than p_2 around $log^2 N$ iterations.*

Proof. First, consider how $p_2(i)$ and $p_0(i)$ vary as iterations proceed. We have three cases: (a) two regions are left in the queue, (b) one region is left in the queue, or (c) no region is left in the queue. Case (a) occurs only when the λ-optimal search line is in the neighborhood of the tangent to the parabola at the point being considered. The size of the neighborhood depends on the distance between the two probe points (e.g., if the distance is large, even a large difference in the slopes can result in the retention of both points.) If the distance is small, only a small difference in the slopes will lead to the retention of both points. Since the distance gets smaller as iterations proceed, $p_2(i)$ decreases monotonically. On the other hand, the probability of case (c) (e.g., both candidate regions are removed) monotonically increases as the distance between the probe points gets smaller. The optimal parabola gets flatter and the lower bound of λ (λ_l) increases as a better λ-optimal path is found, which also causes $p_2(i)$ to decrease and $p_0(i)$ to increase.

Consider the point when $p_2(i)$ becomes smaller than $p_0(i)$. When a single child is retained at each step, it takes $log^2 N$ iterations until the optimal path is found because we have a binary tree. Thus the probability that the search algorithm retains one child decreases fast beyond $log^2 N$ iterations. This results in $p_0(i)$'s fast increase since its increasing rate is related to the decreasing rate of $1 - p_2(i) - p_0(i)$ by $\frac{\Delta p_0(i)}{\Delta i} = -\frac{\Delta(1-p_2(i)-p_0(i))}{\Delta i} + (-\frac{\Delta p_2(i)}{\Delta i})$. Thus, $p_0(i)$ becomes larger than $p_2(i)$ around $i = log^2 N$. Fig. 3 illustrates these behaviors of $p_2(i)$ and $p_0(i)$.

Theorem 4.4. *The average running time of Algorithm 1 is $O(N^2 log^4 N)$.*

Proof. The total running time is determined by considering the computation before and after $p_2 \leq p_0$ is satisfied separately. By Lemma 4.3 the computational cost before $p_2 \leq p_0$ is satisfied is $O(N^2 log^2 N)$. Then, we have $O(log^2 N)$ candidate regions. In the worst case, all the remaining candidate regions should be searched.

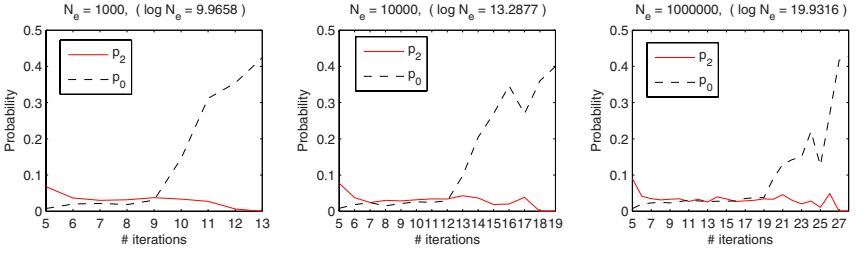


Fig. 3. p_2 and p_0 was calculated from 5000 random convex hulls with N_e extreme points.

Since the height of the sub-trees spanning from those candidate regions is bounded by h and they all satisfy $p_2 \leq p_0$, the computational cost after $p_2 \leq p_0$ is satisfied is $O(N^2 log^4 N)$ by Lemma 4.2. Thus, the overall running time is $O(N^2 log^4 N)$.

5 Algorithm Evaluation

We have evaluated empirically the performance of Algorithm 1 against the exhaustive λ-optimal search proposed in [9] using simulation data and real data from the taxi database.

5.1 Experimental Data

Grid Structure: The simulation data set is a square bidirectional grid structure where each edge has a random mean and variance uniformly distributed between 0 and 1. Grid structures of size from 10×10 to 100×100 are used. The origin and destination are two diagonally opposite points.

Physical Road Network: The Delay Statistics Map built using the taxi database is used as a physical test bed. The map has about 29,000 nodes and 39,000 edges. It is dense around the city area and more sparse in rural areas.

5.2 Running Time

The running time of Algorithm 1 is compared with the exhaustive λ-optimal searches using the two data sets above. Fig. 4 shows the results on the simulation data. The individual effect of each method in Sections 4.3 and 4.4 on the running time is also shown. Algorithm 1 with both methods runs fastest. The speedup is by at least a factor of 10 over the algorithm in [9]. The speedup is due to the reduced number of λ-optimal searches. For the algorithm in [9], the number of λ-optimal searches gradually increases from 17 to 119 as the number of nodes increases from 100 to 10000, whereas it increases from 5 to 7 for Algorithm 1.

The running time of Algorithm 1 on the Delay Statistics Map for a route 144km long is 14 seconds with 5 λ-optimal searches when the deadline is 3 hours. The same

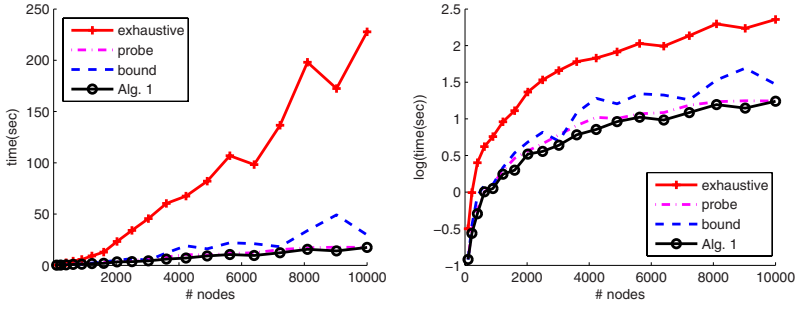


Fig. 4. Running time measured at the square bidirectional grid structure where each edge has a random mean and variance between 0 and 1 with a deadline of half grid size. "exhaustive" is the exhaustive λ-optimal search, "probe" is just applying the candidate-region probing method, "bound" is just applying the bounds of λ and "Alg. 1" is Algorithm 1.

query took 178 seconds when we used the exhaustive λ-optimal search yielding 75 λ-optimal searches.

5.3 Path Example

Fig. 1 shows different optimal paths from an origin (the arrow) to a destination (the "D"), according to three different criteria: the minimum distance route (this is the same route recommended by Google Maps and is indicated as the topmost route), the minimum expected time route (the middle route), and the maximum probability route with a deadline of 14 minutes (the bottom route). Our system estimates that minimum distance route (which is 3.1 miles) will take 18 minutes on a Tuesday afternoon. Our system's minimum expected time route (which is 3.5 miles) takes only 11 minutes and 45 seconds. The maximum probability route (which is 4.1 miles) takes 11 minutes and 51 seconds with 90.3% guarantee of arriving on time. The minimum distance route and the minimum expected time route have lower probabilities at 1% and 88.5%, respectively.

5.4 Overall Path Goodness

Four different routes from an origin to a destination were examined using taxi paths and human test driving. The estimated mean of each path from 7 am to 9 pm was 872 seconds, 899 seconds, 816 seconds, and 795 seconds for route 1 (6.9 km), route 2 (7.2 km), route 3 (6.7 km), and route 4 (6.2 km) respectively. The measured mean was 869 seconds, 895 seconds, 811 seconds, and 799 seconds. Thus, the estimated minimum expected time path, route 4, agrees with the measurement. Fig. 5 gives the maximum probability path. The estimated probability is similar to the measured probability. From both estimation and measurement, for a deadline less than or equal to 12 minutes, route 3 is the best, but for a deadline larger than 12 minutes route 4

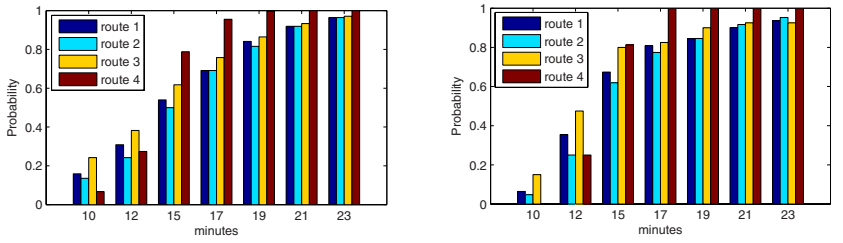


Fig. 5. The probability of arriving within a given deadline from an origin to a destination. (Left) Estimation. (Right) Measurement.

is the best. We can observe that route 4, which is the minimum expected time path is the worst path for a deadline less than 12 minutes.

5.5 Independent Gaussian Assumption

To test the independent Gaussian assumption, we identified a route with a large number of travel samples in the taxi trajectories database. We used a path from an origin to a destination, which is 1.12 km long and has 5 intersections and 6 road segments. From Fig. 6 (Right), we can observe that the empirical data and the estimation by independent Gaussian assumption is very close. More specifically, Fig. 7 shows that the empirical data is very similar to the Gaussian distribution, especially in the probability interval from 0.05 to 0.95. Thus, our assumptions will hold well for the stochastic planning for reaching the destination with the probability in this range. The discrepancies observed over 0.95 and under 0.05 show the limitation of our algorithms due to our assumption. For example, as shown in Fig. 7 (Bottom left), our system will estimate that the users can reach the goal with 99% probability if they leave 230 seconds before the deadline, but the empirical data shows that we will get only a 97% chance, and if users want a 99% guarantee they should leave about 270 seconds before the deadline. The discrepancy over 0.95 is caused by some unusual long delay, which might be due to unexpected events, construction

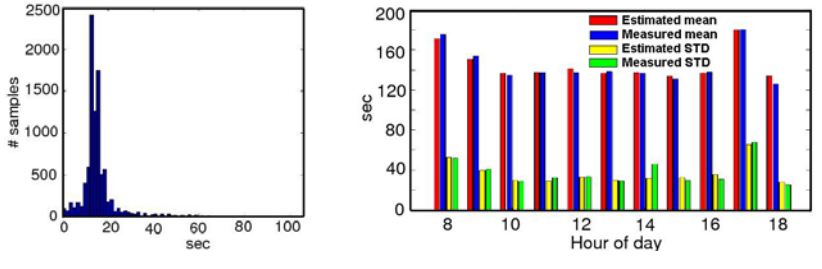


Fig. 6. (Left) Delay distribution for one segment. (Right) The comparison of mean and standard deviation between empirical measurement and estimation for various time slots.

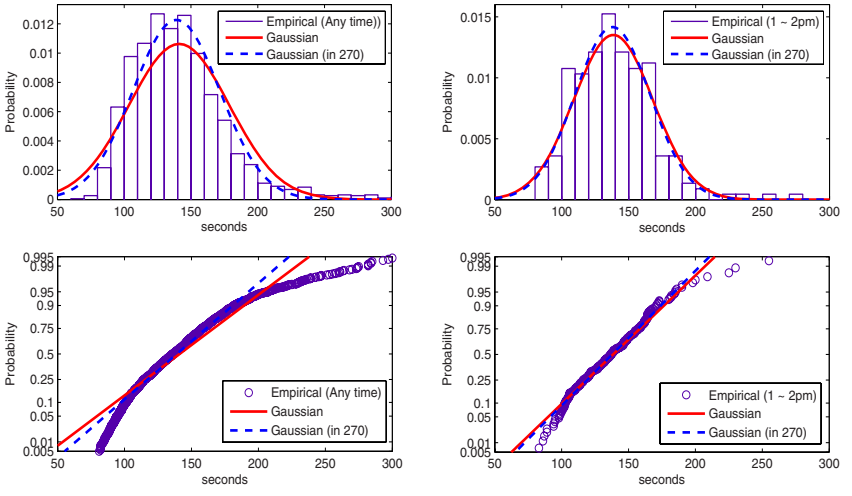


Fig. 7. "Empirical" indicates the travel time measurement by driving, "Gaussian" indicates the Gaussian fit for the entire data and "Gaussian(in 270)" indicates the Gaussian fit using only the data in 270 seconds. (Top left) Histogram of empirical travel time data and Gaussian fits for any time for weekdays. (Bottom left) Probability comparison between empirical data and Gaussian fits for any time for weekdays, where the Y axis was scaled to make the Gaussian CDF linear (Right) Corresponding plots to the left plots for $1 \sim 2$ pm for weekdays

work, or data gathering noise such as taxi drivers' intentional stops or slow drives. The discrepancy below 0.05 is due to the Gaussian distribution spanning to the negative value whereas the travel time cannot be negative. We observe less discrepancy in case of the right plots, which use only the data from $1 \sim 2$ pm whereas the left plots are for entire hours. This result suggests that narrowing the data by conditions that affect the traffic delays such as time of day makes the delay distribution look more like independent Gaussian distribution. Thus, in our ongoing research, we are investigating the proper conditions that constrain the traffic delays.

This observation provides some evidence that the independent Gaussian assumption used for Algorithm 1 holds for the taxi trajectories database, but more testing on road segments with more associated travel data is necessary. As the database grows every day, we plan to continue this validation.

6 Conclusion and Future Work

We developed efficient stochastic shortest-path algorithms that can be used for a practical traffic information system. We evaluated the system with actual measured travel time for selected routes, and observed that our system's optimal path and travel time estimates are close to reality.

In the future, we are interested in developing path planning algorithms for multiple users. We are also interested in improving our algorithms and system by

considering dependencies of each road segment and by using better modeling of delay distributions. We are currently extending the algorithms to integrate current traffic information with historical information to make more accurate estimations and to predict the future traffic conditions. Considering various conditions affecting traffic like weather, construction work, and events is also part of our current plans. Finally, we plan to integrate this planning system with autonomous vehicles.

Acknowledgments

We thank PlanetTran for participating in the taxi data collection. This work has been supported in part by NSF grants EFRI-0710252 and IIS-0426838, by the SWARMS MURI, and by a Samsung Scholarship. We are grateful for this support.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Mathematics of Operations Research 16(3), 580–595 (1991)
2. Carstensen, P.: The complexity of some problems in parametric linear and combinatorial programming. Ph.D. Thesis, Mathematics Dept., U. of Michigan, Ann Arbor, Mich. (January 1983)
3. Chabini, I.: Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. Transportation Research Record 1645, 170–175 (1998)
4. Chrobok, R., Wahle, J., Schreckenberg, M.: Traffic forecast using simulations of large scale networks. In: IEEE Conference on Intelligent Transportation Systems, Oakland, CA, USA (August 2001)
5. Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A.K., Shih, E., Balakrishnan, H., Madden, S.: CarTel: A Distributed Mobile Sensor Computing System. In: 4th ACM SenSys., Boulder, CO (November 2006)
6. Loui, R.: Optimal paths in graphs with stochastic or multidimensional weights. Communications of the ACM 26(9), 670–676 (1983)
7. Murthy, I., Sarkar, S.: Exact algorithms for the stochastic shortest path problem with a decreasing deadline utility function. European Jounal of Operational Research 103, 209–229 (1997)
8. Nikolova, E., Brand, M., Karger, D.: Optimal route planning under uncertainty. In: International Conference on Automated Planning and Scheduling (2006)
9. Nikolova, E., Kelner, J.A., Brand, M., Mitzenmacher, M.: Stochastic shortest paths via quasi-convex maximization. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 552–563. Springer, Heidelberg (2006)
10. Sanwal, K.K., Walrand, J.: Vehicles as probes. Technical Report UCB-ITS-PWP-95-11, California Partners for Advanced Transit and Highways (PATH) (January 1995)
11. Schrank, D., Lomax, T.: The 2007 urban mobility report. Annual report, Texas Transportation Institute, The Texas A&M University System (September 2007)
12. Sigal, C.E., Pritsker, A.A.B., Solberg, J.J.: The stochastic shortest route problem. Operations Research 28(5), 1122–1129 (1980)

13. Wellman, M.P., Ford, M., Larson, K.: Path planning under time-dependent uncertainty. In: 11th Conference on Uncertainty in Artificial Intelligence, August 1995, pp. 532–539 (1995)
14. Xu, H., Dailey, D.J.: Real time highway traffic simulation and prediction using inductance loop data. In: Vehicle Navigation and Information Systems Conference, Seattle, WA, USA (July 1995)
15. Yoon, J., Noble, B., Liu, M.: Surface street traffic estimation. In: MobiSys 2007: Proceedings of the 5th international conference on Mobile systems, applications and services, pp. 220–232. ACM, New York (2007)

On Probabilistic Search Decisions under Searcher Motion Constraints

Timothy H. Chung

Abstract. This article presents a sequential decision-theoretic formulation for conducting probabilistic search for a stationary target in a search region. A general recursion expression describing the evolution of the search decision (i.e., presence or absence of the target) is derived, which relates the temporal sequence of imperfect detections, both false positives and false negatives, to the spatial search conducted by a search agent. This relationship enables quantification of the decision performance – time till decision – for a given search strategy. Also, the role of searcher motion constraints, represented by a *search graph*, on the time till decision is characterized by the second smallest eigenvalue of the Laplacian of this graph. Numerical studies demonstrate this relationship.

1 Introduction

As intelligent systems are increasingly endowed with greater autonomy and information processing capabilities, the ability to make decisions is required to execute higher level tasks. Robotic search of an environment to ascertain the presence or absence of a target provides an example of such a task. Probabilistic search can be defined as the search for a target of interest by one or more search agents in a spatially bounded search region. A general framework which incorporates the probabilistic nature of systems and integrates imperfect observations appropriately is necessary in many applications, and the theoretical foundation presented herein may provide both insight and intuition in the dynamics of decision making for probabilistic search.

The theory of spatial probabilistic search first arose in the operations research community [16]. Classic texts include [27, 23], and the most recent survey of classical search problems and techniques is presented in [2]. These works largely investigate the problem of *nonadaptive* search, in which a search plan over the discretized

Timothy H. Chung
Department of Operations Research, Naval Postgraduate School, Monterey, CA 93943 USA
e-mail: thchung@nps.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 501–516.
springerlink.com

search region is optimized at the beginning of search and executed without incorporation of new information due to observations. These earlier works do not consider the possibility of false positive detections (a.k.a. "false alarms") in their formulations, limiting the analysis to having only false negative observations (a.k.a. "missed detections") as a means of enabling analytic results for the optimization. However, realistic scenarios comprise both types of errors, and the problem of false alarms becomes even more relevant in cases where search takes place in the presence of clutter [15], such as the search for a person in a crowd or a hiker lost in the woods. Recent efforts [17] begin to address both detection error types, but still rely on the independence of cells such that detections in one cell do not affect the probability in other cells. This lack of dependence between cells fails to capture and incorporate dynamic search-related information into the search planning process. For example, positive detection of the target in a given cell (albeit possibly erroneous) should reflect that it is believed by the search agent less likely to be present in another cell.

Independently, with advances in autonomous systems and greater demand for persistent information gathering capabilities, there has been a growth of research in employing robotic agents to conduct search-related tasks. Methods for information-based motion planning, where observations of the environment or situation are used in conjunction with feedback to enable online improvement of robot trajectories, have also been applied to the spatial search problem described above. Such algorithms for robotic searchers address a diverse range of topics, such as exploration and mapping [4], uncertain motion planning [1], visibility constraints [19] (such as line-of-sight sensors), and robotic coverage [6]. In particular, the authors of [3] construct probability maps for possible target locations and deploys coordinated searchers (i.e., unmanned aerial vehicles) and employ greedy or switching algorithms to optimize the probability of detecting a target. Analysis conducted in [9] provides a polynomial-time algorithm for approximate solutions, as well as bounds on the resulting sub-optimality of search trajectories. In addition, [25] employ obstacle and evader probability maps with greedy algorithms to demonstrate a multi-agent, hierarchical implementation. More recently, work in [18] applies elements of classical search theory to indoor environments, where a smaller state space enables explicit calculation of optimal paths for maximal detection probability when travel time between rooms is considered.

This paper proposes a decision-theoretic approach for examination of the search problem described above, by representing the search task as a decision for which sequentially obtained information is integrated over time. In this manner, the temporal evolution of the decision can be examined, with the decision performance measured by the time till the search decision is made. The main contributions of this paper include further theoretical and algorithmic developments of the search-as-a-decision approach posed in [7, 8], including lower bounds on the expected decision time found by examining the special case of perfect detections. However, whereas this previous work considered the search strategy employed by a mobile searcher to determine *where* it should search, this paper examines the influence of different degrees of mobility, i.e., *how* the searcher reaches its goal, on the search decision performance. Constraints on the search agent's motions are formulated as an

associated search graph, for which a relationship between the performance of the decision, measured by the decision time, and the second smallest Laplacian eigenvalue, known as *algebraic connectivity*, of this search graph is established. Section 2 formulates the probabilistic search problem described above in the context of a search decision in the presence of imperfect detections. Further, the recursion expression for the decision evolution and study of some special cases are also presented. Section 3 defines the search graph as it represents the constraints on searcher motions, and examines the relationship between properties of this graph representation and the time until the probabilistic presence or absence of the target is determined. A summary of conclusions and discussion of future work are provided in Section 4.

2 Probabilistic Search as a Sequential Decision

The problem of probabilistic search for an object can be thought of as a binary hypothesis test, which represents the uncertainty in the target's presence or absence in the search region. Such a formulation encapsulates two objectives in a unified manner: localization of a target which is assumed present in the search region combined with appropriate termination of the search process when the target, in fact, is not present. The hypothesis test utilizes a binary (i.e., Bernoulli) random variable, H, which takes a value of either zero or one, such that the probability that the stationary target is present in the search region is given by $P(H=1)$.

2.1 Search Environment

Consider a discretization of the search environment, $\mathscr{A}$, into $|\mathscr{A}|$ cells. Use of discrete cells may reflect physical structure present in the search region (such as distinct rooms in a building) or may capture the effect of finite detection range of the sensor. As an example, consider a search-and-rescue (SAR) objective for a lost hiker in the woods using an unmanned aircraft equipped with a downward-facing camera. Terrain features (e.g., rivers, ridgelines) may divide the area into natural discrete sectors, or physical characteristics, such as the camera's field-of-view and altitude, may drive the discretization of the environment. The presented framework is independent of the cellular decomposition approach chosen (e.g., trapezoidal, Voronoi), and other issues (e.g., visibility, distance, resolution) can be used to determine the exact method.

The presence or absence of the target within a specific cell $c \in \{1,\ldots,|\mathscr{A}|\}$ in the search region is also described by a Bernoulli random variable, X_c, such that the probability, or the searcher's *cell belief*, that the target is located in the c^{th} cell at time $t=0$ is given by $p_c^0 \stackrel{\text{def}}{=} P(X_c=1)$. Note that one can define all space not in the search region as a *virtual* cell, indexed by $\emptyset$, such that $p_\emptyset^0 \stackrel{\text{def}}{=} P(H=0) = 1-P(H=1)$. These initial cell belief values form an arbitrary probability distribution and can represent

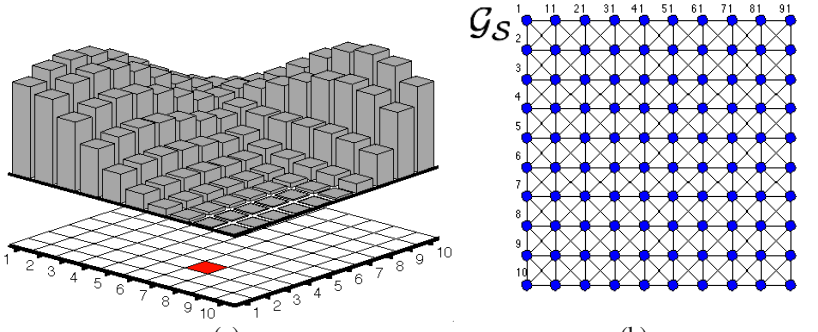


Fig. 1. (a) Example of a search region and initial belief distribution. The search region is a square 10×10 grid. The initial belief distribution for this example is given by a mixture of two bivariate Gaussians, with means and covariances $\mu_1 = (1,3), \mu_2 = (8,10)$ and $\sigma_1 = (7,15), \sigma_2 = (10,2)$, respectively, normalized to sum to $B(0) = \frac{1}{2}$. The target is shown present in grid cell $(7,3)$. (b) Illustration of an example search graph, $\mathcal{G}_{\mathcal{S}}$, where a searcher can move to all neighboring vertices, including diagonals, given by an augmented grid graph with hop length one.

any prior knowledge available to the searcher before commencing the search[1]. In the lost hiker example, information such as last known whereabouts and input by subject-matter experts can be incorporated to construct this initial belief distribution.

The summation of cell beliefs over all cells within the search region composes the *aggregate belief*, which is the overall cumulative probability that the target is present somewhere in $\mathcal{A}$, i.e., $\sum_c^{|\mathcal{A}|} p_c^0 = P(H=1) \stackrel{\text{def}}{=} B(0)$. An illustrative example of the search region and the initial belief distribution is depicted in Figure 1(a), where the search region is a square grid of 10×10 cells ($|\mathcal{A}| = 100$), and the height of each discrete gray bar represents the cell belief of that cell.

As the search agent takes observations in different cells over time, the cell beliefs, p_c^t, are recursively updated, thereby changing the aggregate belief, $B(t)$, that the target is present. This sequential evolution of the belief depends on the sequence of cells the search agent visits and is described in Section 2.3.

2.2 Search Agent Modeling

Consider the single searcher whose location at time t within the search region is denoted $s(t) \in \{1, \ldots, |\mathcal{A}|\}$. Initially at $s(0)$, the search agent follows a trajectory through time t given by $\mathcal{S}(t) = \{s(1), \ldots, s(t)\}$.

Given the decomposition of the search region, the collection of cells can be equivalently represented as a set $\mathcal{V}$ of $|\mathcal{A}|$ vertices. Definition of a particular *search graph*, $\mathcal{G}_{\mathcal{S}} = (\mathcal{V}, \mathcal{E})$, of the search agent requires specification of the set of edges, $\mathcal{E}$.

[1] In the absence of such prior information, a noninformative prior (i.e., uniform) distribution may be used. In this case, $p_c^0 = \frac{1}{|\mathcal{A}|}, \forall\, c \in \mathcal{A}$.

These edges represent the motion constraints of the searcher, in that the presence of an edge between vertices k and l signifies that the searcher is able to move between the two corresponding cells in the search region in a single time step at unit cost. Figure 1(b) illustrates an example search graph where each cell is connected to all adjacent (including diagonal) cells, termed an augmented grid graph. Note that environmental constraints such as the presence of obstacles can also be incorporated in the search graph by exclusion of edges incident to those vertices corresponding to obstacle locations. In the case of unmanned aircraft, a given search graph reflects the speed and maneuverability of the aircraft to reach different destinations, subject to requirements for high-fidelity observations (e.g., need to stabilize prior to imaging).

The search graph, $\mathscr{G}_{\mathscr{S}}$, is completely described by its adjacency matrix, $\mathrm{Adj}(\mathscr{G}_{\mathscr{S}})$, where the $(k,l)^{\text{th}}$ element is one if and only if there exists an edge between cells k and l, and zero otherwise. Note that the k^{th} cell is assumed to be adjacent to itself (i.e., the search agent can choose to remain at its current location in the next time step). The search graph may represent constraints on the searcher dynamics, such as maximum speeds, or may reflect the physical nature of the search environment, such as hallways and doorways. Further discussion of the search graph is given in Section 3.

The search agent is endowed with target detection capabilities, which are subject to uncertainty in the form of false positive and false negative errors [7]. This general representation of probabilistic observations encompasses a vast class of sensing modalities, ranging from those that can utilize a simple threshold rule (e.g., is temperature above or below a specified value?) to those that require significant data processing resources (such as machine vision methods for target feature recognition, as could be used in the lost hiker example).

Let $Y_{s(t)}$ be the search agent's detection measurement of cell $s(t)$ at time step t, where each $Y_{s(t)}$ is also modeled as a Bernoulli random variable representing the uncertainty in the detection. For a given observation in cell $s(t)$ at time t, the random variable $Y_{s(t)}$ takes on a value of one if the target is detected, and zero if no target is detected. The detection likelihood function describes the conditional probability of observing $Y_{s(t)}$ given the true presence or absence of the target in the c^{th} cell, X_c, and is typically prescribed by the particular choice of detector. Incorrect detections of both types are possible in general scenarios, and are characterized by the detector's error probabilities (assumed independent of time and cell), defined as:

$$P(Y_c = 1 | X_c = 0) \stackrel{\text{def}}{=} \alpha \qquad \text{(false positive)}$$

$$P(Y_c = 0 | X_c = 1) \stackrel{\text{def}}{=} \beta \qquad \text{(false negative)}$$

As a final definition, define $\mathscr{Y}(t) = \left\{ Y_{s(1)}, \ldots, Y_{s(t)} \right\}$ as the history of observations taken by the search agent through time t.

2.3 Decision Evolution

The integration of information from sequential observations represents an evolution of the decision in time. This evolution is a stochastic process which depends on

both the collection of imperfect detections and the search agent's trajectory through the search region. The objective in *active decision making* is to determine a search control strategy which improves the performance of the decision, as measured by some metric, such as minimal time till detection [7, 8]. As with the control of any dynamical system, the recursion expressions presented in this section show how the decision evolution is a dynamical process itself, which can be controlled by input of the search agent's location.

The decision evolves as new information is combined with previous information by use of a Bayesian update rule. First, for an observation at time t in cell $s(t)$, define the following functions of observation random variable $Y_{s(t)}$:

$$\begin{aligned}\Phi\left(Y_{s(t)}\right) &\stackrel{\text{def}}{=} \left(1-Y_{s(t)}\right)(1-\alpha)+Y_{s(t)}\ \alpha \\ \Psi\left(Y_{s(t)}\right) &\stackrel{\text{def}}{=} \left(1-Y_{s(t)}\right)\ \beta+Y_{s(t)}\ (1-\beta),\end{aligned}$$

where α and β are the false alarm and missed detection probabilities, respectively. These functions are intimately related to the marginalization of the observation:

$$P(Y_{s(t)}) = \Phi\left(Y_{s(t)}\right)\cdot\left(1-p^t_{s(t)}\right)+\Psi\left(Y_{s(t)}\right)\cdot p^t_{s(t)},$$

which gives the probability that this observation is either a zero or a one. This term provides the normalization constant for the Bayesian observation updates of cell belief probabilities, given in terms of the above functions as:

$$p^t_c = \frac{\Theta_c\left(Y_{s(t)}\right)\ p^{t-1}_c}{\Phi\left(Y_{s(t)}\right)+\Omega\left(Y_{s(t)}\right)\ p^{t-1}_{s(t)}}, \tag{1}$$

where the following further definitions facilitate the analysis:

$$\begin{aligned}\Omega\left(Y_{s(t)}\right) &\stackrel{\text{def}}{=} \Psi\left(Y_{s(t)}\right)-\Phi\left(Y_{s(t)}\right) \quad = \quad \left(2Y_{s(t)}-1\right)(1-\alpha-\beta) \\ \Theta_c\left(Y_{s(t)}\right) &= \begin{cases}\Psi\left(Y_{s(t)}\right), \text{ if } s(t)=c \\ \Phi\left(Y_{s(t)}\right), \text{ if } s(t)\neq c\end{cases}\end{aligned}$$

The term, $\Theta_c\left(Y_{s(t)}\right)$, enables a general representation for when the observation is taken in the cell where the update is conducted (i.e., $c = s(t)$) or when the observation occurred in some other cell (i.e., $s(t) \neq c$). Since an observation anywhere affects beliefs everywhere (by their inter-dependence), Equation 1 enables succinct and straightforward computation of these updated cell beliefs.

For an iterative implementation, the expression given by Equation 1 is sufficient for each single time step update, requiring only the belief distribution from the previous time step and the current observation information, demonstrating the usefulness of a canonical Bayesian filtering approach. Alternatively, one may wish to consider the entire time evolution of a given cell's belief over the course of observations, $\mathscr{Y}(t)$, and search agent motions, $\mathscr{S}(t)$, as a function of the initial belief distribution, $p^0_c, \forall c$:

$$p_c^t = \frac{\prod_{k=1}^{t} \Theta_c\left(Y_{s(k)}\right) p_c^0}{\prod_{k=1}^{t} \Phi\left(Y_{s(k)}\right) + \sum_{k=1}^{t} \left(\prod_{l=1}^{k-1} \Theta_{s(k)}\left(Y_{s(l)}\right) \right) \Omega\left(Y_{s(k)}\right) \left(\prod_{m=k+1}^{t} \Phi\left(Y_{s(m)}\right) \right) p_{s(k)}^0}, \tag{2}$$

which can be derived by recursive application of Equation 1.

Note that the cell belief of the virtual cell, $p_\emptyset^t$, can also be updated as above, noting that $\Theta_\emptyset\left(Y_{s(t)}\right)$ will always be $\Phi\left(Y_{s(t)}\right)$, since by construction no observation will occur outside the search region, such that

$$p_\emptyset^t = \frac{\Phi\left(Y_{s(t)}\right)\ p_\emptyset^{t-1}}{\Phi\left(Y_{s(t)}\right) + \Omega\left(Y_{s(t)}\right)\ p_{s(t)}^{t-1}}, \tag{3}$$

with the analogous expression for Equation 2 for the virtual cell given by

$$p_\emptyset^t = \frac{\prod_{k=1}^{t} \Phi\left(Y_{s(k)}\right) p_\emptyset^0}{\prod_{k=1}^{t} \Phi\left(Y_{s(k)}\right) + \sum_{k=1}^{t} \left(\prod_{l=1}^{k-1} \Theta_{s(k)}\left(Y_{s(l)}\right) \right) \Omega\left(Y_{s(k)}\right) \left(\prod_{m=k+1}^{t} \Phi\left(Y_{s(m)}\right) \right) p_{s(k)}^0}. \tag{4}$$

Noting that the aggregate belief evolution is given by $B(t) = 1 - p_\emptyset^t$, the significance of Equation 4 is that the evolution of the aggregate belief, for an arbitrary initial belief distribution, can be explicitly examined given the search agent's trajectory and observation history. This expression relates the temporal aspect of the change in belief due to imperfect observations and the spatial search trajectory, highlighting the spatio-temporal nature of search.

A decision regarding the presence or absence of the target in the search region can be made under two conditions. The target is deemed present if the cell belief (and thus the aggregate belief) in any one cell exceeds an upper threshold, $\overline{B}$, where alternatively the target is classified as absent if the aggregate belief falls below a lower threshold, $\underline{B}$. Designation of these two threshold values arises from the particular confidence requirements one has for the search task. For example, setting $\overline{B} = 0.97$ requires that the searcher attain at least 97% certainty that the target is present in a particular cell, and $\underline{B} = 0.05$ implies that a negative decision (i.e., that $H = 0$) is made if there is only 5% probability that the target is present. Figure 2 illustrates a sample time evolution of the aggregate belief with decision boundaries used in the example above. Modulating the decision boundaries clearly affects the time until either decision is made (e.g., decreasing $\overline{B}$ should speed up an affirmative decision). Investigation of a relationship between the design of these thresholds and the expected time till decision is subject of ongoing research.

Some special cases are of particular interest and can be examined directly from the above expressions. In the extremal case of perfect detections, evolution of the aggregate belief remains nontrivial if only all detections up until time t are negative detections. (Otherwise, the affirmative decision would be made upon the first

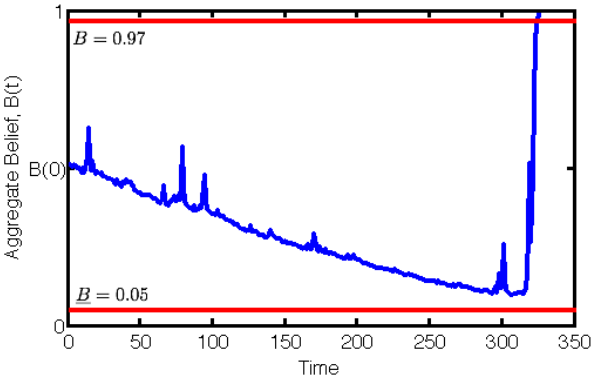


Fig. 2. Sample trace of the evolution of the aggregate belief, starting at $B(0) = \frac{1}{2}$, with upper and lower decision thresholds given by $\overline{B} = 0.97$ and $\underline{B} = 0.05$, respectively. In this case, the target is deemed present in the search region with a decision time of 327 time steps.

positive detection and the search process would terminate.) In this simplified case, Equation 4 reduces to a deterministic expression:

$$p^t_{\emptyset} = \frac{p^0_{\emptyset}}{1 - \sum_{k=1}^{t} p^0_k} \quad \Rightarrow \quad B(t) = 1 - p^t_{\emptyset} = \frac{\sum_{k \in \mathscr{U}(t)} p^0_k}{\sum_{k \in \mathscr{U}(t)} p^0_k + p^0_{\emptyset}},$$

after further assuming a cell is inspected at most once (since no informational benefit is derived from re-visiting a cell due to perfect detections). The corresponding aggregate belief expression, $B(t)$, where all cells not yet inspected at time t are denoted by the set, $\mathscr{U}(t) \subseteq \mathscr{A}$, demonstrates the intuitive result that the belief is simply the ratio between the probability mass still contained in the search region and the total remaining probability mass.

Of greater interest is the expected time until a decision is made, such that analysis for the simplified case of perfect detections can be used to derive a lower bound on the expected decision time. A perfect detector will make an affirmative decision (i.e., $H = 1$) upon inspecting the c^{th} cell if and only if the target is present in cell c, which it is with probability p^0_c. Uncertainty in the decision evolution is introduced only by the randomness of the target location (as there are no errors in observations). Then the expected time till decision, t_D, can be computed as

$$E[t_D] = \sum_{k=1}^{|\mathscr{A}|} t_{s(k)} \cdot p^0_{s(k)} + |\mathscr{A}| \cdot p^0_{\emptyset}, \tag{5}$$

for choice of search agent trajectory, $\mathscr{S}(t) = \{s(1), \ldots, s(|\mathscr{A}|)\}$, where $t_{s(k)}$ is the time when cell $s(k)$ is inspected. The transit time (which is also the search path length for unit speed) for the searcher to reach cell $s(k)$ from $s(k-1)$ is given by $t_{s(k)} - t_{s(k-1)}$. For example, consider a sweeping search (a.k.a. ladder search, in land

SAR) where cells are observed in ascending label order, i.e., $\mathscr{S}(t) = \{1, 2, \ldots, |\mathscr{A}|\}$, which yields an expected decision time of $E[t_D] \approx 77.4$ time steps for the sample belief distribution illustrated in Figure 1(a). Alternatively, suppose the search control strategy employed by the searcher was to inspect the cell with maximal belief at each time step (e.g., no motion constraints), which amounts to a re-indexing of the cells in descending order of cell belief. Again using the example distribution, the search trajectory for such a strategy is given by $\mathscr{S}(t) = \{98, 21, 31, \ldots, 40, 20, 10\}$ (i.e., cells 98 and 10 have maximal and minimal cell beliefs, respectively), such that $E[t_D] \approx 64.3$ time steps for this "snapshot" or "saccadic" [7] approach. These values serve as lower bounds on the expected decision time for their respective strategies, and further analyses of special cases are similarly facilitated by the general expression for the aggregate belief evolution.

While in simple cases the analysis is manageable, in the imperfect sensor case, the additional degrees of freedom of varying detection error probabilities, α and β, render derivation of a closed-form expression for the expected time-till-decision intractable. Instead, stochastic simulation methods may be used to explore the parameter space [7]. Such investigations can provide insight into the constrained searcher problem, despite the fact that determination of the optimal search trajectory, $\mathscr{S}$, is computationally challenging [24].

3 Characterization of Searcher Motion Constraints

One facet of the constrained search problem that remains to be examined is the role of the searcher motion constraints on the performance of the probabilistic search. Classical search theory pertains mostly to the case where motion is restricted to only neighboring cells. However, the framework presented in this paper extends to a larger class of search problems, enabling analysis and comparisons across a greater variety of search agent motion capabilities. For example, an unmanned fixed-wing aircraft with greater maximum speed can reach a greater number of cells in a single time step than a slower unmanned helicopter, but may sacrifice sensing fidelity (i.e., increased detection error probabilities). The influence of searcher motion capabilities on the overall decision performance is the subject of the sequel, and the results therein can assist in determining the most appropriate search platform to best accomplish the search objective.

One efficient strategy can simply be to inspect the cell with maximal belief probability within the set of cells reachable in the next time step. This reachability can formally be studied by examination of the *algebraic connectivity* of the graph [12], which is defined as the second smallest eigenvalue, denoted λ_2, of the graph Laplacian. Some well-known results for λ_2 include the fact that $\lambda_2 = 0 \Leftrightarrow \mathscr{G}_{\mathscr{S}}$ is not connected, and that $\lambda_2 = |\mathscr{A}|$ when $\mathscr{G}_{\mathscr{S}}$ is a complete graph. The graph Laplacian, $\mathscr{L}(\mathscr{G}_{\mathscr{S}})$, is computed by the difference between the degree matrix, $\mathrm{Deg}(\mathscr{G}_{\mathscr{S}})$, and the adjacency matrix, $\mathrm{Adj}(\mathscr{G}_{\mathscr{S}})$, of the search graph (see, e.g., [20]):

$$\mathscr{L}(\mathscr{G}_{\mathscr{S}}) = \mathrm{Deg}(\mathscr{G}_{\mathscr{S}}) - \mathrm{Adj}(\mathscr{G}_{\mathscr{S}}).$$

As mentioned in Section 2.2, the adjacency matrix allows for general representation of arbitrary search environments and situations. Consider now a class of square augmented grid graphs (where diagonal vertices, in addition to horizontal and vertical neighbors, may also be considered adjacent) [14]. Such grid graphs can represent the discretization of rectilinear search regions. The *hop length*, h, is the maximum (integral) distance for which two vertices on the graph are directly connected, i.e., there exists an edge in $\mathscr{G}_{\mathscr{S}}$ between vertices k and l if cell l is within h cells of cell k. For example, the augmented grid graph illustrated in Figure 1(b) has hop length of one, which corresponds to the constrained search motion limited to neighboring cell search. Note that a hop length of $h = \sqrt{|\mathscr{A}|} - 1$ is sufficient to make $\mathscr{G}_{\mathscr{S}}$ a complete graph. This latter case reflects the situation where the search agent has the ability to "jump" to any cell from any other cell in the search region in a single time step, e.g., inspection by remote camera on a sensor network.

The relationship between hop length and the algebraic connectivity of the augmented grid graph is depicted in Figure 3. This plot demonstrates the characterization of search motion constraints by quantitative spectral analysis of the search graph, $\mathscr{G}_{\mathscr{S}}$, and is consistent with the known result that λ_2 is a non-decreasing function for graphs with the same vertex set $\mathscr{V}$ [12]. While this relationship is illustrated only for the class of augmented grid graphs, in principle, the connectivity of the search graph ranges from minimally connected to complete in a more general fashion. The framework presented in this paper also applies to arbitrary connected graphs (for which λ_2 lies on the curve illustrated in Figure 3) and points towards addressing directional motion limitations using directed edges (e.g., one-way streets in a road network or nonholonomic constraints in motion capabilities).

Optimization of the search for minimizing the time till detection is provably NP-complete [24], and as such, determination of an optimal search trajectory is intractable. Although existing approximation methods, such as branch and bound techniques [11] for partially-observable Markov decision processes, enable study

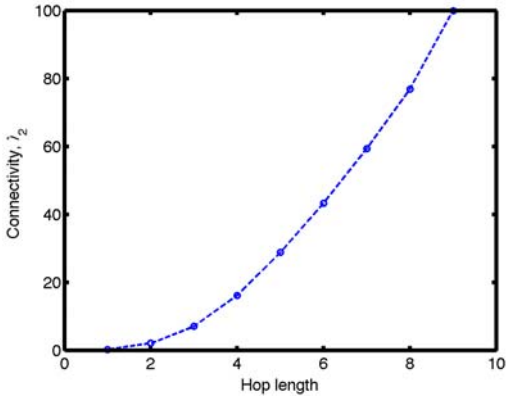


Fig. 3. Relationship between hop length, h, and connectivity, λ_2, of the corresponding (augmented grid) search graph for $|\mathscr{A}| = 100$ (i.e., 10×10 grid).

of near-optimal solutions, the search problem here is further complicated by the possibility of false positive detections, which gives rise to a non-monotonic aggregate belief evolution. Furthermore, in the context of *active search*, where search and control are bound together by feedback, a search strategy which is both computationally efficient and effective (at the expense of optimality) is essential for practical implementations.

Consider the search strategy described in Section 2.3 of seeking and observing the maximal cell belief location, now subject to the searcher's motion constraints. Such a strategy is motivated by the fact that the incremental change (in absolute value) in aggregate belief is monotonically increasing for increasing cell belief for the observed cell, which can easily be shown by computing the one-step change in belief found by Equation 3. This observation agrees with known results, such as for unconstrained search [24], or for special conditions of concavity and continuity of detection functions [23].

Once this maximal cell has been identified, the task of planning and transiting from the search agent's current location, $s(t)$, to the goal cell, $c^\star$, can be constructed as a shortest path problem on the search graph, $\mathscr{G}_{\mathscr{S}}$, for which efficient algorithmic solution methods can be utilized, such as Dijkstra's method. The search agent must transit this shortest path of length T, for which each cell visited (but uninspected) en route to the maximal belief cell represents unit cost (i.e., one time step). Bayesian update of the belief distribution occurs upon the searcher's arrival and inspection of cell $s(t+T) = c^\star$. Note that it is assumed in this strategy that observations are not made while in transit to the maximal belief cell. This type of limitation may represent sensing modalities which require the search agent to be (nearly) stationary during the observation (e.g., image stabilization). Alternatively, resource limitations (e.g., power) or risk minimization (e.g., avoiding target lock) may require that the detector be only turned on in specific instances. Noting that the effect of additional information gained from taking observations during transit leads to reduced decision times [7], the role of the initial belief distribution is more prominent as average path lengths become a contributing factor to the decision performance. The study of these factors for the case of "streaming" observations, rather than the "snapshot" approach examined in this paper, is subject of ongoing research.

As done in the previous section, the special case of zero detection errors can once again be used to derive both intuition as well as analytic results on the decision performance (i.e., lower bounds on the search decision time) for the given search strategy. Using the example initial belief distribution, the trajectory of the search agent can easily be determined as a function of the algebraic connectivity (parameterized by the hop length) of the search graph. Figure 4 illustrates the computed expected decision time as determined by Equation 5, where $t_{s(k)}$ includes the time to transit the path as calculated by the shortest path algorithm of the given search graph.

The expected decision time is reduced by increasing algebraic connectivity of the search graph. Also shown in Figure 4 are the expected value traces for decision times for when the target is determined present in the search region and when it is not. The horizontal line corresponds to the decision time for the nonadaptive

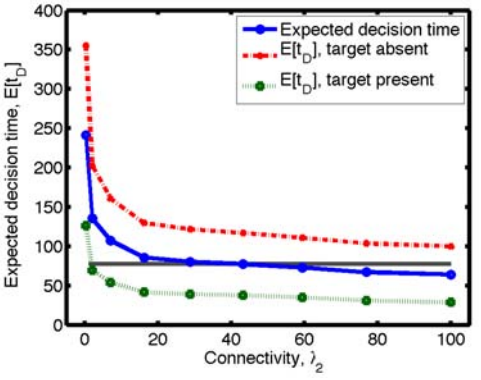


Fig. 4. Expected decision times for the perfect detector (i.e., $\alpha = \beta = 0$). Bounding traces represent the expected decision time when the target is determined absent (upper) or present (lower) in the search region. The intersecting plot (straight line) represents the expected decision time if employing the nonadaptive sweeping search strategy with zero detection errors.

sweeping search. The plot shows that for hop length $h \leq 5$, the naïve sweeping search may perform, on average, better than the "snapshot" search method, due to transit time penalties for longer average path lengths. The practical interpretation of this insight is that when employing a slow-moving searcher (e.g., ground search parties), a simple sweeping search is more efficient (on average) than a more targeted approach to searching cells, whereas a faster unmanned aerial vehicle can take advantage of its speed to utilize the latter search strategy.

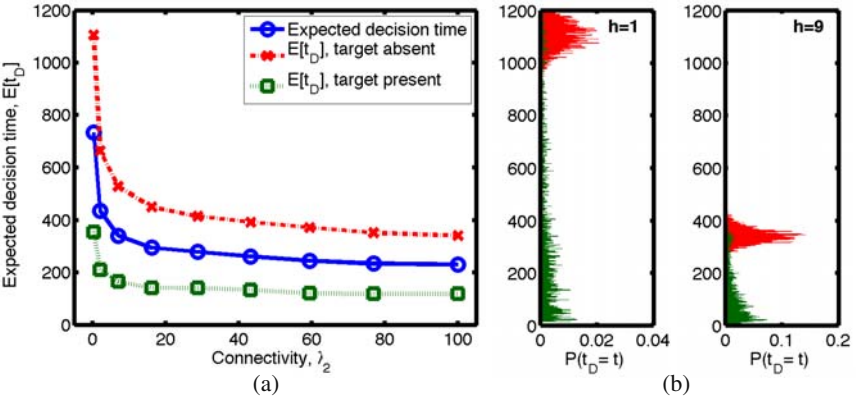


Fig. 5. Expected decision times for the imperfect detector, with false positive and false negative error probabilities $\alpha = 0.25$ and $\beta = 0.20$, resp. (a) Upper and lower bounding traces represent expected decision times for when the target is absent and present. (b) Decision time probability histograms over 5000 simulated trials, shown for hop lengths $h = 1$ and $h = 9$. The different peaks (corresponding trials are differentiated by color) in each histogram corresponds to when the target was present (lower) or absent (upper).

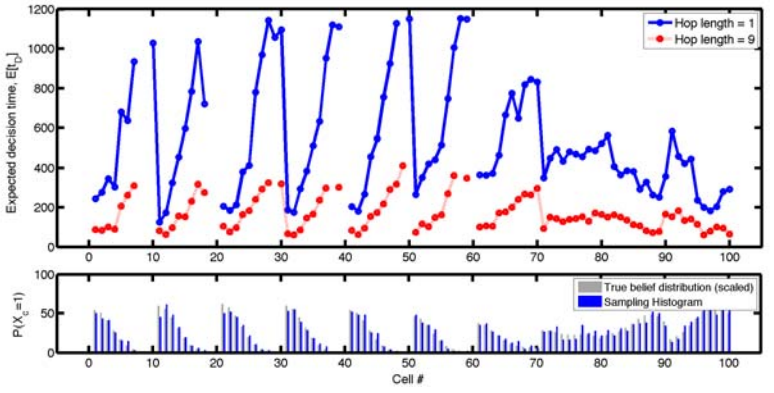


Fig. 6. Decision time as a function of the initial belief distribution. The top figure illustrates a comparison of two hop lengths ($h = 1$ and $h = 9$) over repeated simulation trials. The bottom plot shows the sampling histogram (compared against the true initial belief distribution) of the true target location for the simulated trials.

In the more interesting case of imperfect detections, analogous insight into the relationship between time till decision and algebraic connectivity of the search graph can be extracted from statistical studies. Figure 5 illustrates the decision time results for increasing algebraic connectivity over $N = 5000$ simulation trials, with the search agent initially located in cell $s(0) = 1$. The initial target location is randomly drawn according to the initial belief distribution, which includes the possibility (with probability $p_{\emptyset}^0 = \frac{1}{2}$) that the target is not present in the search region. Simulated detections are subject to false positive and false negative errors of $\alpha = 0.25$ and $\beta = 0.20$, respectively.

As expected, the algebraic connectivity of the search graph plays a definite role in the determination of the search decision time, in that greater algebraic connectivity leads to better decision performance (i.e., reduced time till decision). Further, errors in detection observations can only increase the decision time, as the decision time is influenced both by uncertainty in detections and uncertainty in target location. Figure 5(a) shows the expected decision time traces, including upper and lower expected value bounds corresponding to the cases where the target is absent and present, respectively. One can clearly observe that the greatest improvement occurs with only mild increases in searcher mobility with only diminishing additional benefit thereafter. For example, doubling the maximum speed of the UAV searcher from the nominal value (i.e., from one hop to two) decreases the decision time by 40.5% but doubling it again (to $h = 4$) only yields half the improvement (59.6% overall reduction). This information can be used to balance the advantage of employing a faster search platform with other deciding factors, e.g., higher expense. Figure 5(b) shows the decision time probability histograms of 5000 trials for the extremal hop lengths of $h = 1$ and $h = 9$. The bi-modal nature of these histograms highlights the distribution of the two possible outcomes of the search decision, i.e., $H = 0$ or $H = 1$.

As mentioned previously, the uncertainty in the target's true location influences the expected time till decision. The target location for each simulation trial was drawn according to the example initial belief distribution. Figure 6 illustrates the relevance of the initial cell beliefs on the decision time, shown for hop lengths $h = 1$ and $h = 9$ (see upper plot). The lower plot depicts the probability histogram as a function of cell number, i.e., the probability that the target is located in cell c, $P(X_c = 1)$. As can be seen, the sampling of target locations over all trials is consistent with the underlying initial belief distribution. The decision time is inversely related to the initial cell belief. In other words, if the target is located in less likely locations, the time necessary to find the target is greater. This effect is mitigated by increasing the connectivity of the search graph. It merits noting that even if the search agent is misinformed (i.e., the initial belief distribution is inaccurate), the searcher will still successfully determine the presence (and location) or absence of the target within the confidence levels prescribed by the upper and lower decision thresholds.

4 Conclusions and Future Work

This paper presented a framework for conducting and analyzing the probabilistic search of a stationary target in a discretized search region. The decision-theoretic approach of representing the search task as a sequential binary hypothesis test enabled the derivation of a recursive expression for the time evolution of the search decision as a function of imperfect detections. This analysis facilitated the use of the expected time till decision as a metric of decision performance. Further, the use of the *search graph* was introduced as a general representation of the searcher's constrained motion, due to dynamic or environmental constraints. A distinct relationship between the algebraic connectivity, given by the second smallest eigenvalue of the search graph's Laplacian matrix, and the expected decision time was demonstrated by statistical studies. Also, the dependence on searcher motion constraints can be utilized, along with specification of detection errors and decision thresholds, to optimize the construction of the search problem, e.g., choose better spatial decomposition of the search environment or improve the motion capabilities of the search agent, to reduce the search decision time.

Active decision making in autonomous systems research continues to maintain significant interest, and there exist many avenues of future research based on the foundation proposed in this work. The role of target motion in the context of search theory has largely only been studied in the limited cases of independence of cells and without false positive detections, for both nonreactive [5, 26] and adversarial [13, 21] targets. However, incorporation of the decision framework may address the aforementioned shortcomings, as well as examine the case when the target can move in and out of the search region. Application of the discrete Chapman-Kolmogorov equation can capture probabilistic target motion models as an extension of the proposed Bayesian filtering framework.

Further, search using multiple agents (see, e.g., [7, 22, 10]) highlights the additional challenges of communication, coordination, optimization, and data fusion for

distributed systems. With increasingly greater emphasis on distributed architectures of complex systems, research in such systems continues to be an active area. Also, while a single target scenario is pertinent in many realistic situations, the class of problems becomes richer in the case of multiple objects. The challenge (and thus its appeal) increases especially in the case where objects must be identified in addition to being located, combining classification with the search task [15]. Applications of an integrated search and identification framework include tasks such as person-tracking in crowded environments or assessment of possible threats in a scenario. The sequential binary hypothesis framework may be extended by using composite hypotheses to address these multivariable decisions.

References

1. Alterovitz, R., Simeon, T., Goldberg, K.: The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty. In: Proceedings of Robotics: Science and Systems, Atlanta, GA, USA (June 2007)
2. Benkoski, S.J., Monticino, M.G., Weisinger, J.R.: A Survey of the Search Theory Literature. Naval Research Logistics 38, 469–494 (1991)
3. Bourgault, F., Furukawa, T., Durrant-Whyte, H.: Optimal Search for a Lost Target in a Bayesian World. Field and Service Robotics (STAR Springer Tracts in Advanced Robotics) 24, 209–222 (2006)
4. Bourgault, F., Makarenko, A., Williams, S., Grocholsky, B., Durrant-Whyte, H.: Information Based Adaptive Robotic Exploration. In: Proc. of 2002 IEEE Int'l. Conf. on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September-5 October, vol. 1, pp. 540–545 (2002)
5. Brown, S.S.: Optimal Search for a Moving Target in Discrete Time and Space. Operations Research 28(6), 1275–1289 (1980)
6. Choset, H.: Coverage for Robotics - A Survey of Recent Results. Annals of Mathematics and Artificial Intelligence 31, 113–126 (2001)
7. Chung, T.H., Burdick, J.W.: A Decision-Making Framework for Control Strategies in Probabilistic Search. In: Proc. 2007 IEEE Int'l. Conf. on Robotics and Automation (2007)
8. Chung, T.H., Burdick, J.W.: Multi-agent Probabilistic Search in a Sequential Decision-theoretic Framework. In: Proc. 2008 IEEE Int'l. Conf. on Robotics and Automation (2008)
9. DasGupta, B., Hespanha, J.P., Riehlb, J., Sontag, E.: Honey-pot Constrained Searching with Local Sensory Information. Nonlinear Analysis 65(9), 1773–1793 (2006)
10. Dell, R.F., Eagle, J.N., Martins, G.H.A., Santos, A.G.: Using Multiple Searchers in Constrained-Path, Moving-Target Search Problems. Naval Research Logistics 43(4), 463–480 (1996)
11. Eagle, J.N., Yee, J.R.: An Optimal Branch-and-Bound Procedure for the Constrained Path, Moving Target Search Problem. Operations Research 38(1), 110–114 (1990)
12. Fiedler, M.: Algebraic Connectivity of Graphs. Czechoslovak Mathematical Journal 23(98), 298–305 (1973)
13. Gal, S.: Search Games. Academic Press, New York (1980)
14. Hamburger, P., Vandell, R., Walsh, M.: Routing Sets in the Integer Lattice. Discrete Applied Mathematics 155(11), 1384–1394 (2007)

15. Kalbaugh, D.V.: Optimal Search Among False Contacts. SIAM Journal on Applied Mathematics 52(6), 1722–1750 (1992)
16. Koopman, B.O.: Search and Its Optimization. The American Mathematical Monthly 86(7), 527–540 (1979)
17. Kress, M., Szechtman, R., Jones, J.S.: Efficient Employment of Non-Reactive Sensors. Military Operations Research (2008) (to appear)
18. Lau, H.: Optimal Search in Structured Environments. PhD thesis, University of Technology, Sydney, Australia (2007)
19. LaValle, S.M., Hinrichsen, J.: Visibility-Based Pursuit-Evasion: The Case of Curved Environments. IEEE Transactions on Robotics and Automation 17(2), 196–201 (2001)
20. Merris, R.: Laplacian Matrices of Graphs: A Survey. Linear Algebra and its Applications 197/198, 143–176 (1994)
21. Murrieta-Cid, R., Muppirala, T., Sarmiento, A., Bhattacharya, S., Hutchinson, S.: Surveillance Strategies for a Pursuer with Finite Sensor Range. Int'l Journal of Robotics Research 26(3), 233 (2007)
22. Song, N.-O., Teneketzis, D.: Discrete Search with Multiple Sensors. Mathematical Methods of Operations Research 60(1), 1–13 (2004)
23. Stone, L.D.: Theory of Optimal Search, 2nd edn. Academic Press, London (1989)
24. Trummel, K.E., Weisinger, J.R.: The Complexity of the Optimal Searcher Path Problem. Operations Research 34(2), 324–327 (1986)
25. Vidal, R., Shakernia, O., Kim, H., Shim, D., Sastry, S.: Probabilistic Pursuit-Evasion Games: Theory, Implementation and Experimental Evaluation. IEEE Trans. on Robotics and Automation 18(5), 662–669 (2002)
26. Washburn, A.R.: Search for a Moving Target: The FAB Algorithm. Operations Research 31(4), 739–751 (1983)
27. Washburn, A.R.: Search and Detection, 4th edn. Topics in Operations Research Series. INFORMS (2002)

Planning with Reachable Distances

Xinyu Tang, Shawna Thomas, and Nancy M. Amato

Abstract. Motion planning for spatially constrained robots is difficult due to additional constraints placed on the robot, such as closure constraints for closed chains or requirements on end effector placement for articulated linkages. It is usually computationally too expensive to apply sampling-based planners to these problems since it is difficult to generate valid configurations. We overcome this challenge by redefining the robot's degrees of freedom and constraints into a new set of parameters, called *reachable distance* space (RD-space), in which all configurations lie in the set of constraint-satisfying subspaces. This enables us to directly sample the constrained subspaces with complexity linear in the robot's number of degrees of freedom. In addition to supporting efficient sampling, we show that the RD-space formulation naturally supports planning, and in particular, we design a local planner suitable for use by sampling-based planners. We demonstrate the effectiveness and efficiency of our approach for several systems including closed chain planning with multiple loops, restricted end effector sampling, and on-line planning for drawing/sculpting. We can sample single-loop closed chain systems with 1000 links in time comparable to open chain sampling, and we can generate samples for 1000-link multi-loop systems of varying topology in less than a second.

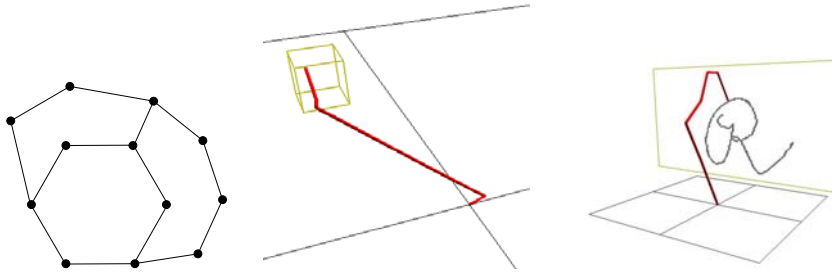
1 Introduction

Spatially constrained systems are systems with constraints such as requiring certain parts of the system to maintain contact or to maintain a particular clearance from

Xinyu Tang
Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043 USA
e-mail: xtang@google.com

Shawna Thomas and Nancy M. Amato
Department of Computer Science and Engineering, Texas A&M University,
College Station, TX 77843 USA
e-mail: sthomas@cse.tamu.edu, amato@tamu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 517–531.
springerlink.com



(a) (b) (c)

Fig. 1. Examples of spatially constrained systems. Each must satisfy certain closure constraints. (a) Multiple loops: all loops must remain closed. (b) End-effector restrictions: the end effector must remain within the specified boundary (in wireframe). (c) Drawing/sculpting: the end effector must follow a desired drawing trajectory while remaining in contact with the canvas.

each other. They have many applications in robotics and beyond, such as parallel robots [18], grasping for single and multiple robots [13], reconfigurable robots [14, 20], closed molecular chains [24], and computer animation [11]. Figure 1 gives some examples of spatially constrained systems.

Motion planning for these systems is particularly challenging due to the additional constraints placed on the system. Since the complexity of deterministic algorithms [23, 15] is exponential in the number of degrees of freedom (DOF) of the robot, they are impractical for most systems of practical interest. While sampling-based planning methods are successful for many high DOF systems, their performance degrades for spatially constrained systems in which the set of valid configurations occupies a small volume of configuration space (the set of all configurations, valid or not). In these situations, the probability of randomly sampling a configuration that also satisfies the spatial constraints approaches zero (e.g., closed chain systems [16]).

In this paper, we generalize the reachable distance representation we proposed for single-loop closed chains [25] to handle other types of spatial constraints in addition to closure constraints and to satisfy multiple constraints simultaneously. Our framework handles a wide range of systems including both 2D and 3D chains, single and multiple loops, end effector placement/trajectory requirements, and prismatic joints. This new representation for spatially constrained systems, called *reachable distance space* (RD-space), enables more efficient sampling of valid configurations. It is defined by a set of reachable distances for the robot instead of, e.g., by joint angles, as in configuration space. Instead of sampling in the configuration space, we sample in the RD-space.

We describe our new representation, RD-space, in Section 3 and give a recursive sampling algorithm with complexity linear in the system's DOF in Section 4. We also describe a RD-space local planning method suitable for sampling-based planners. Then, we present applications of our approach for multiple-loop closed chain planning (Section 5), restricted end effector sampling (Section 6), and on-line

planning for drawing/sculpting (Section 7). We demonstrate that our method is scalable to thousands of DOF and show that it outperforms other randomized sampling methods and other specialized methods for closed chains.

2 Related Work

In theory, exact motion planning algorithms can handle systems with spatial constraints by explicitly computing the constraint-satisfying subspaces in configuration space (i.e., the set of all robot configurations, valid or not) [23, 15]. However, since these algorithms are exponential in the dimension of configuration space, they are generally impractical for systems with more than 4 or 5 DOF.

Randomized algorithms such as Probabilistic Roadmap Methods (PRMs) [12] and Rapidly-exploring Randomized Trees (RRTs) [17] are widely used today for many applications. These methods randomly sample the configuration space, retaining valid configurations, and connect these samples together to form a roadmap (i.e., a graph or tree) that represents the connectivity of the valid configuration space. While successful for many high DOF systems, their performance degrades for spatially constrained systems as the probability of randomly sampling a configuration satisfying the constraints approaches zero (see, e.g., [16]).

Much work has been done to optimize these sampling-based planners for closed chain systems. Closed chain systems are a special type of spatially constrained system where the linkage must satisfy the closure constraints at all times during planning. In the first sampling-based method for closed chains, Kavraki et al. [16, 30] randomly sample configurations and then apply an iterative random gradient descent method to push the configuration to the constraint surface. They solved planar closed chains with up to 8 links and 2 loops in several hours with PRM [16] and several minutes with RRT [30]. Kinematics-based PRM (KBPRM) [6] first builds a roadmap ignoring all obstacles, then populates the environment with copies of this kinematics roadmap, removes invalid portions, and connects copies of configurations with a rigid-body local planner. This method can solve closed chain problems with 7–9 links in under a minute. KBPRM has been extended to better handle larger linkages [29], reduce running time [3, 2], and deal with multiple loops [1]. PRM-MC combines PRMs and Monte Carlo simulations to guarantee closure constraints [5]. It can generate 100 samples of a 100-link closed chain and of a 2-loop system containing 16 links in under a minute. Trinkle and Milgram proposed a path planning algorithm [26] based on configuration space analysis [19] that does not consider self-collision. Han et al. [10] proposed a set of geometric parameters for closed chain systems such that the problem can be reformulated as a system of linear inequalities. They extend this work to handle multiple loops [9]. They show results for a 1000-link closed chain and a 1000-loop system containing 3000 links. However, they do not discuss the algorithm's complexity or provide an experimental performance study. Their work is similar to ours in that both methods reframe the original joint-based problem into another set of parameters where satisfying a set of spatial constraints is easier. However, our work proposes a different set of parameters

that enables efficient sampling of configurations and naturally results in a simple local planner to connect configurations, the two primitive operations necessary for sampling-based planning.

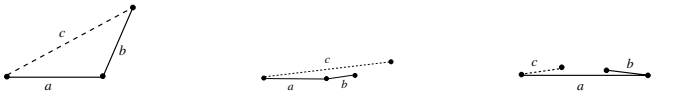
In addition to closed chain systems, there has been work on other types of spatially constrained systems. Garber and Lin [4] use constrained dynamics to plan motions to ensure constraints such as joint connectivity, the spatial relationship between multiple robots, or obstacle avoidance. They show results for a 6 DOF robot arm. Oriolo et al. plan articulated motions where the end effector must travel along a given trajectory for fixed-base manipulators [22] and mobile manipulators [21]. They extend probabilistic planning methods for this system and present results for up to a 6 DOF robot arm on a mobile planar base. Yao and Gupta [31] propose two approaches, Alternate Task-space and Configuration-space Exploration (ATACE) and randomized gradient descent, to plan paths for manipulators with general end effector constraints; results are presented for up to a 9 DOF robot. Han et al. [8] unify their work on serial chains [7] and closed chains [10] to provide inverse kinematic solutions that satisfy 3D, 5D, and 6D end effector placement constraints. They can sample a single configuration for a 1000-link arm in 19 ms.

3 Reachable Distance Framework

The reachable distance data structure (or RD-tree) is a hierarchy of reachable distances defined by recursively partitioning the original system into smaller subsystems. This is a generalized version of the data structure developed in our previous work for single closed loop systems [25]. The main advantage of this representation over the traditional joint angle representation is that it encodes spatial constraint information. For a given set of constraints, this new representation helps us quickly determine whether the robot is able to satisfy those constraints, and if so, generate configurations that satisfy them.

We begin with a simple example illustrating the main ideas of our approach. Figure 2 presents a simple robot with two variable length links, a and b. Suppose we are given a spatial constraint where the distance between the base and the end effector needs to be c. To satisfy this constraint, the length of each link has to be in an appropriate range to satisfy the triangle inequality: $|a-b| \leq |c| \leq |a+b|$. We call this range the *available reachable range* (ARR), i.e., the set of distances/lengths which allow the spatial constraints involved in the sub-chain to be satisfied. In this case, the ARR of link a, for example, can be calculated from the constraint c and the *reachable range* of the other link b. We can first sample a length for link a and update the ARR of the other link b, and then sample b. Once we find valid values of a and b, then a, b and c form a triangle and we can calculate the joint angle between link a and link b.

In the following we show how to extend this sampling strategy to handle more general spatially constrained systems. We note that this scheme only ensures that the spatial constraints are satisfied — collision checking is still required. To simplify the exposition, here we consider articulated robots where the distance between



(a) a and b are of appropriate length to satisfy constraint c.

(b) a and b are too short.

(c) a is too long relative to b.

Fig. 2. Three configurations of an articulated system composed of 2 variable length links a and b where the distance between the base and the end effector needs to satisfy spatial constraint c.

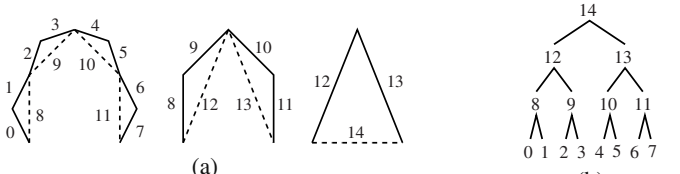


Fig. 3. (a) Articulated linkage with 8 links (labeled 0–7). Two consecutive links form a parent vlink (dashed lines). This repeats to form a hierarchy (b) where the parents in one level become the children in the next level.

the base and the end effector must be in a certain range. However, our reachable distance representation is general and may be applied to other systems and other spatial constraints including 2D and 3D chains, single and multiple loops, end effector requirements, and prismatic joints.

Articulated Linkage as a Hierarchy of Virtual Links. In an articulated linkage, two (or more) consecutive links (called *children*) form a *virtual link* (called the *parent*). The virtual link, or *vlink*, and its children compose a *sub-chain*. For example, in Figure 3(a), vlink 8 is the parent of the two actual links 0 and 1, while vlink 12 is the parent of vlinks 8 and 9. We iteratively build higher level vlinks until we obtain a single vlink (14) at the highest level, see Figure 3(b).

Reachable Range of a Virtual Link. For a particular configuration of a vlink, the *reachable distance* of a vlink (sub-chain) is the distance between the endpoints of the sub-chain it represents. It has different values for different configurations. We call the range of those different values the *reachable range* (RR). For example, the RR of the "root" vlink is the RR of the entire chain, while the RR of an actual link is simply the range of its length.

The RR of a parent can be calculated from the RRs of its children. If we always build a vlink using 0 or 2 children, RRs are computed as follows. Consider a vlink with no children. It has only 1 link. Let l_{min} and l_{max} be the minimum and maximum allowable values of the link's length, respectively. (If the link is not prismatic, $l_{min} = l_{max}$.) Its RR is $[l_{min}, l_{max}]$.

Now consider the vlink l with 2 children a and b in Figure 4(a). They form a triangle. Let $[a_{min}, a_{max}]$ be the RR of the first child and $[b_{min}, b_{max}]$ be the RR of the second child. (If a or b is an actual link, its RR is defined by the problem,

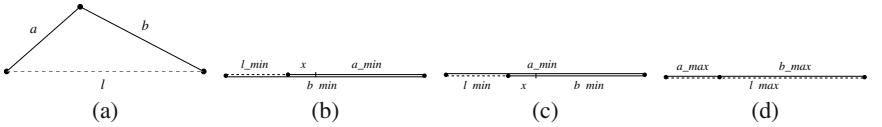


Fig. 4. (a) A sub-chain with 2 children a and b and its vlink l. (b) A configuration where l is minimum when $a_{min} < b_{min}$. $x = b_{min} - a_{min}$ if $a_{max} > b_{min}$ and $x = a_{max} - a_{min}$ otherwise. (c) A configuration where l is minimum when $a_{min} > b_{min}$. $x = a_{min} - b_{min}$ if $b_{max} > a_{min}$ and $x = b_{max} - b_{min}$ otherwise. (d) A configuration where l is maximum.

otherwise it is a function of the RRs of its children. RRs are defined from the bottom up, recursively.) The RR of the parent is then $[l_{min}, l_{max}]$ where

$$l_{min} = \begin{cases} max(0, b_{min} - a_{max}), & a_{min} < b_{min} \\ 0, & a_{min} = b_{min} \\ max(0, a_{min} - b_{max}), & a_{min} > b_{min} \end{cases} \quad (1)$$

and

$$l_{max} = a_{max} + b_{max}. \quad (2)$$

Note that these are not limited to computing parent link RRs. Given the RRs of any two links in the same triangular sub-chain, these equations calculate the RR of the remaining link to satisfy the triangle inequality.

Available Reachable Range of a Virtual Link. The ARR of a vlink is the set of distances/lengths which allow it to satisfy the spatial constraints when only considering the other links in the same sub-chain. The ARR is a subset of the RR and is a function of the ARRs of the other links in the same sub-chain loop. Changes in the ARR of one link cause changes in the ARRs of the other links in the sub-chain.

Before sampling begins (i.e., no spatial constraint is imposed on the robot), the ARR of each link is equal to its RR. However, as we sample and enforce spatial constraints by fixing the length of a link, portions of the RRs of the other links in the same sub-chain may no longer be available. When this happens, we update the ARRs in the other links in the sub-chain using Equations 1 and 2. So, given a specified value of one link we can find the ARR of the other two in the sub-chain and sample available lengths for them. The sampling cost is discussed below in Section 4.1.

Now we have a set of lengths, one for each vlink, for a configuration that satisfies the spatial constraints. We can then compute the joint angles between vlinks using basic trigonometry functions. We will illustrate reachable distance sampling through example applications in more detail in the following sections.

4 Primitives for Sampling-Based Planning

Here we describe two primitive operations that suffice to implement most sampling-based motion planners: sampling (i.e., generating valid configurations) and local planning (i.e., finding a valid path between two samples).

4.1 *Sampling with the* RD*-tree*

Sampling in the RD-space involves the following steps:

1. Recursively sample vlink lengths from their ARR.
2. Sample the orientation of each sub-chain.
3. Compute appropriate joint angles from the vlink lengths and orientations.

Note that this sampling does not necessarily determine validity, i.e., it will still need to be checked for (self) collision.

This sampling scheme samples the RD-space uniformly at random. While every point in RD-space satisfies the spatial constraints, a set of samples in RD-space may not have the same distribution in the subset of joint space that also satisfies the constraints. In other words, a sampling method with a uniform distribution in RD-space does not guarantee a uniform distribution in the spatial constraint-satisfying subset of joint space. We describe each step in more detail below.

1) Recursively sample link lengths. Recall that we define the ARR as the subset of the RR that satisfies the spatial constraints with respect to the rest of the sub-chain. Once we fix the length of an ARR in a sub-chain, the other ARRs may be restricted. Thus, we can generate a configuration by sampling reachable distances and updating ARRs starting at the the root of the RD-tree and recursing until all sub-chain reachable distances are sampled. Note that by sampling in this way, an ARR may never become empty; in its most constrained case, its minimum and maximum may become equal.

The sampling algorithm is called once for each vlink, represented by a single internal node in the hierarchy. Because the hierarchy is constructed as a binary tree, there are $O(n)$ internal nodes, where n is the number of actual links in the chain. Thus, the sampling algorithm requires $O(n)$ time to generate a configuration.

2) Sample link orientations. Each sub-chain forms a triangle with multiple configurations with the same vlink length: two in 2D (i.e., concave and convex, see Figure 5(a)) and many in 3D depending on the dihedral angle between its triangle and its parent's (see Figure 5(b)). Thus, we also sample the vlink's orientation.

3) Calculate joint angles. Consider the joint angle θ between links a and b. Links a and b are connected to a vlink c to form a triangle. Let l_a, l_b, and l_c be the lengths of links a, b, and c, respectively. The joint angle can be computed using the law of cosines: $\theta = \arccos((l_a^2 + l_b^2 - l_c^2)/(2l_a l_b))$.

(a) (b)

Fig. 5. (a) In 2D, the same vlink represents two configurations: a concave and convex triangle. (b) In 3D, the same vlink represents many configurations with different dihedral angles ρ.

4.2 *Local Planning in* RD*-space*

We propose a local planner for spatially-constrained systems. Given two configurations, q_s and q_g, a local planner attempts to find a sequence $\{q_s, q_1, q_2, \ldots, q_g\}$ of valid configurations to transform q_s into q_g at some user-defined resolution. We describe a simple local planner that uses a straight-line interpolation in RD-space.

To generate the sequence of intermediate configurations, this local planner interpolates the lengths of each vlink between q_s and q_g. We then determine the vlink's orientation sequence. In 2D, if the orientation is the same in q_s and q_g, we keep it constant. Otherwise, we must "flip" the links as described below. In 3D, we simply interpolate between the dihedral angles in q_s and q_g. In addition to checking collision as with other local planners, we determine the validity of the sequence by checking that each vlink's length is in its ARR.

Concave/convex flipping for 2D chains. Consider the two configurations in Figure 5(a) with different orientations. To flip the vlink, we need to expand (or open) the parent's vlink enough so its children can change orientation while remaining in their ARR. At some point, the parent's ARR must be large enough to accommodate the summation of its children's reachable distances (i.e., allow the children to be "flat"). Such a constraint can be easily handled by first recalculating the ARR for this minimum "flat" constraint. If available, we sample a configuration where the vlinks are "flat" and try connecting it to both the start and goal as above.

5 Application: Closed Chain Planning with Multiple Loops

We first describe how our representation of reachable distances can be used to ensure the closure constraint and how to handle simultaneous constraints such as with multiple loops. In our preliminary work [25], we demonstrated how this method efficiently samples closed configurations for single-loop closed chains with thousands of links in just seconds and can be used in a complete motion planning framework. Here we show results for a multiple loop system. All experiments were performed on a 3GHz desktop computer and were compiled using gcc4 under linux. Our current implementation supports planar joints and spherical joints.

Enforcing the closure constraint. A closure constraint can be considered as a special type of spatial constraint where the end effector (the last link) is always connected to the base (the first link). Thus, we simply require that the distance between the first link and the last link is zero by making the root vlink length zero.

Handling multiple loops. For a closed chain system with more than one loop, we construct a RD-tree for each loop. Then the whole system can be represented by a set of RD-trees with some sub-trees in common. The common sub-chain between two loops now becomes a shared node in both trees. To make both loops closed, the common edge should satisfy the closure constraints in both trees, i.e., the ARR of a common node should be the intersection of the ARRs of the same vlink on both trees. Loops must be ordered such that each loop only has junctions with its predecessors (e.g., from an ear decomposition [27, 28]).

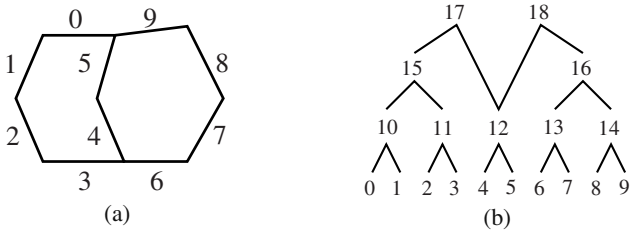


Fig. 6. (a) Multiple loop system and (b) corresponding set of RD-trees.

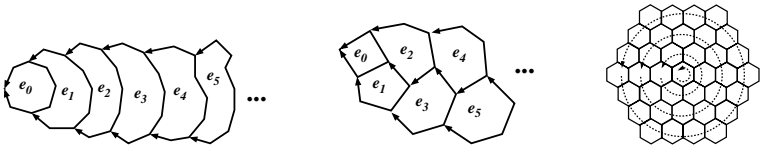
Figure 6 shows an example of a closed chain system with two loops (a) and the corresponding set of RD-trees (b). Two trees rooted at nodes 17 and 18 correspond to the two loops in this system. Both share the same node 12. When we update the ARR of 12, we should consider the ARR of both 15 and 16. Given such a set of RD-trees, we can perform the sampling and planning on each loop sequentially. As for a single loop, we set the length of each root link to be zero and then sample the lengths of other links.

Figure 7 shows an example of a 3-loop system with 14 links. Given only the (a) start configuration and the (e) goal configuration, our method took only 0.01 seconds to find this path containing 40 intermediate configurations.

Fig. 7. A path for a three-loop closed chain system from (a) to (e).

To demonstrate the efficiency and scalability of this approach, we study the time required to generate 1000 samples for multiple loop systems. In previous work [25], we performed a similar study with single loop closed chain systems. There we compared to the original KBPRM [6] and to the extension for larger linkages [29] which was already demonstrated to perform better than existing closed chain methods such as Randomized Gradient Descent [16, 30]. Our method was able to scale to thousands of links while the KBPRM extension, the closest competitor, was only able to generate samples for loops up to 200 links in the 12 hour time limit. We were unable to compare to [7, 10] because we do not have access to their Matlab implementation. They report that they can generate a single closed conformation for a 1000-link chain in 19 ms on a "desktop PC" (processor speed not reported).

Here we look at three different multiple loop topologies, see Figure 8. The multiple loop system can be partitioned into ears (indicated by the arrows). Topology 1 arranges the ears such that the endpoints of ear e_i connect to ear e_{i-1}. Topology 2 arranges the ears such that the endpoints of ear e_i connect to ears e_{i-1} and e_{i-2}. Topology 3 arranges the ears in a "honeycomb" pattern. For topologies 1 and 2, we vary the number of loops in the system. For topology 3, we vary the number of



(a) Topology 1 (b) Topology 2 (c) Topology 3

Fig. 8. Multiple loop topologies studied. (a) Example with 8-link ears (indicated with arrows). The ears are arranged such that the endpoints of ear e_i connect to ear e_{i-1}. (b) Example with 4-link ears (indicated with arrows). The ears are arranged such that the endpoints of ear e_i connect to ears e_{i-1} and e_{i-2}. (c) "Honeycomb" example with 4 rings (indicated with dashed arrows).

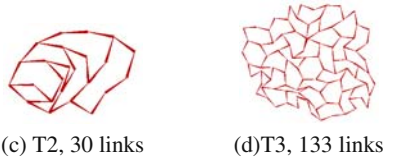
rings in the honeycomb pattern from 1 to 4. Figure 9 summarizes the results, averaged over 10 runs. Even with 1024 links and 256 loops in topologies 1 and 2, our method takes under 1.5 minutes to generate 100 samples. Figure 9(b) shows that our method performance is only somewhat affected by system topology when collision is ignored. However, different topologies require different numbers of attempts to generate collision-free configurations because they place the links in different proximities to each other. For example, topology 2 requires many more attempts with only 30 links because the links in the inner loops must be very close together to close, see Figure 9(c). We were unable to directly compare to [9] because we do not have access to their Matlab implementation. They report that they can generate a constraint-satisfying configuration for a 1000 loop chain with 3 links per loop (running time and topology not reported).

# Loops	Loop Size	Time (s) T1	Time (s) T2
1	1024	5.085	
2	512	9.403	
4	256	12.422	12.332
8	128	16.059	15.640
16	64	17.770	16.598
32	32	24.910	22.038
64	16	38.526	33.053
128	8	57.139	49.272
256	4	88.5424	80.222

(a)

# Links	# Loops	Time (s) No CD T1	No CD T2	No CD T3	With CD T1	With CD T2	With CD T3
6	1		0.009			0.014	
30	7	0.324	0.321	0.320	1.595	402.378	1.611
73	19	1.137	1.140	2.707	12.670	68.097	181.678
133	37	2.623	2.415	10.167	144.879	1182.302	55067.010

(b)



(c) T2, 30 links (d)T3, 133 links

Fig. 9. Running time to generate 100 samples of multiple loop systems, averaged over 10 runs. (a) Results for topologies 1 and 2 (T1 and T2). (b) Results for all three topologies, both with and without collision detection. Example configuration for (c) topology 2 and (d) topology 3.

6 Application: Restricted End Effector Sampling

Here we discuss how to apply this reachable distance framework to efficiently sample configurations of an articulated linkage when its end effector is required to remain within a specified boundary, such as a work area or safe zone. Consider the robot in Figure 1(b). The fixed base manipulator end effector is restricted to remain inside the box B. Randomly sampling such a configuration in joint space is unlikely. Recall that the RR of this robot is $[l_{min}, l]$ where l is the sum of its link lengths, and l_{min} is the minimum ARR. Let the *range* of the robot be the distance from the base to the end effector. Observe that all configurations with end effectors inside the boundary have ranges $[d_{min}, d_{max}]$ where d_{min} (d_{max}) is the minimum (maximum) distance between the robot's base and B. The range $[d_{min}, d_{max}]$ is much smaller than the original range $[l_{min}, l]$. We simply restrict the ARR of the end effector to $[d_{min}, d_{max}]$.

Enforcing end effector placement. We can easily restrict the end effector distance by setting the ARR of the vlink connecting the base and the end effector to $[d_{min}, d_{max}]$ and calling the above sampling scheme. However, this alone does not guarantee that the end effector will be in B. A simple way to guarantee this is to first randomly sample a point b in B. Then we calculate the distance between the robot's base and b. We set the ARR of the vlink connecting the base and the end effector to $[b, b]$ and sample as before. Let e be the resulting end effector placement. We then compute a rotation R to rotate e to b and apply this rotation to the robot base.

Results. Here we compare the performance of our sampling scheme in RD-space to uniform random and RRT-style sampling [17] in joint space. (For RRT-style sampling, we do not count the time to check edge validity.) All samplers use the same validity checker: first checking the end effector placement and then a collision check.

Table 1. Restricted end effector sampling comparison.

Method	# Robot Links	Time (s)	Samples Generated	Sample Attempts
Reachable Distance	3	0.02	1000.0	1000.5
	10	0.11	1000.0	1817.8
	20	0.50	1000.0	4311.9
	50	7.13	1000.0	24486.2
	100	29.29	1000.0	51835.3
Uniform Random	3	13.89	1000.0	1326106.9
	10	142.04	1000.0	5418904.7
	20	3572.60	82.1	9568969.6
	50	n/a	n/a	n/a
	100	n/a	n/a	n/a
RRT	3	519.11	1000.0	392073.6
	10	49.67	1000.0	106202.3
	20	66.11	1000.0	121840.4
	50	n/a	n/a	n/a
	100	n/a	n/a	n/a

The robots have 3 to 100 links of varying length, while the sum of each robot's link lengths is the same.

Table 1 shows how each sampler performs. Each sampler was asked to generate 1000 valid configurations within 1 hour. Results are averaged over 10 runs. Neither uniform random sampling or RRT were able to generate a single sample in 1 hour for 50 and 100 links. Reachable distance sampling was not only able to generate samples for 100 links, it could do so faster than RRT for *any* robot and faster than uniform random sampling for any robot other than 3 links.

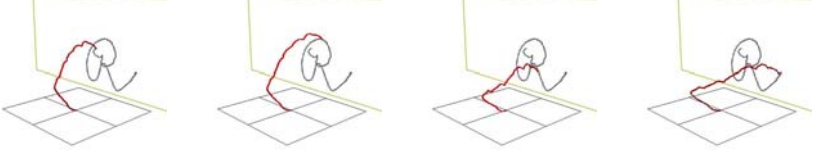
Note that RRT performs significantly slower for the 3 link robot than for the other larger robots. The success of RRT depends on the placement of the starting configuration. For the 3 link robot, minor changes in joint angles of the starting configuration pull the end effector outside the restricted area. Thus, RRT spends more time on the initial tree samples for the 3 link robot.

7 Application: Drawing/Sculpting

An interesting application is robotic drawing and sculpting. Figure 1(c) displays an articulated linkage drawing the letter "R" on a canvas. There are many applications in robotics where the manipulator needs to follow a trajectory. Here it is not sufficient for the planner to keep the robot's end effector in a restricted space (e.g., the canvas); it must also follow a specific trajectory.

Enforcing the End Effector Trajectory. Again we can take advantage of the RD-tree. We propose a two phase approach. With a local planner, we first find a valid path between q_{min} and q_{max} where q_{min} (q_{max}) is a configuration with end effector distance d_{min} (d_{max}) and d_{min} (d_{max}) is the minimum (maximum) distance between the base and any point in the drawing trajectory. We then use the precomputed path to follow the drawing trajectory by selecting the configuration in the path with the appropriate end effector length and rotating the configuration to align with the drawing target for each point along the drawing trajectory. Note that if the local planner returns a path containing configurations with end effectors outside the range $[d_{min}, d_{max}]$, we simply remove these portions.

Results. We applied our drawing algorithm to robots with 3, 5, 10, 20, 50, and 100 links. Figure 10 shows the planning results of an articulated robotic arm drawing

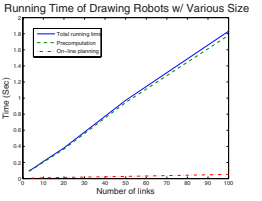


(e) (f) (g) (h)

Fig. 10. A 100-link robotic arm drawing the letter "R" on the canvas.

Time (s)	Number of Links 3	5	10	20	50	100
Phase I	0.090	0.119	0.203	0.366	0.945	1.780
Phase II	0.006	0.006	0.009	0.014	0.027	0.052
Total	0.096	0.125	0.212	0.380	0.972	1.831

(a)



(b)

Fig. 11. (a) Running times of different drawing robots averaged over 10 runs. (b) Running time scalability.

the character "R" on a canvas for a 100 link robot. The target trajectory is composed of 560 points generated from a scanned in drawing.

The table in Figure 11(a) shows the running time needed for each robot. Phase I is the time to perform the local planning, and phase II is the time to morph the path to the drawing trajectory. In all cases, the total time is very small and phase II planning is short enough for on-line applications. Figure 11(b) shows how the planning time scales with the robot's DOF. As expected, the phase I pre-computation is linear in the robot's DOF and dominates the overall planning. Phase II remains nearly constant and is thus well-suited for on-line applications. Note that other randomized planners such as PRMs or RRT would be infeasible here since it is even more constrained than restricted end effector sampling where they could not generate a single valid sample for a 50 link robot in 1 hour.

8 Conclusion

We presented a new method to plan the motion of spatially constrained systems based on a hierarchical representation as a set of RD-trees. We showed how this framework can efficiently sample and plan motions for closed chain systems (with single and multiple loops), restricted end effector sampling, and robotic arm drawing/sculpting. For all system types, our experimental results show that this framework is fast and efficient in practice, making the cost of generating constraint satisfying configurations comparable to traditional sampling without constraints.

Acknowledgements. The work of X. Tang was done when he was a Ph.D. student in the Department of Computer Science and Engineering at Texas A&M University. This research supported in part by NSF Grants EIA-0103742, ACR-0113971, CCR-0113974, ACI-0326350, CRI-0551685, CCF-0833199, CCF-0830753, by Chevron, IBM, Intel, HP, and by King Abdullah University of Science and Technology (KAUST) Award KUS-C1-016-04. Thomas supported in part by an NSF Graduate Research Fellowship, a PEO Scholarship, a Department of Education Graduate Fellowship (GAANN), and an IBM TJ Watson PhD Fellowship.

References

1. Cortés, J., Siméon, T.: Probabilistic motion planning for parallel mechanisms. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Taipei, Taiwan, pp. 4354–4359 (2003)
2. Cortés, J., Siméon, T.: Sampling-based motion planning under kinematic loop-closure constraints. In: Algorithmic Foundations of Robotics VI, pp. 75–90. Springer, Heidelberg (2005)
3. Cortés, J., Siméon, T., Laumond, J.P.: A random loop generator for planning the motions of closed kinematic chains using PRM methods. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Washington, DC, pp. 2141–2146 (2002)
4. Garber, M., Lin, M.C.: Constraint-based motion planning using Voronoi diagrams. In: Algorithmic Foundations of Robotics V, pp. 541–558. Springer, Heidelberg (2003)
5. Han, L.: Hybrid probabilistic roadmap — Monte Carlo motion planning for closed chain systems with spherical joints. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), New Orleans, LA, pp. 920–926 (2004)
6. Han, L., Amato, N.M.: A kinematics-based probabilistic roadmap method for closed chain systems. In: New Directions in Algorithmic and Computational Robotics, pp. 233–246. A. K. Peters, Boston (2001)
7. Han, L., Rudolph, L.: Inverse kinematics for a serial chain with joints under distance constraints. In: Robotics Science and Systems II, pp. 177–184. MIT Press, Cambridge (2007)
8. Han, L., Rudolph, L.: A unified geometric approach for inverse kinematics of a spatial chain with spherical joints. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Roma, Italy, pp. 4420–4427 (2007)
9. Han, L., Rudolph, L.: Simplex-tree based kinematics of foldable objects as multi-body systems involving loops. In: Robotics Science and Systems IV. MIT Press, Cambridge (2009)
10. Han, L., Rudolph, L., Blumenthal, J., Valodzin, I.: Stratified deformation space and path planning for a planar closed chain with revolute joints. In: Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR), New York (2006)
11. Kallmann, M., Aubel, A., Abaci, T., Thalmann, D.: Planning collision-free reaching motion for interactive object manipulation and grasping. Computer Graphics Forum 22(3), 313–322 (2003)
12. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Automat. 12(4), 566–580 (1996)
13. Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., Casal, A.: Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS), Osaka, Japan, pp. 546–553 (1996)
14. Kotay, K., Rus, D., Vona, M., McGray, C.: The self-reconfiguring robotic molecule: Design and control algorithms. In: Agarwal, P.K., Kavraki, L.E., Mason, M.T. (eds.) Robotics: The Algorithmic Perspective, pp. 375–386. A. K. Peters, Boston (1998)
15. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers, Boston (1991)
16. LaValle, S., Yakey, J., Kavraki, L.: A probabilistic roadmap approach for systems with closed kinematic chains. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Detroit, MI, pp. 1671–1676 (1999)
17. LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: Progress and prospects. In: New Directions in Algorithmic and Computational Robotics, pp. 293–308. A. K. Peters (2001)

18. Merlet, J.-P.: Still a long way to go on the road for parallel mechanisms. In: ASME Bienneal Mech. Rob. Conf., Montreal, Canada (2002)
19. Milgram, R.J., Trinkle, J.C.: The geometry of configuration spaces for closed chains in two and three dimensions. Homology, Homotopy Appl. 6(1), 237–267 (2004)
20. Nguyen, A., Guibas, L.J., Yim, M.: Controlled module density helps reconfiguration planning. In: New Directions in Algorithmic and Computational Robotics, pp. 23–36. A. K. Peters, Boston (2001)
21. Oriolo, G., Mongillo, C.: Motion planning for mobile manipulators along given end-effector paths. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Barcelona, Spain, pp. 2154–2160 (2005)
22. Oriolo, G., Ottavi, M., Vendittelli, M.: Probabilistic motion planning for redundant robots along given end-effector paths. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS), Lausanne, Switzerland, pp. 1657–1662 (2002)
23. Reif, J.H.: Complexity of the mover's problem and generalizations. In: Proc. 1EEE Symp. Foundations of Computer Science (FOCS), San Juan, Puerto Rico, October 1979, pp. 421–427 (1979)
24. Singh, A.P., Latombe, J.-C., Brutlag, D.L.: A motion planning approach to flexible ligand binding. In: Int. Conf. on Intelligent Systems for Molecular Biology (ISMB), pp. 252–261 (1999)
25. Tang, X., Thomas, S., Amato, N.M.: Planning with reachable distances: Fast enforcement of closure constraints. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), Roma, Italy, pp. 2694–2699 (2007)
26. Trinkle, J.C., Milgram, R.J.: Complete path planning for closed kinematic chains with spherical joints. Int. J. Robot. Res. 21(9), 773–789 (2002)
27. Whitney, H.: Non-separable and planar graphs. Transactions of the American Mathematical Society 34, 339–362 (1932)
28. Whitney, H.: 2-isomorphic graphs. American Journal of Mathematics 55, 245–254 (1933)
29. Xie, D., Amato, N.M.: A kinematics-based probabilistic roadmap method for high DOF closed chain systems. In: Proc. IEEE Int. Conf. Robot. Autom. (ICRA), New Orleans, LA, pp. 473–478 (2004)
30. Yakey, J.H., LaValle, S.M., Kavraki, L.E.: Randomized path planning for linkages with closed kinematic chains. IEEE Trans. Robot. Automat. 17(6), 951–958 (2001)
31. Yao, Z., Gupta, K.: Path planning with general end-effector constraints: using task space to guide configuration space search. In: Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS), Edmonton, Alberta, Canada, pp. 1875–1880 (2005)

Part IX
Medical Applications

3D Motion Planning Algorithms for Steerable Needles Using Inverse Kinematics

Vincent Duindam, Jijie Xu, Ron Alterovitz, Shankar Sastry, and Ken Goldberg

Abstract. Steerable needles can be used in medical applications to reach targets behind sensitive or impenetrable areas. The kinematics of a steerable needle are nonholonomic and, in 2D, equivalent to a Dubins car with constant radius of curvature. In 3D, the needle can be interpreted as an airplane with constant speed and pitch rate, zero yaw, and controllable roll angle.

We present a constant-time motion planning algorithm for steerable needles based on explicit geometric inverse kinematics similar to the classic Paden-Kahan subproblems. Reachability and path competitivity are analyzed using analytic comparisons with shortest path solutions for the Dubins car (for 2D) and numerical simulations (for 3D). We also present an algorithm for local path adaptation using null-space results from redundant manipulator theory. The inverse kinematics algorithm can be used as a fast local planner for global motion planning in environments with obstacles, either fully autonomously or in a computer-assisted setting.

1 Introduction

Steerable needles [18] form a subclass of flexible needles that provide steerability due to asymmetric forces acting at the needle tip, for example due to a beveled

Vincent Duindam and Shankar Sastry
Dept. of EECS, University of California, Berkeley, CA
e-mail: v.duindam@ieee.org, sastry@coe.berkeley.edu

Jijie Xu
Ph.D. Program, GCCIS, Rochester Institute of Technology, NY
e-mail: jijie.xu@rit.edu

Ron Alterovitz
Dept. of CS, University of North Carolina at Chapel Hill, NC
e-mail: ron@cs.unc.edu

Ken Goldberg
Dept. of EECS and IEOR, University of California, Berkeley, CA
e-mail: goldberg@berkeley.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 535–549.
springerlink.com

Fig. 1. Model setup of a steerable needle and a kinematically equivalent airplane with fixed speed and pitch rate, zero yaw, and controllable roll rate.

surface [18] or a kink near the end of the needle [7]. By rotating the needle at the base, the orientation of the tip can be changed and hence the trajectory of the needle can be controlled. Steerable needles differ in this respect from symmetric flexible needles, which can only be controlled by applying asymmetric forces at the base [4], not at the tip. The extra mobility of steerable needles over rigid needles can be harnessed in medical applications such as brachytherapy [2] and brain surgery [7] to reach difficult targets behind sensitive or impenetrable areas.

Experimental studies [17] show that the motion of steerable needles can be approximated as having a constant radius of curvature that is independent of insertion speed. The control inputs for the needle are the insertion speed and rotation (roll) angle, although for motion planning (the topic of this paper) insertion speed is often not important. The rotation angle is then the only real control input and trajectories can be parameterized by insertion depth. A steerable needle is thus kinematically equivalent to an airplane with fixed speed and pitch rate, zero yaw, and controllable roll rate (Fig. 1b).

Motion planning for steerable needles is an important problem and has been studied in several ways in literature. Most studies focus on planar motion, for which the control input reduces to switching between curve-left and curve-right. Alterovitz et al. [2, 1, 3] present a roadmap-based motion planning framework that explicitly incorporates motion uncertainty and computes the path that is most likely to succeed. Minhas et al. [12] show planning based on fast duty cycle spinning of the needle, effectively removing the limitation of a fixed-radius path but requiring continuous angular control input. Kallem et al. [10] introduce a controller that stabilizes the needle motion to a plane, allowing practical implementation of planar methods.

The first 3D motion planning algorithm was introduced by Park et al. [14, 15] and used diffusion of a stochastic differential equation to generate a family of solution paths. The authors also describe several extensions to avoid obstacles. Duindam et al. [6] presented a second 3D motion planning algorithm that uses cost function optimization to compute feasible paths in 3D environments with obstacles.

This paper presents a different solution to the 3D motion planning problem for steerable needles, based on inverse kinematics. We propose a new geometry-based algorithm inspired by the Paden-Kahan subproblems in traditional inverse kinematics algorithms [13]. Just as the Paden-Kahan subproblems, our algorithm (Sect. 3) can be fully described in geometric terms of intersecting lines, planes, and circles, and computing the solution simply requires a few trigonometric functions. We analyze reachability and competitivity of the solution (Sect. 4) using analytic comparison to the Dubins car solution and numerical simulations. We also present a method to locally adapt needle paths using null-motions (Sect. 5).

2 Problem Statement and Modeling Assumptions

2.1 Model Parameters and Assumptions

Throughout this paper, we only consider the idealized kinematics of the needle in a static environment. We assume that the motion of the needle is fully determined by the motion of the needle tip, that the motion of the needle tip is instantaneously along a perfect circle of constant radius r, and that rotations of the base are instantly transmitted to rotations of the tip. Experimental results [17] show that needle materials can be chosen such that the needle indeed moves along an arc of approximately constant radius, but the effects of tissue inhomogeneity, friction, and needle torsion can be significant and will require compensation [10] in practical applications.

Under these assumptions, the motion of the needle is determined kinematically by two control inputs: the insertion velocity, denoted v, and the tip rotation velocity, denoted ω. We present the kinematics model for general $v(\cdot)$ but remove this redundant degree of freedom in the next section.

Fig. 1c illustrates the model setup. We rigidly attach a coordinate frame Ψ_n to the tip of the needle, with axes aligned as in the figure, such that the z-axis is the direction of forward motion v and needle orientation ω, and the beveled tip causes the needle to rotate instantaneously around the line parallel to the x-axis and passing through the point $(0,-r,0)$.

Following standard robotics literature [13], the position and orientation of the needle tip relative to a reference frame Ψ_s can be described compactly by a 4×4 matrix $g_{sn}(t)\in SE(3)$ of the form

$$g_{sn}(t)=\begin{bmatrix} R_{sn}(t) & p_{sn}(t) \\ 0 & 1 \end{bmatrix} \tag{1}$$

with $R_{sn}\in SO(3)$ the rotation matrix describing the relative orientation, and $p_{sn}\in T(3)$ the vector describing the relative position of frames Ψ_s and Ψ_n.

The instantaneous linear and angular velocities of the needle are described by a twist $V_{sn}\in se(3)$ which in body coordinates Ψ_n takes the convenient form

$$V_{sn}^n(t) = \begin{bmatrix} 0 \\ 0 \\ v(t) \\ v(t)/r \\ 0 \\ \omega(t) \end{bmatrix} \qquad \hat{V}_{sn}^n(t) = \begin{bmatrix} 0 & -\omega(t) & 0 & 0 \\ \omega(t) & 0 & -v(t)/r & 0 \\ 0 & v(t)/r & 0 & v(t) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

in equivalent vector and matrix notation. The twist relates to g_{sn} as

$$\dot{g}_{sn}(t) = g_{sn}(t)\hat{V}_{sn}^n(t) \tag{3}$$

This kinematic model is the same as the unicycle model by Webster et al. [16]. When the twist is constant, (3) becomes a linear ordinary differential equation (ODE) that can be integrated as

$$g_{sn}(t) = g_{sn}(0)\exp(t\hat{V}_{sn}^n) \tag{4}$$

for which a relatively simple analytic expression exists [13]. In the path planning algorithms, we construct paths for which $\hat{V}_{sn}^n$ is piecewise constant and compute the resulting transformation using this efficient analytic expression.

2.2 Problem Statement

The objective of the motion planning algorithms in this paper is to find feasible paths between given start and goal configurations in the absence of obstacles. More precisely, the inputs to the algorithm are an initial needle pose $g_{\text{start}} \in SE(3)$ and a desired needle pose $g_{\text{goal}} \in SE(3)$. The outputs of the algorithm are control functions $v(\cdot)$ and $\omega(\cdot)$ and a finite end time $0 \leq T < \infty$, such that the solution $g_{sn}(\cdot)$ of the differential equation (3) with $g_{sn}(0) = g_{\text{start}}$ satisfies $g_{sn}(T) = g_{\text{goal}}$. If no feasible path can be found, the algorithm returns failure.

The kinematics equations (2) and (3) are invariant to time scaling, in the sense that the path traced out by the needle does not change if the control inputs $v(\cdot)$ and $\omega(\cdot)$ are scaled by the same (possibly time-varying) factor. Therefore, we can simplify the motion planning problem by assuming without loss of generality that $v(\cdot) \equiv 1$, which is equivalent to parameterization by insertion depth [10, 6]. The insertion time T thus represents the total path length since $\int_0^T |v(t)|dt = \int_0^T dt = T$.

Although motion planning is based on connecting general 3D poses (full position and orientation), solving a difference in initial or final roll angle is trivial as this degree of freedom is directly controlled through $\omega(\cdot)$. So although the inputs to the algorithm are general elements of $SE(3)$, we often mainly focus on path planning between given start and goal position and *direction* of the needle, i.e. only considering the z-axis of Ψ_n. Additionally required roll-rotations can be added directly at the start and end of the path.

3 Path Planning Using Inverse Kinematics

We present two motion planning solutions based on inverse kinematics, one for the planar (2D) case and one for the general spatial (3D) case. The motion planning problem is considered planar if the start position p_s, start direction z_s, goal position p_g, and goal direction z_g are all in the same plane.

In both inverse kinematics solutions, we look for a control input function $\omega(\cdot)$ of a very specific form, namely a function that is zero everywhere except for a fixed number of Dirac impulses (two in the planar case, four in the spatial case, see Fig. 2). Geometrically, this means we look for trajectories that are concatenations of a fixed number of circular segments with radius r: the needle moves along a circle when $\omega = 0$, and instantaneously changes direction at the time instants that ω is a Dirac impulse. Furthermore, the magnitudes of the Dirac impulses in the planar case are constrained to be exactly π, corresponding to a change in direction between curve-left and curve-right. We also choose a spatial solution for which the last two impulsive rotations are π, making the last three segments of the spatial path co-planar as well.

The specific choices in the structure of $\omega(\cdot)$ result in geometrically intuitive solutions that are relatively straight-forward to compute. The simplicity of the proposed solutions comes at the cost of not necessarily being optimal in terms of path length or total control effort. Practical implementations should clearly not use impulsive rotational control and constant insertion speed, but alternate between pure insertion until the desired depth t_i is reached, and pure rotation to the desired angle θ_i.

3.1 Inverse Kinematics in 2D

We first consider the planar path planning problem with $\omega(\cdot)$ as in Fig. 2a. The relative position of the start and goal are described by two displacements x and y, their relative orientation by a single angle θ. The purpose of the path planning algorithm is to find the three insertion depths t_1, t_2, t_3 describing a feasible needle path from start to goal. Note that the needle travels a distance $t_i = r\alpha_i$ when moving along a circle of radius r for α_i radians, and hence we can equivalently look for the three angles α_i in Fig. 3. These should be such that if we start at p_s in the direction z_s heading left, move $r\alpha_1$ forward, turn π, move $r\alpha_2$ forward, turn π, and move $r\alpha_3$

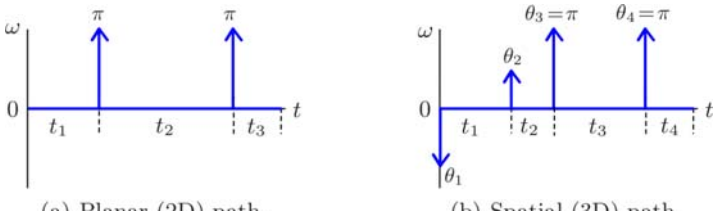


Fig. 2. Structure of the solution $\omega(\cdot)$ for the 2D and 3D motion planning problems.

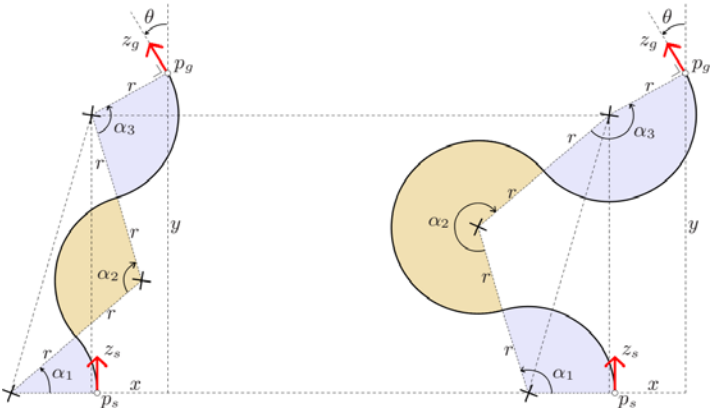


Fig. 3. Two geometric solutions for the same planar inverse kinematics problem, both using sequential bevel-left, bevel-right, bevel-left motions.

forward, we arrive exactly at the desired goal pose. The mirrored case starting with a right turn can be computed similarly.

We can solve for the angles α_i by looking at the setup in Fig. 3 and realizing that the three centers of rotation (marked by $\times$ in the figure) form a triangle with known edge lengths. Using the cosine rule for this triangle, we can write

$$\cos(\alpha_2) = 1 - \frac{(x + r - r\cos(\theta))^2 + (y - r\sin(\theta))^2}{8r^2} \tag{5}$$

This equation has two solutions for α_2, which correspond to the two paths shown in Fig. 3. With α_2 known, the other two angles follow uniquely as

$$\alpha_1 = \operatorname{atan2}(y - r\sin(\theta), x + r - r\cos(\theta)) - \frac{1}{2}(\pi - \alpha_2) \tag{6}$$

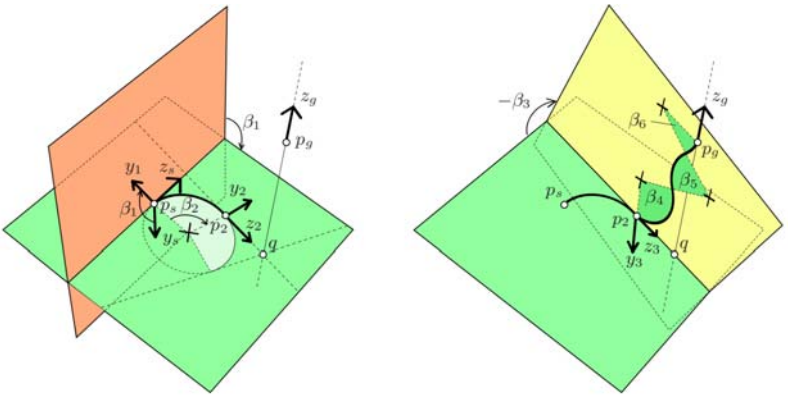
$$\alpha_3 = \theta - \alpha_1 + \alpha_2 \tag{7}$$

with atan2 the inverse tangent function solved over all quadrants. Since the needle can only move forward, angles must be chosen as $\alpha_i \in [0, 2\pi)$. The required insertion distances t_i follow immediately as $t_i = r\alpha_i$.

3.2 *Inverse Kinematics in 3D*

Now consider the 3D inverse kinematics problem of connecting two general poses in $SE(3)$ by a valid needle path. We propose one solution using eight consecutive insert and turn motions as shown in Fig. 2b; an explicit geometric solution using fewer motions is still an open problem.

The geometry of this solution is illustrated in Fig. 4. The problem is split into two parts: first, the needle is turned and inserted such that its instantaneous line of



(a) Rotate β_1 around z_s until the needle's (y,z)-plane contains q. Insert $r\beta_2$ until q is on the line through z_2.

(b) Rotate β_3 around z_2 until the needle's (y,z)-plane contains p_g. Solve the remaining planar problem.

Fig. 4. Geometric derivation of an inverse kinematic solution on $SE(3)$.

motion (the instantaneous direction of the needle) intersects the line describing the goal position and direction. Second, the remaining planar problem is solved using the solution from Sect. 3.1.

More precisely, we first choose any point q on the line defined by p_g and z_g. This point q will be the intersection point of the two lines defining the remaining planar problem. With q defined, the needle is first rotated by $\theta_1 = \beta_1$ until its (y,z)-plane contains q. The required angle β_1 satisfies

$$\tan(\beta_1) = -\frac{x_s^T(q-p_s)}{y_s^T(q-p_s)} \tag{8}$$

which has two solutions β_1 that differ by π.

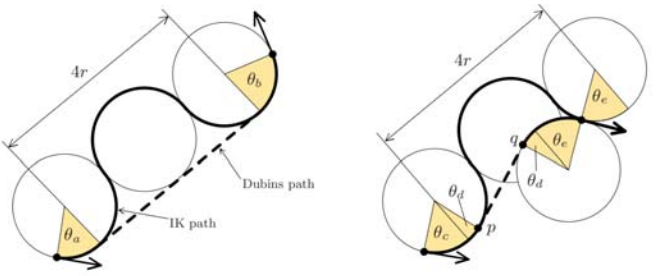
Second, the insertion distance $t_1 = r\beta_2$ is solved such that the line through the needle tip in the direction z_2 contains the point q. If q is outside the circle describing the needle motion along β_2, two solutions exist, with z_2 either pointing towards (as in the figure) or away from q. These solutions are

$$\beta_2 = \operatorname{atan2}\left(z_s^T q_v, y_1^T q_v\right) \pm \arccos\left(\frac{r}{|q_v|}\right) \tag{9}$$

with $q_v := q - p_s + r y_1$. No solution exists if q is inside the circle ($|q_v| < r$).

Third, the needle is rotated by $\theta_2 = \beta_3$ until p_g (and hence the whole line through q and p_g) is contained in the needle's (y,z)-plane:

$$\tan(\beta_3) = -\frac{x_2^T(p_g-p_2)}{y_2^T(p_g-p_2)} \tag{10}$$



(a) Left-straight-left Dubins path. (b) Left-straight-right Dubins path.

Fig. 5. Paths generated by the 2D IK algorithm vs. optimal paths for a Dubins car.

which again gives two solutions that differ by π. The remaining angles $\beta_4, \beta_5, \beta_6$ (corresponding to the time segments t_2, t_3, t_4) can then be solved using the 2D planner from Sect. 3.1.

In this algorithm, we have the following degrees of freedom to choose a solution. First is the choice of the point q: this can be anywhere on the line containing p_g and z_g, which means varying q generates a one-dimensional subspace of possible solutions. Second, we can choose one of two possible solutions β_i for each of the four angles in equations (8–10) and (5), resulting in 2^4 possible combinations. Not all of these choices may give feasible paths for a given start and end pose, and it is also not directly obvious which choice will result in the 'best' path between the two poses. Still, since the inverse kinematics solution can be computed very quickly, one can simply compute all combinations for a number of choices of q and pick the best solution, with 'best' defined for example using a cost function [6].

4 Reachability and Competitivity

To evaluate the quality of the paths generated by the presented inverse kinematics (IK) solutions, we study the set of reachable needle poses and competitivity [9, 8] of the computed solutions. Competitivity in this case refers to the path length of the computed solution; it has no relation to competitivity in the sense of computational speed (the IK algorithm runs in constant time).

4.1 Reachability and Competitivity in 2D

Consider first the solution to the planar problem as described in Sect. 3.1. The algorithm will clearly only find a solution if the right-hand side of (5) has norm less than or equal to one, or geometrically, if the centers of the circles tangent to the start and goal poses are no farther than $4r$ apart. This condition defines the set of reachable relative needle poses.

To describe competitivity of the algorithm, we compare the IK solutions to the optimal trajectory when allowing an infinite number of direction changes. In that

case, a trajectory can be generated with an arbitrary radius of curvature larger than or equal to r by asymmetrically switching between heading-left and heading-right and taking the limit of this switching frequency to infinity. This means that in the limit, the needle behaves like a Dubins car [5] with minimum radius of curvature r.

The optimal path for a Dubins car is known to consist of two circular arcs with radius r connected by another circular arc or a straight line [5]. For a given start and end pose, the IK solution only differs from the Dubins path if the connecting segment for the Dubins path is a straight line; the IK solution will still contain a (sub-optimal) circular arc. Furthermore, the Dubins path may start and end with circular arcs in the same direction or in opposite directions (Fig. 5), whereas the IK solution always starts and ends with a turn in the same direction. It is intuitively clear that the largest difference in path length occurs at the border of the reachable space where the three circles of the IK solution are aligned. In the first case (Fig. 5a), the maximum ratio of the path lengths is

$$\sup \frac{\|\text{IK path}\|}{\|\text{Dubins path}\|} = \sup_{\theta_i \geq 0} \frac{r\theta_a + r\theta_b + 2\pi r}{r\theta_a + r\theta_b + 4r} = \frac{\pi}{2} \tag{11}$$

For the second case (Fig. 5b), we can compute the length of the straight-line segment $q - p$ as

$$\|q - p\|^2 = 4r^2 \left(2 - \sin(\theta_d) - \sin(\theta_e)\right)^2 + 4r^2 \left(\cos(\theta_d) - \cos(\theta_e)\right)^2 \tag{12}$$

from which we find that the maximum path length ratio equals

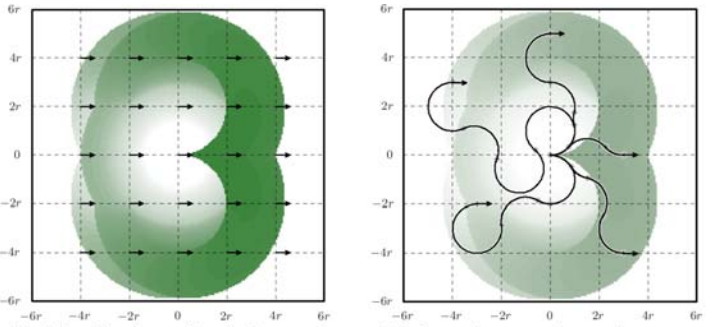
$$\sup \frac{\|\text{IK path}\|}{\|\text{Dubins path}\|} = \sup_{\theta_i \geq 0} \frac{r\theta_c + 2\pi r - r\theta_e}{r\theta_c + 2r\theta_d + r\theta_e + \|q - p\|} \approx 1.63 > \frac{\pi}{2} \tag{13}$$

The degree of competitivity of the 2D IK solution is hence approximately 1.63. Note that this is a bound on the competitivity that does not take into account the number of direction changes; for medical applications, this number should be kept small to avoid excessive tissue damage.

4.2 Reachability and Competitivity in 3D

Continuing with the 3D IK solution from Sect. 3.2, we present a reachability and competitivity analysis based on numerical simulation. Formal geometric proofs and bounds of the algorithm are subject of future research; at this point we do not have a good approximation for the optimal path and simply compare the IK solutions to the Euclidean distance between the start and goal positions. Future work could compare the presented solution to the paths generated by the method of Park et al. [15]. We take $q = p_g$ throughout the analysis; different choices give qualitatively similar results.

First consider Fig. 6. This illustrates the lengths of the IK trajectories starting at the center of the figure and ending at goal positions in the plane of the figure,



(a) Visualization of relative path lengths (shorter paths are darker) as a function of the goal position.

(b) Several example paths corresponding to various points in the image (∗ delimits path segments).

Fig. 6. Reachability and relative path lengths obtained using the inverse kinematics algorithm. The algorithm tries to find a path from the center of the image to each pixel in the image, with both start and goal direction aimed to the right.

with start and goal directions equal and pointing to the right. The brightness of each pixel indicates the length of the IK path divided by the Euclidean distance between the start and goal locations: dark colors represent small ratios (good paths) while light colors represent large ratios (bad paths). Fig. 6b illustrates several examples of solution trajectories.

A set of reachable states shows up in the figure as the 'figure-eight' around the start location. Points outside this shape cannot be reached for the given goal direction. The figure also shows a distinction between poses that can be reached with reasonably short curves (darker region) and poses that require significantly longer paths (lighter region). This distinction is sharp in the area in front of the needle (right side of the figure) but is more diffuse for poses on the sides of the needle (top and bottom of the figure). If we consider competitivity in an informal way, meaning whether the algorithm can generated paths of reasonable lengths, we can say that the algorithm is competitive in the darker region of the figure; relative poses that are in the lighter region may be reachable, but the paths are so unwieldy that they are of little practical use in medical applications.

Fig. 7 shows additional plots generated by varying the two remaining degrees of freedom in placing the goal pose: the in-plane yaw angle and out-of-plane pitch angle (the inverse kinematics solution is invariant to roll about the initial and final needle directions).

The figure shows that as the goal direction is turned away from the straight-ahead case (change in yaw), the set of reachable and competitive paths rotates in the same direction while maintaining a roughly similar shape. As the goal direction is rotated out of plane (change in pitch), the competitive paths near the starting pose disappear until only poses at a significant distance (three to five times the radius of curvature) are competitive.

In the three-dimensional case, it remains a difficult task to precisely characterize the set of poses that can be reached with a reasonably short path. Comparing the

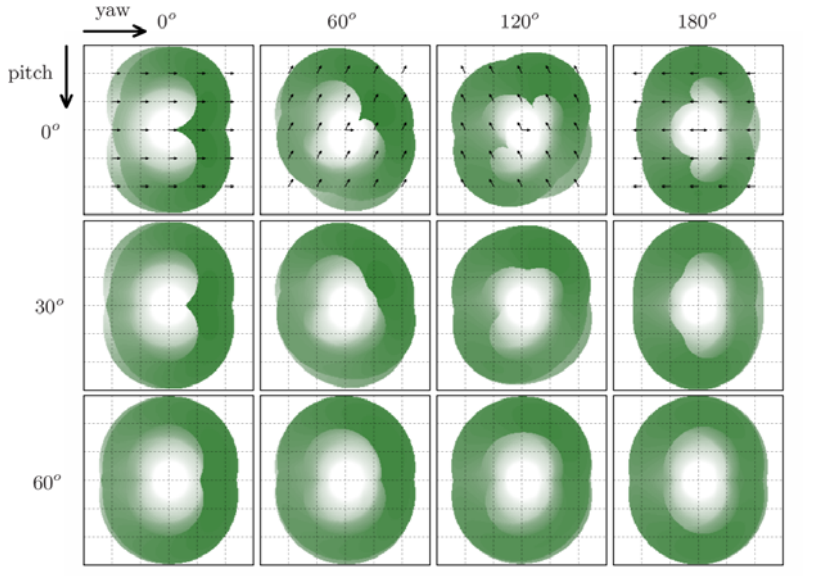


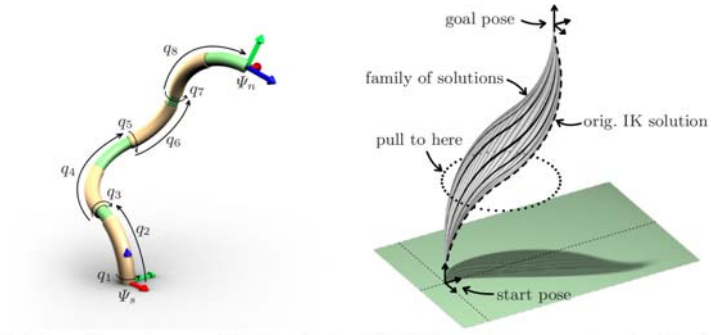
Fig. 7. Reachability and relative path lengths as in Fig. 6 for various relative yaw and pitch angles for the goal pose. Grid lines are $2r$ units apart.

solutions to the Euclidean distance provides some insight but no global bound on the solutions: competitivity measures are unbounded when comparing to the Euclidean distance, since for infinitely close but non-collinear needle orientations the path length of the IK solution remains finite.

To use the inverse kinematics planner as a local planner in global roadmap-based planning methods such as the Probabilistic Roadmap method [11], it is important to characterize the set of goal poses that are reachable from a given needle pose. Numerical simulations such as those shown in Fig. 7 can be used to construct an approximation of the set of goal poses that can be reached with a competitive path (with 'competitive' in the informal meaning of 'reasonably short path'). This set can be used as the definition of 'neighborhood', i.e. those needle poses that are likely to be connectible and can become edges in the roadmap if they do not intersect obstacles.

5 Path Adaptation Using Null-Motions

The presented inverse kinematics (IK) solution can be computed very quickly but is generally not the optimal solution in the sense of avoiding obstacles or minimizing path length or required control effort. Earlier work considered path planning as a pure numerical optimization problem [6], but in this section we show how sub-optimal paths such as those generated by the IK planner can be locally optimized and adjusted using null-motions.



(a) A needle path as an 8-joint robot. (b) Null-motions generate a family of needle paths between start and goal.

Fig. 8. Representation of a needle path as a robot with eight joints, and use of its null motions to generate a family of solution curves by pulling in different directions.

Consider an IK solution between two general 3D needle poses. By construction, this solution describes a path from start to goal consisting of eight consecutive turning and insertion control actions. We can think of this path as a redundant serial robot manipulator arm with eight joints (Fig. 8a). Since the relative pose of the goal is given by six parameters, standard robotics theory [13] tells us that the robot has a two-dimensional space of null-motions, provided it is not at a singularity. If the joints are moved in this null-motion space, the shape of the robot (i.e. the shape of the needle path) will change without changing the pose of the end effector (i.e. the needle tip).

The set of null motions is described by the null space of the geometric Jacobian $J(q) \in \mathbb{R}^{6\times 8}$ of the robot, which relates the spatial twist V^s_{sn} to the joint velocities $\dot{q}$ as $V^s_{sn} = J(q)\dot{q}$ [13]. Given the Jacobian, we can change the shape of the path by changing the joint angles in such a way that $\dot{q} \in \text{Null}(J(q))$ at all times.

Fig. 8b shows an example of how one inverse kinematics solution can be locally transformed in this way into a family of solution curves. Intuitively, this set of curves was generated by starting from the IK solution indicated in the figure (the dashed line), and 'pulling' on the curve from several points laid out in a circle, while holding the start and end pose of the needle fixed. More precisely, we model the robot as a viscous system with damping in each joint that counteracts applied forces, and write the governing equations as

$$\dot{q} = B(q)J_i^T(q)F_i + B(q)J^T(q)\lambda \tag{14}$$

$$0 = J(q)\dot{q} \tag{15}$$

with F_i the wrench [13] corresponding to the (known) externally applied pulling force, $J_i(q)$ the Jacobian of link i at which F_i is applied, λ the required constraint wrench acting at the tip to constrain its motion, and $B(q) > 0$ the symmetric positive-definite inverse damping matrix that relates the joint torques τ to the joint velocities as $\dot{q} = B(q)\tau$. The first equation relates the total torque (due to external forces F_i

and λ) to the change in the joint angles, the second equation describes the end point constraint that should be satisfied. Note that these equations do not relate to any actual physical needle motions and only represent a mathematical procedure.

Substituting (14) into (15), solving for λ, and substituting back into (14) results in an unconstrained equation for $\dot{q}$ that no longer contains λ:

$$\dot{q} = \left(I - BJ^T(JBJ^T)^{-1}J\right)BJ_i^T F_i \tag{16}$$

This ODE describes the evolution of q under the influence of an external wrench F_i and tip constraint $J\dot{q} = 0$. It has a unique solution if B is invertible and J has full rank (no singularity). Equation (16) projects the velocity $BJ_i^T F_i$ due to the wrench F_i along the columns of BJ^T onto the null space of J. The matrix $B(q)$ defines a metric on the space of torques that can be chosen in any appropriate way, e.g. as a function that drives the system away from configurations that are singular or contain negative-length path segments.

For the example of Fig. 8b, we chose B diagonal with $B_{jj}(q) \to 0$ as $q_j \to 0$ for all joints j describing insertion path segments, thus avoiding negative-length path segments. We applied a linear force in the middle of the kinematic chain (link 5), directed toward one of the dots, and integrated (16) over time to obtain the pod-shaped family of needle paths shown in the figure.

This method of path adaptation can be used in fully automated motion planning (e.g. to perform gradient descent on some cost function with penalty costs for obstacle penetration) with changes in q constrained to be null motions. More directly, in interactive (computer-assisted) motion planning scenarios, it can provide the user with an intuitive path adjustment tool similar to the control points on a spline curve that can be moved to change its local shape. Although there is no guarantee this approach will always work, it provides the user with an additional tool to construct suitable needle paths.

6 Conclusions and Future Work

This paper presents constant-time geometrically motivated motion planning algorithms for steerable needles and airplanes with constant speed and pitch rate, zero yaw, and controllable roll. The first algorithm uses inverse kinematics (IK) to explicitly compute feasible paths in 3D, the second uses null-motions to adapt paths to avoid obstacles or achieve other objectives.

As briefly discussed, these algorithms can be used as components in larger computer-assisted motion planning schemes that use limited user-input to guide automatic local planning. In future work, we also plan to use the IK algorithm as a local planner in (autonomous) roadmap-based algorithms such as PRM [11]. Recent results using Rapidly-Exploring Random Trees [19] are encouraging, although computation requirements are several orders of magnitude larger than with direct optimization-based algorithms [6].

Another main future direction of our research is to find a systematic way to include uncertainty during motion planning. Our application of steerable needles

contains several sources of uncertainty, including needle motion uncertainty, tissue flexibility and friction, and sensing inaccuracies. These uncertainties should be taken into account in the motion planning stage, as discussed and implemented for the 2D case in previous work [3]. The presented fast local motion planning algorithm can be used to quickly test connectivity and iteratively study the effect of perturbations.

Finally, reachability and competitivity analyses were presented for the 2D and 3D inverse kinematics algorithms. In future work, we plan to extend the analysis of the 3D algorithm to provide bounds on the competitivity compared to the optimal shortest-path solution. A promising direction is the comparison with paths generated by the approach of Park et al. [15].

Acknowledgments

This work is supported in part by the National Institutes of Health under grants R01 EB006435 and F32 CA124138 and by the Netherlands Organization for Scientific Research. We thank Professors Okamura, Chirikjian, and Cowan from Johns Hopkins University for the helpful comments and suggestions, and Prof. Cowan for the idea of using null-motions for path adaptation.

References

1. Alterovitz, R., Branicky, M., Goldberg, K.: Constant-curvature motion planning under uncertainty with applications in image-guided medical needle steering. In: Proceedings of Workshop on the Algorithmic Foundations of Robotics (July 2006)
2. Alterovitz, R., Goldberg, K., Okamura, A.: Planning for steerable bevel-tip needle insertion through 2D soft tissue with obstacles. In: Proceedings of the IEEE International Conference on Robotics and Automation, April 2005, pp. 1640–1645 (2005)
3. Alterovitz, R., Siméon, T., Goldberg, K.: The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In: Proceedings of Robotics: Science and Systems (June 2007)
4. DiMaio, S., Salcudean, S.: Needle steering and motion planning in soft tissues. IEEE Transactions on Biomedical Engineering 52(6), 965–974 (2005)
5. Dubins, L.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics 79(3), 497–516 (1957)
6. Duindam, V., Alterovitz, R., Sastry, S., Goldberg, K.: Screw-based motion planning for bevel-tip flexible needles in 3D environments with obstacles. In: Proceedings of the IEEE International Conference on Robotics and Automation, May 2008, pp. 2483–2488 (2008)
7. Engh, J., Podnar, G., Kondziolka, D., Riviere, C.: Toward effective needle steering in brain tissue. In: Proc. 28th Annu. Intl. Conf. IEEE Eng. Med. Biol. Soc., pp. 559–562 (2006)
8. Gabriely, Y., Rimon, E.: Competitive Complexity of Mobile Robot On Line Motion Planning Problems. In: Algorithmic Foundations of Robotics VI. STAR, vol. 17, pp. 155–170. Springer, Heidelberg (2005)
9. Icking, C., Klein, R.: Competitive strategies for autonomous systems. In: Modelling and Planning for Sensor Based Intelligent Robot Systems, pp. 23–40 (1995)

10. Kallem, V., Cowan, N.: Image-guided control of flexible bevel-tip needles. In: Proceedings of the IEEE International Conference on Robotics and Automation, April 2007, pp. 3015–3020 (2007)
11. Kavraki, L., Švestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
12. Minhas, D., Engh, J., Fenske, M., Riviere, C.: Modeling of needle steering via duty-cycled spinning. In: Proceedings of the International Conference of the IEEE EMBS Cité Internationale, August 2007, pp. 2756–2759 (2007)
13. Murray, R., Li, Z., Sastry, S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
14. Park, W., Kim, J., Zhou, Y., Cowan, N., Okamura, A., Chirikjian, G.: Diffusion-based motion planning for a nonholonomic flexible needle model. In: Proceedings of the IEEE International Conference on Robotics and Automation, April 2005, pp. 4611–4616 (2005)
15. Park, W., Liu, Y., Zhou, Y., Moses, M., Chirikjian, G.: Kinematic state estimation and motion planning for stochastic nonholonomic systems using the exponential map. Robotica 26, 419–434 (2008)
16. Webster III, R., Kim, J., Cowan, N., Chirikjian, G., Okamura, A.: Nonholonomic modeling of needle steering. International Journal of Robotics Research 5/6, 509–525 (2006)
17. Webster III, R., Memisevic, J., Okamura, A.: Design considerations for robotic needle steering. In: Proceedings of the IEEE International Conference on Robotics and Automation, April 2005, pp. 3588–3594 (2005)
18. Webster III, R., Okamura, A., Cowan, N., Chirikjian, G., Goldberg, K., Alterovitz, R.: Distal bevel-tip needle control device and algorithm. US patent pending 11/436,995 (May 2006)
19. Xu, J., Duindam, V., Alterovitz, R., Goldberg, K.: Motion planning for steerable needles in 3D environments with obstacles using rapidly-exploring random trees and backchaining. In: Proceedings of the IEEE Conference on Automation Science and Engineering, August 2008, pp. 41–46 (2008)

Modeling Structural Heterogeneity in Proteins from X-Ray Data

Ankur Dhanik, Henry van den Bedem, Ashley Deacon, and Jean Claude Latombe

Abstract. In a crystallographic experiment, a protein is precipitated to obtain a crystalline sample (crystal) containing many copies of the molecule. An electron density map (EDM) is calculated from diffraction images obtained from focusing X-rays through the sample at different angles. This involves iterative phase determination and density calculation. The protein conformation is modeled by placing the atoms in 3-D space to best match the electron density. In practice, the copies of a protein in a crystal are not exactly in the same conformation. Consequently the obtained EDM, which corresponds to the cumulative distribution of atomic positions over all conformations, is blurred. Existing modeling methods compute an "average" protein conformation by maximizing its fit with the EDM and explain structural heterogeneity in the crystal with a harmonic distribution of the position of each atom. However, proteins undergo coordinated conformational variations leading to substantial correlated changes in atomic positions. These variations are biologically important. This paper presents a sample-select approach to model structural heterogeneity by computing an ensemble of conformations (along with occupancies) that, collectively, provide a near-optimal explanation of the EDM. The focus is on deformable protein fragments, mainly loops and side-chains. Tests were successfully conducted on simulated and experimental EDMs.

1 Introduction

Proteins are not rigid molecules [12, 19]. Each atom is subject to small, temperature-dependent high-frequency vibrations about its equilibrium position. In addition, in

Ankur Dhanik
Mechanical Engineering Department, Stanford University, Stanford, CA 94305, USA
e-mail: ankurd@stanford.edu

Henry van den Bedem and Ashley Deacon
Joint Center for Structural Genomics, SLAC, Menlo Park, CA 94025, USA
e-mail: {vdbedem,adeacon}@slac.stanford.edu

Jean Claude Latombe
Computer Science Department, Stanford University, Stanford, CA 94305, USA
e-mail: latombe@cs.stanford.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 551–566.
springerlink.com

its native state, a protein may also undergo coordinated lower-frequency conformational variations leading to correlated changes in atomic coordinates. Such diffusive motions are of vital interest in the study of the protein's biological functions [29]. Accurately capturing such low-frequency protein dynamics from X-ray crystallography data has remained a challenge.

In a crystallographic experiment, a protein of known sequence is precipitated to obtain a crystalline sample (hereafter called a crystal) containing many copies of the molecule. A three-dimensional electron density map (EDM) is calculated from a set of diffraction images, obtained from focusing X-rays through the sample at different angles. This EDM is an array of voxels, each encoding an electron density. The protein conformation is then modeled by placing the atoms in 3-D space to best match the electron density [11].

In an ideal crystal, all copies of the precipitated protein would have the same conformation. In practice, this is not the case, and corresponding atoms in different cells of a crystal do not occupy exactly the same position. The resulting EDM corresponds to the cumulative distribution of atomic positions over all conformations in the crystal. For instance, an EDM may appear locally blurred when a fragment of the main-chain or a side-chain adopts two or more neatly distinct conformational states (also called *conformers*). To illustrate, Figure 1 shows an iso-surface of an EDM corresponding to a fragment (residues 104-112) of the protein with Protein Data Bank (PDB, [3]) ID 2R4I that occurs in two conformers. Extracting conformers from a locally disordered EDM is then akin to gleaning structure from a 3D image blurred by motion of the articulated subject.

Uncertainty in atomic positions is usually modeled with an *isotropic* Gaussian distribution. This model, further parameterized by the *temperature factor*, accounts for small vibrations about each atom's equilibrium position. Fitting an anisotropic (trivariate) Gaussian function requires estimating 9 parameters per atom, which for the complete model typically exceeds the amount of data in the EDM [28]. A sparser parameterization involves partitioning the protein into rigid bodies undergoing independent equilibrium displacements [22]. Owing to their "equilibrium-displacement" nature, these models are unable to accurately describe distinct conformational substates, such as those caused by the low-frequency diffusive motion of the protein [16, 29].

The presence of distinct conformers in a crystal has been observed on many occasions [4, 27, 29] and the importance of accurately representing structural heterogeneity by an ensemble of conformers has long been recognized [2, 13, 29]. However, while several programs are available for automatically building a structural model into an EDM to a high degree of accuracy [9, 10, 15, 20, 23], these have been engineered towards building a single conformer at unit occupancy. They often leave ambiguous electron density due to correlated changes in atomic coordinates uninterpreted. Building a heterogeneous protein model then requires substantial manual effort by skilled crystallographers using interactive graphics program. In [7, 17, 25] single-conformer, approximate starting models are perturbed to generate a multi-conformer ensemble. However, each one of these conformers can be seen as a possible interpretation of the EDM. Together, they do not provide a

collective interpretation of the EDM. Automatically *building* a heterogeneous model into an EDM is a formidable challenge, and any progress could have a major impact on the way protein models are stored in the PDB.

It is imperative to accurately represent the data from the earliest stages of model building [7, 16]. In a crystallography experiment, the phase angle of a diffracted beam is lost. Only magnitudes are measured on the sensitive surface of the detector [11]. Phases are estimated and improved by building and interpreting successive EDMs using Maximum Likelihood (ML) algorithms [20, 24]. However, disregarding structural heterogeneity in the successive EDMs or omitting fragments from a model altogether bias the phases in this procedure. Providing an ensemble of atom coordinates as initial values to ML algorithms could lead to improving the EDM more quickly.

In this paper, we present a new approach to automatically and accurately model heterogeneity in an EDM. Our main contribution lies in abandoning the single-conformer model in favor of a multi-conformer model where appropriate, and providing an estimate of the relative frequency of occurrence (called *occupancy*) in the crystal for each of the conformers. We focus on protein fragments, mostly loops, which are often the most deformable substructures in a protein [7, 25, 26]. Our method computes an occupancy-weighted *ensemble* of conformations that *collectively* best represents the input EDM. To this end, an idealized protein fragment is modeled as a kinematic linkage, with fixed groups of atoms as links and rotatable bonds as joints. Our method is based on a sample-select protocol, which adaptively alternates *sampling* and *selection* steps. Each sampling step generates a very large set of $\{\text{conformation}, \text{temperature factors}\}$ samples. A subsequent selection step applies an efficient linear-programming algorithm to concurrently fit this set of samples to the input EDM and compute the occupancy of each sample. Samples with small occupancies (less than 0.1) are then discarded. As the sampled space has very high dimensionality, the successive sampling steps consider portions of the protein fragment of increasing lengths. The overall sampling process is guided by the results obtained at previous selection steps. It should be emphasized that the algorithm *infers* the ensemble size from the data; it has no prior knowledge about the number of conformers.

This paper is divided into two main sections. In Section 2 we present and discuss results obtained with our method on both simulated and experimental EDMs. This section allows us to characterize more precisely the type of problem addressed in the paper. In Section 3 we describe in detail our sample-select method to model heterogeneity in an EDM.

2 Results and Discussion

Validation tests against simulated EDMs in Section 2.1 demonstrate that our method extracts the correct ensemble for a variety of fragment lengths over a range of resolution levels, noise levels, occupancies and temperature factors. They furthermore

show that the algorithm correctly identifies and models side-chains in multiple conformations.

We also tested our method against experimental EDMs. In Section 2.2, we show results obtained with the 398-residue Flavoprotein TM0755 (PDB ID 1VME). The main chain for residues A316-A325 is bi-modally disordered [27]. Our method models the two conformations to within 0.6Å RMSD[1]. In Section 2.3 we give additional results on experimental EDMs for side-chains in multiple conformations.

Depending on the length of the fragment to compute and the resolution of the EDM, our partly parallelized implementation takes 2-4 hours to complete.

2.1 Algorithm Validation with Simulated Data

Given a protein structure, the simulated EDM corresponding to its distribution of atoms can easily be calculated at different resolution levels while controlling the temperature factors and occupancy of individual atoms. Such a simulated EDM allows us to test our method, and understand the effects of experimental noise and discrete sampling with idealized geometry.

2.1.1 Single Conformer

We first validated our algorithm on simulated data corresponding to single-conformer fragments, computed at various resolution levels. Six fragments varying in length from 4 to 12 residues at various temperature factors were selected from the PDB. The algorithm consistently identified conformers in the simulated EDM within 0.4Å RMSD of the true conformers (see Table 1). In each case, the returned ensemble contains more than one conformation. However, all conformations in an ensemble are pairwise very close and could easily be merged in a post-processing step by a clustering algorithm. Multiple conformations are returned due to finite resolution in our sampling scheme as described in Section 3. To confirm that, we ran the same test again, but this time we added the true conformers to the sample set. Then for each EDM, our method returned only the true conformer.

The algorithm furthermore returns temperature factors to within a 10.0Å^2 interval of the average, true temperature factors. These temperature factors and coordinate errors are well within the radius of convergence of standard crystallographic refinement packages.

Side-chains commonly occur in multiple rotameric conformations in protein structures determined from X-ray data. To test if the algorithm correctly models side-chains in multiple rotameric conformations while the backbone is best represented by a single conformer, a second rotamer was added to a selected side-chain of each main-chain at 0.5/0.5 occupancy (see the last column of Table 1). The main-chain RMSDs differed not meaningfully from those found earlier, while all dual rotamers were identified at the correct occupancy and within 0.5Å RMSD.

[1] Unless otherwise noted, RMSD denotes the square root of the averaged squared distances between corresponding N, C_α, C_β, C, and O atoms.

Table 1. Single conformer results from validation tests using simulated data. Each row lists PDB ID, map resolution (Res, in Å), anchors and size of loop, average temperature factor of loop atoms in the PDB structure ($\bar{B}$obs, in Å^2), RMSD of calculated conformation to PDB conformation, and average temperature factor of calculated conformation ($\bar{B}$calc, in Å^2). All calculated occupancies sum to 1.0. The final column identifies a side-chain in dual conformation at 0.5/0.5 occupancy.

PDB ID	Res	Loop(size)	$\bar{B}$obs	RMSD	$\bar{B}$calc	side-chain
1AAJ	1.9	82-85(4)	7.9	0.19 0.22	12.9	GLU(84)
1BGC	2.3	40-43(4)	29.5	0.32 0.33 0.41	27.1	LYS(41)
1HFC	1.9	142-149(8)	11.7	0.29 0.31 0.37 0.42 0.44 0.71	10.1	LEU(147)
1Z8H	2.3	71-78(8)	40.9	0.35 0.39 0.56 0.61 0.69 0.69 0.77	36.0	HIS(73)
1TML	1.9	243-254(12)	11.5	0.41 0.41 0.43 0.44 0.55 0.95	12.7	THR(247)
1CTM	2.3	142-149(12)	38.8	0.36 0.44 0.44 0.52 0.68 0.68	34.8	ARG(18)

2.1.2 Dual Conformers

The 123-residue protein with PDB ID 2R4I, a NTF-3 like protein, was solved by the Joint Center for Structural Genomics (JCSG) at a resolution of 1.6Å. The asymmetric unit contained four, nearly identical copies of the molecule, distinguished by chain identifiers A-D in the PDB file. In each of the four chains the fragment spanning the residues 104-112 crystallized in slightly different conformations. We added the atoms from residues 104-112 from chain A to the corresponding residues from chain B (Figure 1). Indeed, the fragment can presumably adopt both of these states. The conformers are closely intertwined, separated by only 1.4Å RMSD.

Simulated electron density data for the dual conformer was generated at different resolutions and at various occupancies. Gaussian noise with a standard deviation of 10% of the magnitude of the calculated data was added to simulate experimental errors. The temperature factors of the individual PDB structures were retained, averaging 19.0Å^2.

The algorithm returns an ensemble in excellent agreement with the actual conformations, with a good estimate of the true occupancy values and average temperature factors (see Table 2). Again the finite discretization of our sampling scheme results in ensembles that contain more than two conformations. But every returned ensemble contains two groups of very similar conformations that could be merged by a clustering algorithm.

We ran the same test again, but this time we added the true conformers to the sample set. The results presented in Table 3 show that in most cases our method returns the true conformers. In some cases, it produced more than two conformers and in all cases occupancies and temperature factors are slightly inexact. These small discrepancies seem to be caused by the Gaussian noise added to the EDM. The greater discrepancy in the results presented in Table 2 are, thus, manifestations of both discretization errors and errors due to added noise.

Furthermore, coordinate error is larger for the lower occupancy conformer. It should be noted that at an occupancy of 0.3, a Carbon atom only scatters at about

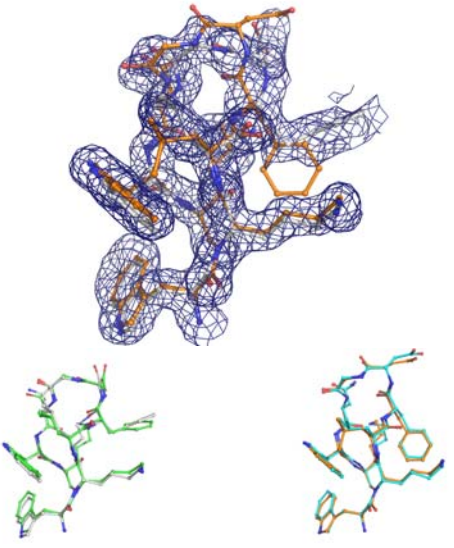


Fig. 1. Residues 104-112 of 2R4I. Top panel: Conformations from chain A and B in the EDM at 0.7/0.3 occupancy. At high contour levels, atoms from the chain at lower occupancy are no longer contained within the iso-surface. Lower panel: PDB fragment from chain A (left) in green and PDB fragment from chain B (right) in cyan together with the calculated conformers.

twice the magnitude of a Hydrogen atom. The signal of a Hydrogen atom is distinguished from the background level only at resolution levels better than 1.3 Å(i.e. $<$ 1.3 Å). At resolution levels considered here, Hydrogens are not explicitly included in PDB files.

2.1.3 Ensemble of Conformations

Solvent exposed fragments may have only weakly preferred substates. This common situation is often characterized by a blurring of the main-chain EDM and ambiguous or weak side-chain density. To emulate this situation, a collection of 20 conformations of the 8-residue loop 142-149 of 1HFC (Table 1) was generated along a coordinated motion of the loop (as shown in Figure 2(a)). The start and finish conformations are 2.7Å apart in RMSD. A 1.9Å EDM was calculated at equal occupancy (0.05) for the members of the collection.

The algorithm returned a 7-conformer ensemble, with occupancies ranging from 0.10 to 0.23. Since the algorithm only retains conformations with calculated occupancy greater than or equal to 0.1, it could not return the full collection of 20 conformations. Nevertheless, it successfully extracted the range of motion from the data (see Figure 2(b)).

We furthermore applied the loop-fitting option of RESOLVE (v2.10) [23], a widely-used crystallographic model-building algorithm, to this EDM. RESOLVE, unable to assign residue identities and side-chains, modeled a single poly-alanine loop

Table 2. Details of calculated dual conformers for loop 104-112 of 2R4I. Each row lists occupancies for the conformers (Occ), map resolution (Res, in Å), RMSD of calculated conformers to PDB conformers, the cumulative calculated occupancies for the conformers (Calc Occ), and average temperature factor of calculated conformers ($\bar{B}$calc, in Å^2). Average, observed temperature factors are 19.0Å^2.

Occ	Res	RMSD	Calc Occ	$\bar{B}$calc
0.5/0.5	1.3	0.26 0.34	0.29	24.3
		0.38 0.64 0.77 1.29	0.71	
0.5/0.5	1.5	0.32 0.64	0.36	27.4
		0.38 0.49 0.52 0.69	0.64	
0.5/0.5	1.7	0.29 0.42 0.42 0.43	0.50	25.5
		0.23 0.23 0.34	0.50	
0.6/0.4	1.3	0.29 0.40 0.64	0.53	25.5
		0.31 0.35 0.64	0.47	
0.6/0.4	1.5	0.30 0.44 0.54 0.58	0.61	25.7
		0.33 0.62	0.39	
0.6/0.4	1.7	0.23 0.24 0.35 0.60	0.58	22.0
		0.23 0.62	0.41	
0.7/0.3	1.3	0.27 0.34 0.34 0.34 0.64	0.70	24.1
		0.48	0.30	
0.7/0.3	1.5	0.33 0.33 0.40	0.64	29.0
		0.61 0.76	0.36	
0.7/0.3	1.7	0.31 0.37 0.42 0.47	0.65	21.7
		0.44 0.62	0.35	

into the EDM, shown in yellow in Figure 2(b). Analysis of the results reveals that occupancy-weighted main-chain EDM correlation coefficients of the ensemble range from 0.84 to 0.96 per residue versus 0.64 to 0.87 for the single poly-alanine conformer. Moreover, the average, occupancy-weighted temperature factor of the ensemble (15.0Å^2) is closer to the average of the 20 conformations (11.7Å^2) than the single, poly-alanine chain (35.7Å^2). Thus, our 7-conformer ensemble is a significantly improved interpretation of the data, both quantitatively and qualitatively (range of motion), over a single, averaged conformer.

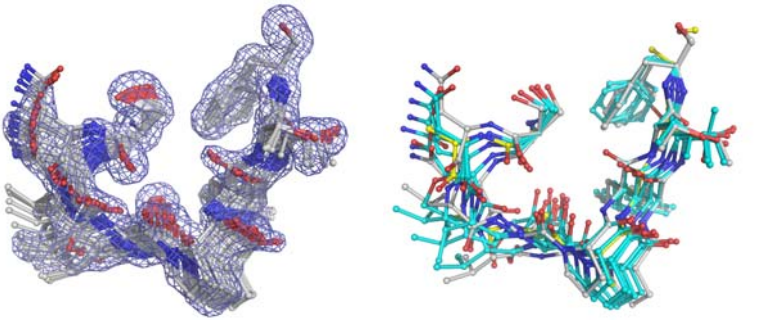
This example suggests that in general the returned ensemble of conformations should not be treated as a true physical model of the actual heterogeneity present in the crystal, but as a representation of uncertainty in atomic positions due (in part) by this heterogeneity.

2.2 Experimental Data: Modeling a Dual Conformer

A structural model for TM0755 was obtained by the JCSG from data at 1.8Å resolution. The asymmetric unit contains a dimer, with a short main-chain fragment around residue A320, and the same fragment around B320, bimodally disordered. Crystallographers had initially abandoned this fragment due to difficulty interpreting the EDM visually. A dual conformation for the fragment A316-A325, separated

Table 3. Details of calculated dual conformers for loop 104-112 of 2R4I. The true conformers were added in the sampling protocol. Each row lists occupancies for the conformers (Occ), map resolution (Res, in Å), RMSD of calculated conformers to PDB conformers, the cumulative calculated occupancies for the conformers (Calc Occ), and average temperature factor of calculated conformers ($\bar{B}$calc, in Å^2). Average, observed temperature factors are 19.0Å^2.

Occ	Res	RMSD	Calc Occ	$\bar{B}$ calc
0.5/0.5	1.3	0.00	0.47	25.2
		0.00	0.53	
0.5/0.5	1.5	0.00	0.48	22.3
		0.00	0.52	
0.5/0.5	1.7	0.00 0.29	0.49	22.0
		0.00 0.23	0.51	
0.6/0.4	1.3	0.00	0.56	20.1
		0.00	0.44	
0.6/0.4	1.5	0.00 0.54	0.61	24.7
		0.00	0.39	
0.6/0.4	1.7	0.00	0.57	23.4
		0.00	0.43	
0.7/0.3	1.3	0.00	0.65	25.9
		0.00	0.35	
0.7/0.3	1.5	0.00 0.33 0.33	0.66	29.0
		0.00 0.61	0.34	
0.7/0.3	1.7	0.00	0.65	20.8
		0.00	0.35	



(a) (b)

Fig. 2. (a) A collection of 20 snapshots of an 8 residue loop while it is transitioning between simulated start and finish conformations. (b) Ensemble of 7 conformers computed by the algorithm (cyan), together with the start and finish conformations (grey) of the simulated collection. A single conformer modeled by RESOLVE is displayed in yellow.

by 2.96Å, was obtained from semi-automated methods at 0.5/0.5 occupancy [27]. The average, occupancy-weighted temperature factor was 24.9Å^2. The structure together with the heterogeneous fragment was refined, subjected to the JCSG's quality control protocol (unpublished) and ultimately deposited in the PDB.

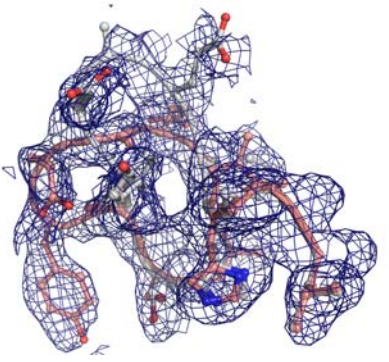


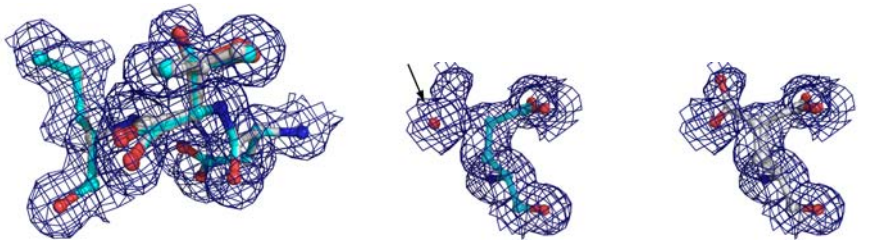
Fig. 3. Two conformations from the 5-conformer ensemble computed for the fragment A316-A325 in the experimental EDM. The conformers deviate by 0.47 Å RMSD and 0.64 Å RMSD from the conformations obtained by the JCSG. Alternate conformers are difficult if not impossible to identify and model visually in ambiguous electron density. For clarity, the main-chain is represented by a cartoon in the figure.

An experimental electron density map was calculated from diffraction images with σ_A-weighted $2mF_o - DF_c$ coefficients [21]. Our algorithm returned a 5-conformer ensemble. Two conformations in the ensemble are 0.47 and 1.24 Å RMSD away from one of the conformations obtained at JCSG with occupancies 0.15 and 0.23. The other three calculated conformers are 0.64, 0.72, and 0.82Å RMSD away from the other conformation obtained at JCSG with occupancies 0.27, 0.23, and 0.12 respectively. The average, occupancy-weighted temperature factor of the ensemble is 30.3Å^2.

This result demonstrates that our method is also highly effective with experimental data which, in contrast to data with simulated measurement errors, may contain substantial phase angle errors. Automatic identification of multi-conformers will greatly enhance the structure determination process.

2.3 Experimental Data: Modeling Alternate Side Chain Conformations

Locally, side-chains too regularly adopt alternate conformations, accommodated by subtle changes in the main-chain [6]. At present, modeling alternate side-chains onto a known main-chain in the final stages of the structure determination process is time-consuming and subject to individual preferences. To assess the value of our algorithm for a high-throughput structure determination pipeline such as the JCSG's, it was modified to model alternate side-chain conformations onto a known main-chain. At a fixed position in the protein chain, trial positions for the C_β atom are generated, and the entire residue is repositioned by adjusting flanking dihedral angles. For each trial C_β position, neighborhoods of rotamers are sampled to obtain a



(a) (b) (c)

Fig. 4. (a) Residue 36THR from the A chain in 2NLV. The PDB model is shown in cyan (2 conformations at 0.5/0.5 occupancy), and our model is shown in grey (3 conformations at 0.31, 0.35, and 0.36 occupancy). Note that the carbonyl oxygen shifts considerably to accommodate an alternate conformation. The side-chain EDM correlation coefficient improved from 0.77 to 0.81. (b) Residue 81GLU as modeled in the PDB conformation, and (c) as modeled by our algorithm. Observe that the PDB conformation mistakingly modeled a water molecule at full occupancy at the position of a carboxyl oxygen, a common mistake. Our alternate side-chain is modeled at 0.33 occupancy.

large set of candidate conformations. This set is then subjected to a selection step to obtain occupancy values. Finally, the coordinates are refined with a standard crystallographic refinement suite [1].

A structural model for Xisl protein-like solved to 1.3 Å resolution (PDB ID 2NLV) was used to test the procedure. The protein is 112 residues in length, and was deposited in the PDB with 20 residues of the A-chain in alternate conformations. The algorithm successfully identified and modeled 85% of residues with alternate conformations, see Figure 4. The side-chain conformations that were not found were outside the sample set. Additionally, 12 multi-conformer alternatives for single-conformer residues were identified for which the data fit improved substantially, see Figure 4.

3 Method

Our goal is to compute an ensemble of conformations, the occupancy of each conformation, and the temperature factor of each atom in every conformation, that together optimally represent the data in an input EDM E of a protein fragment.

One approach – let us call it initialize-optimize – consists of formulating this problem as an optimization problem:

1. Pick an ensemble of k initial conformations, along with their occupancies and temperature factors.
2. Compute the simulated EDM that corresponds to this ensemble.
3. Iteratively modify the k conformations, their occupancies, and the temperature factors to minimize the difference between the experimental and the simulated EDMs.

We actually tried this approach, but even on dual-conformer examples the number of parameters to optimize is huge and the optimization process (Step 3) gets easily trapped into local minima. Monte-Carlo methods with simulated annealing protocols were unable to handle these issues.

This led us to develop a completely different approach, which we call sample-select. Instead of incrementally modifying conformations, we first sample a very large set of conformations and then select the best ensemble from this set. More precisely, our method alternates two steps, SAMPLE and SELECT:

1. SAMPLE samples a large set Q of conformations (and the temperature factors of the atoms in each conformation) that is highly likely to contain a subset S representing E well.
2. SELECT simultaneously identifies this subset S and computes the occupancy factor of each conformation in S.

The space sampled by SAMPLE has high dimensionality, so each run of SAMPLE uses the conformation subset selected at the previous iteration to sample a new set of candidate conformations, which in turn is submitted to SELECT. The core of our method is an efficient linear-programming algorithm that is able to select pertinent ensembles from very large sets of sampled conformations. We first describe this algorithm.

3.1 Selection Step

SELECT is handed a large set $Q = \{q_1, \ldots, q_N\}$ of N conformations, together with a vector t_i specifying the temperature factor of each atom in every conformation q_i. It identifies the subset S of conformations that *collectively* provides the best explanation for the input EDM E, over all possible subsets of Q.

Let G be the grid over which E is defined. Let E_i be the simulated EDM that corresponds to the configuration q_i with the temperature factors in t_i. Let $E(p)$ and $E_i(p)$ denote the values of E and E_i, respectively, at point $p \in G$. The value at p of the EDM that corresponds to $Q = \{q_1, ..., q_N\}$ with occupancies $\alpha_1, \ldots, \alpha_N$ is $\sum_i \alpha_i E_i(p)$. SELECT minimizes the L_1 difference between E and this EDM. Since each $E_i(p)$ is constant, this amounts to solving the following linear problem (LP):

$$\begin{array}{ll} \text{Minimize} & \sum_{p \in G} \mid E(p) - \sum_i \alpha_i E_i(p) \mid \\ \text{such that} & \alpha_i \geq 0, \text{ for } i = 1, \ldots, N \\ & \sum_i \alpha_i = 1. \end{array}$$

The solution is the vector of optimal values for α_i, $i = 1, \ldots, N$. SELECT retains only the conformations q_i whose occupancies are greater than a given threshold (set to 0.1 in our implementation). It returns the set S of retained conformations with occupancies re-normalized to sum up to 1. We use Coin-OR libraries [14] to solve the above LP.

3.2 Conformation Sampling

The goal of SAMPLE is to generate a set $Q = \{q_1, \ldots, q_N\}$ of candidate conformations, together with temperature-factor vectors $t_1, \ldots, t_N$, such that a subset S of Q (with suitable occupancies) provides an optimal explanation of the EDM E. Each SAMPLE step uses the outcome of the previous SELECT step and samples a distinct subspace of reasonably small dimensionality.

Let $n > 3$ be the number of residues in the fragment. We fix bond lengths and dihedral angles ω around peptide bonds to their canonical values. This leads us to treat the fragment as a kinematic linkage [8] whose degrees of freedoms are the dihedral ϕ and ψ angles around N-C_α and C_α-C bonds, the bond angles in the main chain, and the χ angles in the side-chains. We divide the fragment into a front and a back half, each with $p = \lceil \frac{n}{2} \rceil$ residues. We first incrementally build conformations of these two halves. Then, we connect them using an inverse kinematics (IK) algorithm.

We describe the various steps in more detail below. However, it should be noted that there are many possible variants, some of which might work equally well. The key idea is to consider fractions of the front and back halves of increasing size, so that the number of conformations sampled by each SAMPLE step can be handled by the next SELECT operation.

(a) Sampling the main chains of the two halves. Let us temporarily ignore side-chains and temperature factors. We incrementally build candidate partial conformations of the front half's main chain by sampling one ϕ or ψ angle at a time, starting from the N terminus. We sample the first ϕ angle at some uniform resolution ε (set to 2 degrees in our implementation). We also sample the bond angle centered at the N atom preceding this ϕ angle at the same resolution ε in the 12-degree interval around its corresponding Engh-Huber value. We thus obtain a set of $6 \times 2\pi/\varepsilon$ candidate positions for the following C_β and C atoms, on which we run SELECT. Let k_1 be the number of *partial* conformations retained by SELECT. Next, we sample in the same ways the following dihedral angle (a ψ angle) and the two following bond angles centered at C_α and C atoms. We thus get a set of $12 \times 2\pi/\varepsilon \times k_1$ candidate positions for the following O, N, and C_α atoms. We run SELECT on this set and obtain an ensemble of size k_2.

At this point, we re-sample the two ϕ and ψ angles at a finer resolution (0.5 degrees) in small neighborhoods (± 1 degree) of their values in the ensemble of size k_2. This re-sampling step yields an expanded set of candidate conformations to which we apply SELECT. We proceed in the same way with the remaining $p - 1$ residues in the front half's main chain.

The same procedure is applied in reverse to the back half, starting from its C terminus.

(b) Inserting side-chains. Immediately after a pair of consecutive ϕ and ψ angles have been re-sampled, the side-chain of the residue containing those two angles is

inserted. We use a rotamer library [18] to obtain the values of the χ angles. Adding the side-chain multiplies the number of partial conformations of the front half by the number of rotamers for the side-chain. We apply SELECT to this new set. The same procedure is applied to the back half.

(c) Assigning temperature factors. Temperature factors are assigned whenever a ϕ or ψ angle is sampled or a side-chain is inserted. Their values are taken from a finite set T input by the user. However, assigning a distinct temperature factor to every atom would quickly lead to large sets of candidate conformations. So, we define groups of atoms that are assigned the same temperature factors. The C_β and C atoms following a ϕ angle forms one group, so do the O, N, and C_α atoms following a ψ angle and the atoms in a side-chain.

Consider the case where we sample a ϕ angle in the front half. As described in paragraph (a), this gives a number of candidate conformations for the following C_β and C atoms. We pair each of these conformations with a distinct temperature factor from T. Similarly, when we insert a side-chain, we pair each rotamer with a distinct temperature factor from T.

(d) Connecting the front and back halves. We enumerate all pairs of conformations of the fragment's front and back halves computed as above. For each pair, complete closed conformations of the fragment's main chain are obtained by computing six dihedral angles using an analytical IK algorithm [5]. More precisely, for each pair, we consider every three consecutive residues such that at least one belongs to the front half and another one to the back half, and we re-compute the ϕ and ψ angles in those residues using the IK algorithm, so that the fragment's main chain gets perfectly closed. The side-chain conformations and temperature factors for each of these residues are set as in either the front or back half conformation.

We collect all the closed conformations into a candidate set, on which we run SELECT. The result is the final conformation ensemble built by our method. If desired, a clustering algorithm can be run on this ensemble to merge conformations that are pairwise very close.

The above method sometimes eliminates a pertinent partial conformation. This is due to the fact that partial conformations are retained based on their fit with only a subset of the EDM. So, a SELECT step might retain one conformation and discard another based on this *local* fit, while the inverse result could have been obtained if larger fractions of the fragment had been considered. Unfortunately, when a pertinent partial conformation has been discarded, it cannot be recovered later. So, to reduce the risk of eliminating a pertinent partial conformation, we retain a greater number of partial conformations at each selection step. This is done as follows. Let m be the size of the set of conformations given to a selection step and m' the size of the ensemble retained by SELECT. We run SELECT again on the remaining $m - m'$ conformations, and we repeat this operation until a pre-specified number of conformations have been obtained.

4 Conclusion

This paper presents a new method to model structural heterogeneity in an EDM by computing an ensemble of conformations, with occupancies and temperature factors, that collectively provide a near-optimal explanation of the EDM. Instead of being based on an initialize-optimize approach that is classical in single-conformer programs, our method is based on a sample-select approach that adaptively alternates sampling and selection steps. We successfully tested our method on both simulated and experimental EDMs of protein fragments ranging from 4 to 12 residues in length and side-chains.

Modeling structural heterogeneity from EDMs is of major importance and may have a major impact on the way protein models are stored in the Protein Data Bank. However, our work is only a step in that direction. Several issues must still be investigated.

We need to further analyze errors caused by the finite resolution and the locality of our sampling protocol. Experiments with simulated EDMs show that if we include the correct conformations in the set of sampled conformations submitted to a SELECT step, this step reliably returns the exact ensemble of conformers (see Table 3). This suggests that an adaptive sampling protocol could generate better results than our current protocol.

Although is some cases the ensemble returned by our method is a physical model of the actual heterogeneity present in the crystal, this is not always the case. The example in Section 2.1.3 is a good counter-example. In general, we can only say that an ensemble returned by our method is a macromolecular representation that near-optimally fits the EDM, and thus also represents uncertainty in atomic positions. Additional physicochemical evidence is usually required to determine if this outcome is a physical model of the conformational states present in the crystal.

Finally, to be really useful and actually used by crystallographers, our method will have to be integrated into existing suites of modeling software.

Acknowledgements. This work was partially supported by NSF grant DMS-0443939. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF. Test structures used in this work were solved and deposited as part of the JCSG pipeline (www.jcsg.org). The authors thank all members of the JCSG for their assistance in providing data. The JCSG is funded by NIH Protein Structure Initiative grants P50 GM62411, U54 GM074898. This research made use of the BioX cluster that has been partially funded by NSF award CNS-0619926.

References

1. Afonine, P.V., Kunstleve, R.W.G., Adams, P.D.: The phenix refinement framework. CCP4 newsletter 42 (2005)
2. Berman, H.M., Henrick, K., Nakamura, H., Arnold, E.: Reply to: Is one solution good enough? Nature Structural and Molecular Biology 13, 185 (2006)

3. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. Nucleic Acids Research, 235–242 (2000)
4. Burling, F.T., Weis, W.I., Flaherty, K.M., Brunger, A.T.: Direct observation of protein solvation and discrete disorder with experimental crystallographic phases. Science 271(5245), 72–77 (1996)
5. Coutsias, E.A., Seok, C., Jacobson, M.P., Dill, K.A.: A kinematic view of loop closure. Journal of Computational Chemistry 25(4), 510–528 (2004)
6. Davis, I.W., Arendall, W.B., Richardson, D.C., Richardson, J.S.: The backrub motion: How protein backbone shrugs when a sidechain dances. Structure 14, 265–274 (2006)
7. DePristo, M.A., de Bakker, P.I., Blundell, T.L.: Heterogeneity and inaccuracy in protein structures solved by x-ray crystallography. Structure 12, 831–838 (2004)
8. Dhanik, A., Yao, P., Marz, N., Propper, R., Kou, C., Liu, G., van den Bedem, H., Latombe, J.C.: Efficient algorithms to explore conformation spaces of flexible protein loops. In: Giancarlo, R., Hannenhalli, S. (eds.) WABI 2007. LNCS (LNBI), vol. 4645, pp. 265–276. Springer, Heidelberg (2007)
9. DiMaio, F., Kondrashov, D.A., Bitto, E., Soni, A., Bingman, C.A., Phillips Jr., G.N., Shavlik, J.W.: Creating protein models from electron-density maps using particle-filtering methods. Bioinformatics 23(21), 2851–2858 (2007)
10. DiMaio, F., Shavlik, J., Phillips, G.N.: A probabilistic approach to protein backbone tracing in electron density maps. Bioinformatics 22(14), e81–e89 (2006)
11. Drenth, J.: Principles of protein X-ray crystallography. Springer, New York (1999)
12. Frauenfelder, H., Sligar, S.G., Wolynes, P.G.: The energy landscapes and motions of proteins. Science 254(5038), 1598–1603 (1991)
13. Furnham, N., Blundell, T.L., DePristo, M.A., Terwilliger, T.C.: Is one solution good enough? Nature Structure Molecular Biology 13(3), 184–185 (2006)
14. Heimer, R.L.: The common optimization interface for operations research: promoting open-source software in the operations research community. IBM Journal of Research and Development 47(1), 57–66 (2003)
15. Ioerger, T., Sacchettini, J.: The textal system: Artificial intelligence techniques for automated protein model building. In: Carter, C.W., Sweet, R.M. (eds.) Methods in Enzymology, pp. 244–270. Springer, Heidelberg (2003)
16. Kuriyan, J., Petsko, G.A., Levy, R.M., Karplus, M.: Effect of anisotropy and anharmonicity on protein crystallographic refinement: An evaluation by molecular dynamics. Proteins: Structure, Function, and Bioinformatics 190, 227–254 (1986)
17. Levin, E.J., Kondrashov, D.A., Wesenberg, G.E., Phillips Jr., G.N.: Ensemble refinement of protein crystal structures: validation and application. Structure 15, 1040–1052 (2007)
18. Lovell, S.C., Word, J.M., Richardson, J.S., Richardson, D.C.: The penultimate rotamer library. Proteins: Structure, Function, and Bioinformatics 40(3), 389–408 (2000)
19. Okazaki, K.-I., Koga, N., Takada, S., Onuchic, J.N., Wolynes, P.G.: Multiple-basin energy landscapes for large-amplitude conformational motions of proteins: Structure-based molecular dynamics simulations. PNAS 103, 11844–11849 (2006)
20. Perrakis, A., Sixma, T.K., Wilson, K.S., Lamzin, V.S.: wARP: Improvement and extension of crystallographic phases by weighted averaging of multiple-refined dummy atomic models. Acta Crystallographica D 53, 448–455 (1997)
21. Read, R.J.: Improved fourier coefficients for maps using phases from partial structures with errors. Acta Crystallographica A 42, 140–149 (1986)
22. Schomaker, V., Trueblood, K.N.: On the rigid-body motion of molecules in crystals. Acta Crystallographica B 24, 63 (1968)

23. Terwilliger, T.C.: Automated main-chain model building by template matching and iterative fragment extension. Acta Crystallographica D 59, 38–44 (2003)
24. Terwilliger, T.C.: Improving macromolecular atomic models at moderate resolution by automated iterative model building, statistical density modification and refinement. Acta Crystallographica D 59, 1174–1182 (2003)
25. Terwilliger, T.C., Grosse-Kunstleve, R.W., Afonine, P.V., Adams, P.D., Moriarty, N.W., Zwart, P., Read, R.J., Turk, D., Hung, L.-W.: Interpretation of ensembles created by multiple iterative rebuilding of macromolecular models. Acta Crystallographica D 63, 597–610 (2007)
26. Tosatto, S.C.E., Bindewald, E., Hesser, J., Manner, R.: A divide and conquer approach to fast loop modeling. Protein Engineering 15(4), 279–286 (2002)
27. van den Bedem, H., Lotan, I., Latombe, J.C., Deacon, A.M.: Real-space protein-model completion: an inverse-kinematics approach. Acta Crystallographica D 61, 2–13 (2005)
28. Vitkup, D., Ringe, D., Karplus, M., Petsko, G.A.: Why protein R-factors are so large: A self-consistent analysis. Proteins: Structure, Function, and Genetics 46, 345–354 (2002)
29. Wilson, M.A., Brunger, A.T.: The 1.0 Å crystal structure of Ca2+-bound calmodulin: an analysis of disorder and implications for functionally relevant plasticity. Journal of Molecular Biology 301(5), 1237–1256 (2000)

Minimum Resource Characterization of Biochemical Analyses for Digital Microfluidic Biochip Design

Lingzhi Luo and Srinivas Akella

Abstract. Digital microfluidic systems (DMFS) are a class of lab-on-a-chip systems that manipulate individual droplets of chemicals on an array of electrodes. The biochemical analyses are performed by repeatedly moving, mixing, and splitting droplets on the electrodes. In this paper, we characterize the tree structure of biochemical analyses and identify their minimum resource requirements, towards the design of cost and space-efficient biochips. Mixers and storage units are two primary functional resources on a DMFS biochip; mixers mix and split droplets while storage units store droplets on the chip for subsequent processing. Additional DMFS resources include input and output units and transportation paths. We present an algorithm to compute, for a given number of mixers M, the minimum number of storage units $f(M)$ for an input analysis using its tree structure, and design a corresponding scheduling algorithm to perform the analysis. We characterize the variation of the M-depth of a tree (i.e., its minimum number of storage units $f(M)$) with M, and use it to calculate the minimum total size (the number of electrodes) of mixers and storage units. We prove that the smallest chip for an arbitrary analysis uses one mixer and $f(1)$ storage units. Finally, we demonstrate our results on two example biochemical analyses and design the smallest chip for a biochemical analysis with a complete tree structure of depth 4. These are the first results on the least resource requirements of DMFS for biochemical analyses, and can be used for the design and selection of chips for arbitrary biochemical analyses.

1 Introduction

Low-cost, portable lab-on-a-chip systems capable of rapid automated biochemical analysis can impact a wide variety of applications including biological research,

Lingzhi Luo
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: lingzhil@cs.cmu.edu

Srinivas Akella
Dept. of Computer Science, University of North Carolina, Charlotte, NC 28223, USA
e-mail: sakella@uncc.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 567–581.
springerlink.com

point-of-care diagnostics, and biochemical sensing [5, 18, 2, 13]. *Digital microfluidic systems* (DMFS) are an emerging class of lab-on-a-chip systems that manipulate discrete droplets. A digital microfluidic system manipulates individual droplets of chemicals on an array of electrodes by using electrowetting (or dielectrophoresis). We focus on microfluidic systems that manipulate droplets by electrowetting [11]. The chemical analysis is performed by repeatedly moving, mixing, and splitting droplets on the electrodes (Figure 1).

The ability of DMFS devices to control discrete droplets enables complex analysis operations to be performed in a flexible manner. There are several classes of resources (i.e., functional components consisting of electrodes) in a DMFS: mixers, storage units, input/output units, and transportation paths. Figure 2 illustrates their functions. We focus on performing biochemical analyses in batch mode, where the goal is to produce one droplet of the final product. We previously [10] developed algorithms to compute minimum time schedules based on the tree structure of chemical analyses. Here we focus on determining minimum resources and use it to design (or select) the smallest chip for a given analysis.

Biochip users want versatile, yet low cost systems. Hence it is important to identify the class of biochemical analyses a given chip can perform, and further, the smallest chip that can perform a given set of analyses. In this paper, we characterize the structure of different biochemical analyses and classify them according to their least resource requirements. Since we can reduce the fabrication cost of a DMFS chip by reducing the number of electrodes, this work is motivated by two practical questions: For any biochemical analysis, how do we identify the (smallest) biochip with sufficient number of resources to perform it? For a given biochip, what kind of biochemical analyses can be performed on it? We assume that the resources (e.g., mixers and storage units) are fixed at the start of the analysis and do not change over the course of the analysis, and that once processing of a subtree begins, it is completed before a sibling subtree is processed. First, we compute the least resource requirements based on the tree structure of biochemical analyses and design corresponding algorithms to perform them. Then we define the M-depth of a tree to describe such resource requirements and characterize the variation of the M-depth with the number of mixers M. This is the first work on DMFS to compute

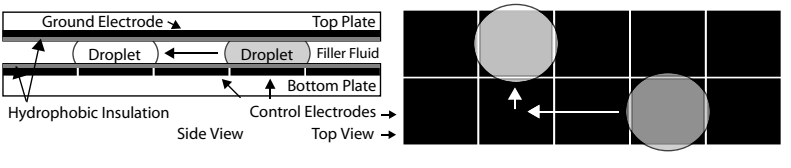


Fig. 1. Droplets on an electrowetting array (side and top views). The droplets are in a medium (usually oil or air) between two glass plates. The gray and white droplets represent the same droplet in initial and destination positions. A droplet moves to a neighboring electrode when that electrode is activated; the electrode is turned off when the droplet has completed its motion. Based on [12].

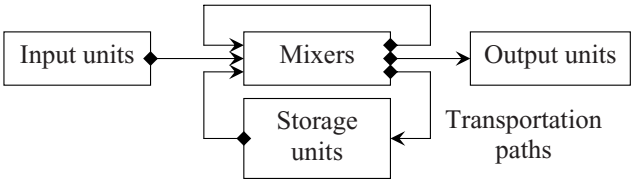


Fig. 2. Five classes of functional resources for DMFS biochips. Transportation paths (shown as arrows starting from diamonds) are used to move droplets from one resource to another. Input units are used to input droplets. Mixers are used to mix two different droplets and split their mixture to produce two new droplets. In batch mode, one of the new droplets will be used and the other will be discarded as a waste droplet at an output unit. The new droplet may be kept idle in a storage unit for some time, stay at the mixer, or be transported to a new mixer for a subsequent mixing. The droplet of final product will be transported to an output unit.

the least resource requirements for biochemical analyses and it can be used to guide the selection and design of chips for arbitrary biochemical analyses.

2 Related Work

Resource Requirements: Su and Chakrabarty presented architecture-level synthesis and geometry synthesis of biochips[14, 15]. They used acyclic sequence graphs to represent the reactions and developed techniques in operation scheduling, resource binding, and module placement. In their papers resource constraints are given in advance by the size of chips or the number of mixers and storage units. However there is no general guideline for selecting chips with proper number of mixers and storage units for biochemical analyses. Griffith and Akella [7] explored the relationship between the number of mixing units and the highest stable input rates in the system, where increasing the number of mixing units permits the system to be stable at a higher input rate and the effectiveness of maintaining system stability by increasing the number of mixers decreases. A variety of modifications to the resources were made to gauge the effects on the stability of the system [8]. The work in [7, 8] is based on simulations while in this paper we are formally analyzing the resource requirements for analyses based on the underlying tree structure.

Layout Design: Layout design maps the functional units such as mixers, storage units, and routing paths to the underlying hardware. Griffith and Akella presented a semi-automated method to generate the array layout in terms of components [7]. Su and Chakrabarty developed an online reconfigurable technique to bypass the fault unit cells in the microfluidic biochips [16].

Scheduling Algorithms: Scheduling algorithms optimize the system performance by properly allocating tasks to resource. Kwok and Ahmad studied the static scheduling of a program on a multiprocessor system to minimize program completion time in parallel processing [9]. Since the general problem is NP-complete, they compared heuristic scheduling algorithms. In contrast to the general problem in parallel computing, the multiple-task reactions in DMFS have certain kinds of

precedence, which can be represented by a full binary tree. Ding et al. [4] and Su and Chakrabarty [14] represent the DMFS reactions using data-flow directed graph and consider scheduling using Integer Linear Programming (ILP). They solve the problem by general ILP solvers and heuristic algorithms without exploiting the structure of DMFS reactions.

Routing: Böhringer modeled the routing problem in DMFS as a multi-robot cooperation problem and used a prioritized A^* search algorithm to generate the optimal plan for droplets [1]. Griffith and Akella presented a general-purpose DMFS and designed routing algorithm based on Dijkstra's algorithm [7, 8]. Su, Hwang and Chakrabarty proposed a two-stage routing method to minimize the number of electrodes used for droplet routing while considering the resource constraint [17].

3 Model and Problem Formulation

3.1 Full Binary Tree Model

The (biochemical) "analysis graph" provides a representation of the operations of a DMFS. It is a directed graph, with an input node for each droplet type entering the system, an output node for each droplet type leaving the system, and a mix node for each mixing operation performed in the system. The nodes are connected based on the droplet types they require and produce. A mixing operation $A + B \rightarrow C$ means reagents A and B are mixed to produce C. The mixing operations during a DNA PCR analysis are described as follows.

$A + B \rightarrow C; C + D \rightarrow E; E + F \rightarrow G;$

$H + I \rightarrow J; J + K \rightarrow L; L + M \rightarrow N; N + G \rightarrow O.$

To model the dependencies among different mixing operations, we introduce a full binary tree structure. A *full binary tree* is a binary tree in which every node has either zero or two child nodes. A *complete binary tree* is a full binary tree in which all leaf nodes are at the same level. (Levels of nodes are defined in a recursive way as follows: the level of the root node is zero; the level of any other node equals one plus that of its parent node.) Given an analysis R, we construct a full binary tree T as follows. Every reagent in R is represented by a node in T; source reagents, which can be directly fetched from the reservoirs, are represented by leaf nodes in T, and intermediate reagents, which are produced by mixing, are represented by parent nodes of those nodes from which they can be produced. So the analysis R is represented by T, where the final product of R is represented by the root node of T. The representation of the PCR analysis by a full binary tree is shown in Figure 3. In the following sections, we will analyze biochemical reactions and design algorithms based on the full binary tree structure.

3.2 Problem Formulation

The problems we want to solve in this paper are: *Given a biochemical analysis, what are the minimum requirements of resources (e.g., the number and size of*

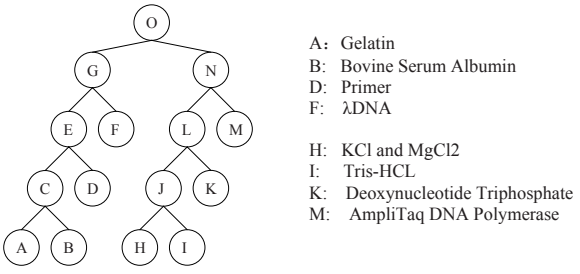


Fig. 3. Representation of a DNA PCR analysis by a full binary tree of depth 4.

mixers, storage units, input/output units and transportation paths) to perform it? How can we design the scheduling algorithm so that the analysis can be performed with the minimum resources? The number of input and output units is determined by the given biochemical analysis. Transportation paths are constructed so that other function units (the input units, output units, mixers and storage units) are connected. We will first focus on the resource requirements of mixers and storage units, and then consider the requirements of other resources as well in Section 5.

3.2.1 Pipelining

For biochemical analyses on a DMFS, the procedure of obtaining a droplet can be divided into several operations: input source droplets, transport droplets, mix (and split) droplets, and output waste droplets. After applying pipelining techniques [10], different operations can be overlapped. Also, since the transportation time is negligible compared to the mixing time [3, 6], we ignore the transportation time in subsequent scheduling. So when discussing the minimum resource requirements, we can just focus on the mixing scheduling.

3.2.2 Mixers and Storage Units

The number of mixers limits how many mixing operations we can perform in a time slot. The number of storage units gives a constraint for droplets that have been produced but cannot immediately be mixed. Figure 4 shows how mixers and storage units are used during the progress of a biochemical analysis.

3.2.3 Scheduling Representation

Each scheduling step can be regarded as the transition from one full binary tree to another where some sibling nodes are removed (see Figure 4). So the whole schedule *Sch* of a biochemical analysis can be represented by a series of full binary trees, starting from the initial full binary tree and ending at the root node, and the transitions between consecutive trees correspond to removing selected sibling nodes.

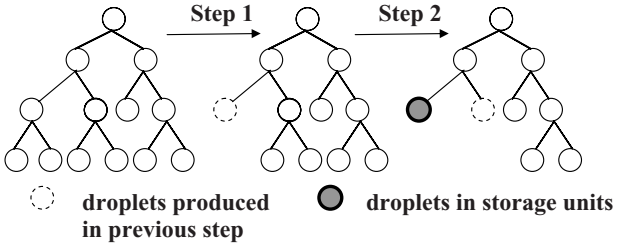


Fig. 4. Progress of a biochemical analysis on a chip with a single mixer. The intermediate reagent produced in the first step is stored in a storage unit since it is not involved in the mixing in the second step.

For a schedule *Sch*, let the number of mixers be M and the number of storage units be S. We can formulate the scheduling problem as the following two questions:

Given an initial full binary tree T, (1) What is the schedule Sch in which S is minimized given M? (2) What is the schedule Sch in which the resource requirements (i.e., the number of electrodes for M mixers, S storage units, input/output units and transportation paths) are minimized?

4 Minimizing the Number of Storage Units with a Given Number of Mixers

In our previous work [10], we analyzed two extreme cases of resource requirements: with just one mixer and with zero storage units. In the first case, we calculated the minimum number of storage units for a given analysis. In the second case, we computed the minimum number of mixers required, and proved the conclusion below.

Lemma 4.1. *[10] The sufficient and necessary condition for performing all the mixing operations without storage units is to use* $\max_{i=0}^{K} \frac{N_i}{2}$ *mixers, where K is the depth of the tree,* N_i *is the number of nodes at the ith level.*

In this paper, we characterize the relationship between the number of mixers M and number of storage units S required for a biochemical analysis based on its tree structure, and present a scheduling algorithm to handle this general case.

In this section, we first compute the minimum number of storage units $f(M)$ and design the scheduling algorithm to perform the biochemical analysis using M mixers and $f(M)$ storage units; second, we define $f(M)$ as the M-depth of the tree structure and give an equation to compute the M-depth for complete trees; third, we characterize the variation with M of the M-depth of the tree. So for any analysis, we can find out whether it can be performed using M mixers and S storage units by computing the M-depth of its tree representation. The analysis is feasible if $f(M) \leq S$, and infeasible otherwise. If two analyses have tree structures with the same M-depth, we say they have the same resource requirements.

4.1 Algorithm Design

Theorem 4.1. *Algorithm 1 returns the minimum number of storage units for a reaction using at most M mixers; Algorithm 2 outputs the corresponding schedule in the order of execution.*

Proof. Suppose $f_M(T)$ is the minimum number of storage units for the analysis tree T using at most M mixers. First we prove the first part of the theorem. According to Lemma 4.1, if the maximum number of nodes at the same level is greater than $2M$, at least one storage unit is required. Since the algorithm scans nodes from bottom up, each node first satisfying this condition needs one storage unit. (Base step.)

If $T.root$ is not a leaf node, obviously, $f_M(T) \geq \max(f_M(T.left), f_M(T.right))$.

If $f_M(T.left) \neq f_M(T.right)$, without loss of generality assume $f_M(T.left) > f_M(T.right)$. Schedule as follows: First, perform all mixing operations in the left subtree, when we need $f_M(T.left)$ storage units. Then perform those in the right subtree, when we need $f_M(T.right)$ storage units for the nodes in the right subtree and one for the $T.left$ node. So $f_M(T) = \max(f_M(T.left), f_M(T.right))$.

If $f_M(T.left) = f_M(T.right)$, we show that $f_M(T) = f_M(T.left) + 1$. Schedule all mixing operations in the left subtree, and then the right subtree. We need $f_M(T.right) + 1$ storage units as discussed above, which is equal to $f_M(T.left) + 1$.

Algorithm 1. Min-Storage-Units-M-Mixers

Input: T, M // the binary tree and the number of mixers
Output: f_M // the number of required storage units
while Scan nodes level by level from bottom up: **do**
  for any scanned node i **do**
    $i.f_M = 0$ // *storage units to produce i*
    if $i.left.f_M == 0$ and $i.right.f_M == 0$ **then**
      // *zero storage units to produce both child nodes*
      Scan the subtree rooted at i, set $i.maxnode$ as the maximum number of nodes at any level in the subtree
      if $i.maxnode > 2M$ **then**
        $i.f_M = 1$
      end if
    else
      // *at least one child node needs storage units*
      if $i.left.f_M == i.right.f_M$ **then**
        $i.f_M = i.left.f_M + 1$
      else
        $i.f_M = \max\{i.left.f_M, i.right.f_M\}$
      end if
    end if
  end for
end while
return $T.root.f_M$

So $f_M(T) \leq f_M(T.left)+1$. If $f_M(T) < f_M(T.left)+1$, then two conditions should be satisfied: First, in the time slot when $f_M(T.left)$ storage units are used for the left subtree, no mixing operations in the right subtree have been performed. Second, in the time slot when $f_M(T.right)$ storage units are used for the right subtree, no mixing operations in the left subtree have been performed. Obviously, they cannot be satisfied at the same time. By contradiction, we conclude $f_M(T) = f_M(T.left)+1$. The induction step is also correct. So the algorithm returns $f_M(T)$, the optimal value.

We now show that the second part of the theorem is correct. Since Algorithm 2 outputs the mixing operations in child-node-first order, the mixing precedence rule is satisfied. Also, the algorithm traces back through $T.root.f_M$ for all subtrees using the recurrence in Algorithm 1, so it outputs the corresponding scheduling results. ■

The complexity of both algorithms is linear in number of nodes in the tree structure.

4.2 *M*-Depth of Tree Structure

From the algorithm in the last section, we see that with M mixers, the minimum number of storage units $f(M)$ for a biochemical analysis only depends on its tree

Algorithm 2. M-Mixer-Scheduling

```
Input: T, i // the binary tree and the time slot index (1 for the initial tree)
Output: F // Schedule, global variable initialized as ∅
if T has only a root node then
    return
else
    if T.root.f_M = 0 then
        // finish mixing operations at one level each time slot
        for all levels of T from bottom up do
            F = F ⋃ {Schedule all mixing operations at the same level in time slot i}
            i = i + 1
        end for
    else
        // first produce the child node requiring more storage units
        if T.root.left.f_M > T.root.right.f_M then
            M-Mixer-Scheduling(T.left, i)
            M-Mixer-Scheduling(T.right, i)
        else
            M-Mixer-Scheduling(T.right, i)
            M-Mixer-Scheduling(T.left, i)
        end if
        F = F ⋃ {Schedule the mixing operation to get T.root in time slot i}
        i = i + 1
    end if
end if
return F
```

structure. Thus we can define $f(M)$ as the M-depth of the tree structure corresponding to M mixers. From Lemma 4.1, we can easily see that:

Lemma 4.2. *Given an analysis tree, $f(M) = 0$ if and only if $M \geq \frac{T.root.maxnode}{2}$.*

For complete trees, we derive an equation to compute M-depth as follows.

Theorem 4.2. *For a complete tree with depth D, its M-depth can be computed:*

$$f(M) = D - (\lfloor log_2 M \rfloor + 1) \tag{1}$$

Proof. Suppose in a tree with depth D, we define the height of a node as D minus its level. According to Lemma 4.2, in a complete tree, any node i with $i.f_M = 0$ must satisfy that $i.maxnode \leq 2M$ and thus has a height at most $\lfloor log_2 2M \rfloor$. And for nodes with height more than $\lfloor log_2 2M \rfloor$, increasing height by 1 will increase $i.f_M$ by 1. So

$$f(M) = T.root.f_M = T.root.height - \lfloor log_2 2M \rfloor$$

$$= D - \lfloor log_2 2M \rfloor = D - (\lfloor log_2 M \rfloor + 1). \qquad \blacksquare$$

Corollary 4.1. *For a binary analysis tree with depth D, its M-depth must satisfy:*

$$f(M) \leq D - (\lfloor log_2 M \rfloor + 1) \tag{2}$$

4.3 Variation of M-Depth with M for a Tree

We first examine the non-increasing property of the M-depth, $f(M)$. For a given biochemical analysis tree, according to Lemma 4.2, if $f(M) = 0$, $f(M+1) = 0$. There exist nodes where $i.f_M > 0$, but $i.f_{M+1} = 0$. Considering the same recurrence as in Algorithm 1, for each node i, $i.f_M \geq i.f_{M+1}$. So $f(M) = T.root.f_M \geq T.root.f_{M+1} = f(M+1)$. That is, with more mixers, we may reduce the minimum number of storage units required.

$$f(M+1) \leq f(M) \tag{3}$$

Theorem 4.3

$$f(M+1) \geq f(M) - 1 \tag{4}$$

Proof. Use mathematical induction. Base step is trivial: for a leaf node, $f(M+1) = f(M) = 0$. Now for a node i with left and right child nodes, suppose $f(M)$ is M-depth of the tree rooted at i, $f_L(M)$ and $f_R(M)$ are M-depths of its left subtree and right subtree respectively. According to the induction hypothesis,

$$f_L(M+1) \geq f_L(M) - 1, f_R(M+1) \geq f_R(M) - 1 \tag{5}$$

There are three possible cases below:

1. $f_L(M) \neq f_R(M)$: Without loss of generality, assume $f_L(M) > f_R(M)$. $f(M) = f_L(M)$. So $f(M+1) \geq f_L(M+1) \geq f_L(M) - 1$ (Inequality 5)
   so $f(M+1) \geq f(M) - 1$.

2. $f_L(M) = f_R(M) = 0$: $f(M) \leq 1$. $f(M+1) \geq 0 \geq f(M) - 1$.
3. $f_L(M) = f_R(M) \neq 0$: According to Theorem 4.1, $f(M) = f_L(M) + 1$

- $f_L(M) = f_R(M) \geq 2$:

$$f_L(M+1) \geq f_L(M) - 1 \geq 1\ ,\ f_R(M+1) \geq f_R(M) - 1 \geq 1 \tag{6}$$

If $f_L(M+1) = f_R(M+1)$,

$$\begin{aligned} f(M+1) = f_L(M+1) + 1 &\geq f_L(M) \text{ (according to (6))} \\ &\geq f(M) - 1\ (f(M) = f_L(M) + 1)) \end{aligned}$$

If $f_L(M+1) \neq f_R(M+1)$, assume $f_L(M+1) > f_R(M+1)$.

$$f_L(M+1) > f_R(M+1) \geq f_R(M) - 1 \text{ (according to (6))}$$

$f_L(M+1) > f_R(M) - 1$. That is $f_L(M+1) \geq f_L(M)$.
So $f(M+1) \geq f_L(M+1) \geq f_L(M) = f(M) - 1$.

- $f_L(M) = f_R(M) = 1$: $f(M) = 2$. From Lemma 4.2, $i.left.maxnode > 2M$ and $i.right.maxnode > 2M$. So $i.maxnode > 2(M+1)$ (At least two nodes of right (or left) subtree will be at the same level with more than $2M$ nodes in left (or right) subtree). By Lemma 4.2 again, $f(M+1) > 0$. So $f(M+1) \geq 1 = f(M) - 1$.

So the induction step is also correct. Thus $f(M+1) \geq f(M) - 1$. ■

Considering Inequalities 3 and 4, we see that for a full binary analysis tree, using one additional mixer either keeps the minimum number of storage units the same or reduces it by one. So we get the conclusion below.

Corollary 4.2. *The total number of mixers and storage units $F(M) = M + f(M)$ is a non-decreasing function of M.*

4.4 Example

We apply the above algorithms and conclusions to characterize biochemical analyses based on resource requirements. The two trees in Figure 5 have different structures, but share the same resource requirements, as illustrated by their identical characteristic resource curve $f(M)$ in Figure 6.

5 Towards Smallest Chip Design

We discussed mixers and storage units in Section 4, without considering their geometry and other resources such as input/output units and transportation paths. In this section, we will consider all resources towards designing the smallest DMFS chip for biochemical analyses. The smaller the number of electrodes in a DMFS chip, the easier the fabrication and the lower the cost.

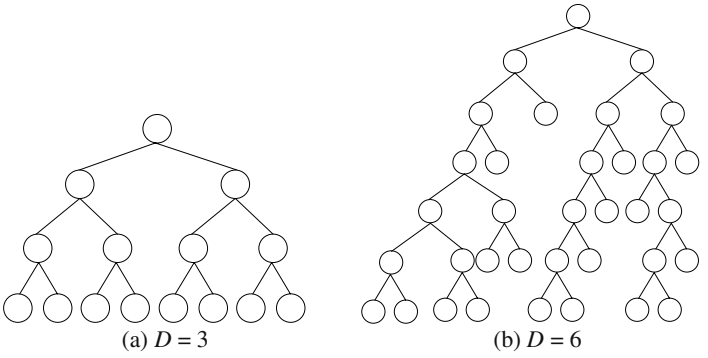


Fig. 5. Two analysis trees that have the same least resource requirements.

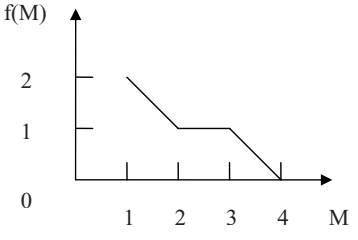


Fig. 6. Characteristic resource curve of $f(M)$ shared by the two trees in Figure 5.

5.1 Mixers and Storage Units

Mixers and storage units are the two primary functional components on a DMFS biochip, where mixers are used to mix and split droplets while storage units are used to store droplets on chip for future usage. They should be large enough to prevent droplets inside them inadvertently mixing with other droplets outside them. As shown in Figure 7, a storage unit needs only one electrode to hold one droplet, while a mixer needs more electrodes to move the mixed droplet inside it and thus needs more surrounding electrodes to keep the droplet inside separate from other droplets outside. It follows that the *size* of one mixer S_{mix} (the number of electrodes in one mixer) is bigger than the size of a storage unit S_{store}.

Fig. 7. The sizes of a mixer and a storage unit. (a) A mixer with 3 electrodes for droplets to move. $S_{mix} = 15$. (b) A storage unit with 1 electrode for droplets to stay. $S_{store} = 9$.

From the above observation and Corollary 4.2, we obtain the following result for a full binary tree whose M-depth is $f(M)$.

Theorem 5.1. *The total size of M mixers and $f(M)$ storage units will monotonically increase with the number of mixers.*

Proof. We need to prove that $M_1 \cdot S_{mix} + f(M_1) \cdot S_{store} < M_2 \cdot S_{mix} + f(M_2) \cdot S_{store}$ when $M_1 < M_2$.

$M_1 \cdot S_{mix} + f(M_1) \cdot S_{store} = (M_1 + f(M_1)) \cdot S_{store} + M_1 \cdot (S_{mix} - S_{store})$
$\leq (M_2 + f(M_2)) \cdot S_{store} + M_1 \cdot (S_{mix} - S_{store}) < (M_2 + f(M_2)) \cdot S_{store} + M_2 \cdot (S_{mix} - S_{store})$
$= M_2 \cdot S_{mix} + f(M_2) \cdot S_{store}$ ■

5.2 Input and Output Units

Input and output units should be connected to the perimeter electrodes of the chip, which we call connection electrodes, as shown in Figure 8(b). For the tree in Figure 8(a), we need just one mixer and zero storage units for the reaction. If we directly connect the input units to the chip as in Figure 8(b), the number of corresponding connection electrodes will be the number of leaf nodes of the tree in Figure 8(a). Since the connection electrodes should be on the perimeter of the chip, the total number of electrodes on the chip will be proportional to the square of the number of connection electrodes, which is much bigger than the size of one mixer. (This assumes the chip is a rectangular array of electrodes.)

An alternative approach is to connect the input and output units to a ring of electrodes (as in [13]) and then connect the ring to a separate working zone of mixers and storage units as shown in Figure 9. The size of the working zone can be selected to have the required functionality, and the working zone can even be in the interior of the ring if it is sufficiently small.

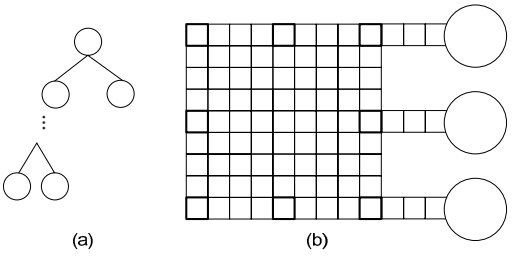


Fig. 8. The size of chip for an analysis tree when directly connecting the input units to the chip. (a) A full binary tree of depth 6, where the right child of each node is a leaf node or a null node. (b) The chip containing sufficient connection electrodes, shown bold on the chip perimeter. (A subset of input units are shown.)

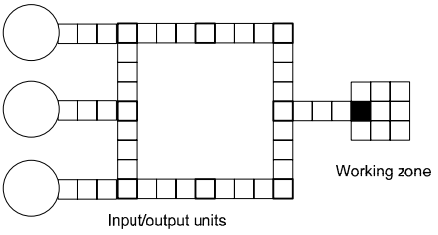


Fig. 9. Connect the input and output units to the working zone of mixers and storage units through a ring so that all droplets to the working zone are input from and output to one electrode (filled in black). Here the working zone is a 3×3 mixer.

5.3 Transportation Paths

Theorem 5.2. *Even considering the transportation paths, the smallest chip is constructed using one mixer and $f(1)$ storage units.*

Proof. Proof by contradiction. First, the smallest chip can be constructed using M mixers and $f(M)$ storage units since we can always convert extra storage units (if more than $f(M)$) to transport electrodes without increasing the chip size. Second, assume $M > 1$ in the smallest chip. The corresponding transportation paths should connect all mixers and storage units so that they are reachable from each other. The size of mixers and storage units is $M \cdot S_{mix} + f(M) \cdot S_{store}$; we assume the number of electrodes for input/output units are determined by the analysis tree and therefore constant. Since the size of a mixer is bigger than that of a storage unit, we can retain one mixer and change all other mixers to be storage units. The transportation paths can still connect all mixers and storage units, and the size of mixers and storage units is now reduced to $1 \cdot S_{mix} + (M - 1 + f(M)) \cdot S_{store}$. The size of one mixer and $f(1)$ storage units is $1 \cdot S_{mix} + f(1) \cdot S_{store}$. From Corollary 4.2, $M + f(M) \geq 1 + f(1)$, which implies the chip with M mixers is larger than the chip with one mixer, leading to a contradiction. We conclude that even considering the transportation paths, the smallest chip is constructed using one mixer and $f(1)$ storage units. ■

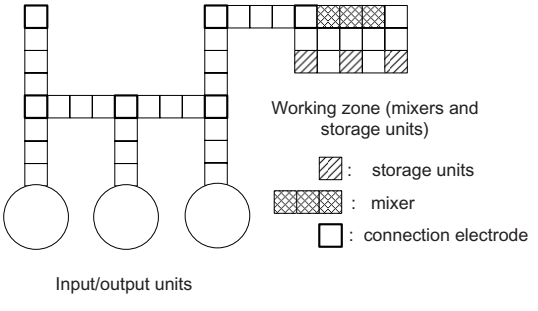


Fig. 10. The partial layout of the smallest chip for a complete tree of depth 4.

5.4 Example

Consider a complete tree of depth 4. By Theorem 4.2, $f(1)$ for the complete tree is 3. Suppose that mixing and splitting droplets is performed in a mixer with 3 electrodes for droplets to move as shown in Figure 7 (a). Since we can place the mixer and storage units along the perimeter of the chip and overlap a subset of the surrounding electrodes, the size of a mixer and a storage unit can be smaller than that shown in Figure 7. Figure 10 shows the layout of the smallest chip, without showing all input and output units.

6 Conclusion

In this paper, we focused on characterizing the resource requirements of arbitrary biochemical analyses on DMFS biochips from their tree structure. We designed a corresponding scheduling algorithm to perform the biochemical analyses using the least resource requirements. We also defined the M-depth of an analysis tree to describe such resource requirements and infer the variation in the number and size of resources as a function of M and the tree structure. We can use these results to design the smallest biochip for any biochemical analysis and also determine whether a particular biochip can be used for a biochemical analysis. These results represent our initial steps towards automated design of digital microfluidic biochips. In our future work, we will use these results to explore the tradeoff between resource requirements and the completion time of biochemical analyses, and combine these results with automated layout design and routing algorithms for DMFS biochips.

Acknowledgements. This work was supported in part by the National Science Foundation under Award Nos. IIS-0713517, IIS-0730817, and CNS-0709099.

References

1. Böhringer, K.-F.: Modeling and controlling parallel tasks in droplet-based microfluidic systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20(12), 1463–1468 (2001)
2. Chen, X.X., Wu, H.K., Mao, C.D., Whitesides, G.M.: A prototype two-dimensional capillary electrophoresis system fabricated in poly(dimenthylsiloxane). Anal. Chem. 74, 1772–1778 (2002)
3. Cho, S.K., Moon, H., Kim, C.: Creating, transporting, cutting, and merging liquid droplets by electrowetting-based actuation for digital microfluidic circuits. J. Microelectromech. Syst. 12(1), 70–80 (2003)
4. Ding, J., Chakrabarty, K., Fair, R.B.: Scheduling of microfluidic operations for reconfigurable two-dimensional electrowetting arrays. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(2), 329–339 (2006)
5. Dittrich, P., Manz, A.: Lab-on-a-chip: microfluidics in drug discovery. Nature Reviews Drug Discovery 5(3), 210–218 (2006)

6. Fair, R.B., Srinivasan, V., Ren, H., Paik, P., Pamula, V., Pollack, M.G.: Electrowetting-based on-chip sample processing for integrated microfluidics. In: IEEE International Electron Devices Meeting (IEDM), pp. 779–782 (2003)
7. Griffith, E.J., Akella, S.: Coordinating multiple droplets in planar array digital microfluidic systems. International Journal of Robotics Research 24(11), 933–949 (2005)
8. Griffith, E.J., Akella, S., Goldberg, M.K.: Performance characterization of a reconfigurable planar array digital microfluidic system. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(2), 340–352 (2006)
9. Kwok, Y., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computing Surveys, 406–471 (1999)
10. Luo, L., Akella, S.: Optimal scheduling for biochemical analyses on digital microfluidic systems. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2007, pp. 3151–3157 (2007)
11. Paik, P., Pamula, V.K., Fair, R.B.: Rapid droplet mixers for digital microfluidic systems. Lab on a chip 3, 253–259 (2003)
12. Pollack, M.G., Fair, R.B., Shenderov, A.D.: Electrowetting-based actuation of liquid droplets for microfluidic applications. Appl. Phys. Lett. 77(11), 1725–1726 (2000)
13. Srinivasan, V., Pamula, V.K., Fair, R.B.: An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids. Lab on a Chip 5(3), 310–315 (2004)
14. Su, F., Chakrabarty, K.: Architectural-level synthesis of digital microfluidics-based biochips. In: Proc. IEEE International Conference on CAD, pp. 223–228 (2004)
15. Su, F., Chakrabarty, K.: Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips. In: Proc. IEEE/ACM Design Automation Conference, pp. 825–830 (2005)
16. Su, F., Chakrabarty, K.: Module placement for fault-tolerant microfluidics-based biochips. ACM Transactions on Design Automation of Electronic Systems, 682–710 (2006)
17. Su, F., Hwang, W., Chakrabarty, K.: Droplet routing in the synthesis of digital microfluidic biochips. In: Design, Automation and Test in Europe (DATE) Conference, Munich, Germany, March 2006, pp. 323–328 (2006)
18. Zheng, B., Gerdts, C., Ismagilov, R.F.: Using nanoliter plugs in microfluidics to facilitate and understand protein crystallization. Current Opinion in Structural Biology 15, 548–555 (2005)

Path Planning for Flexible Needles Using Second Order Error Propagation

Wooram Park, Yunfeng Wang, and Gregory S. Chirikjian

Abstract. In this paper we propose a computationally efficient method for the steering of flexible needles with a bevel tip in the presence of uncertainties for the case when there are no obstacles in the environment. Based on the stochastic model for the needles, we develop a new framework for path planning of a flexible needle with a bevel tip. This consists of three parts: (a) approximation of probability density functions for the needle tip pose; (b) application of a second order error propagation algorithm on the Euclidean motion group; and (c) application of the path-of-probability (POP) algorithm. The probability density functions are approximated as Gaussians under the assumption that the uncertainty in the needle insertion is fairly small. The means and the covariances for the probability density functions are estimated using the error propagation algorithm that has second order accuracy. The POP algorithm is adapted to the path planning for the flexible needles so as to give the appropriate steering plan. Combining these components and considering 5 degree-of-freedom targets, the new method gives the path of the flexible needle that hits the target point with the desired hitting direction.

1 Introduction

A number of recent works have been reported on the topic of the steerable flexible needles with bevel tips that are inserted into soft tissue for minimally invasive

Wooram Park and Gregory S. Chirikjian
Department of Mechanical Engineering, Johns Hopkins University,
Baltimore, MD 21218, USA
e-mail: {wpark7,gregc}@jhu.edu

Yunfeng Wang
Department of Mechanical Engineering, The College of New Jersey, Ewing,
NJ 08628 USA
e-mail: jwang@tcnj.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 583–595.
springerlink.com

medical treatments [1, 2, 7, 12, 17]. In this problem, a flexible needle is rotated with the angular speed $\omega(t)$ around its tangent while it is inserted with translational speed $v(t)$ in the tangential direction. Due to the bevel tip, the needle will not follow a straight line even when $\omega(t) = 0$ and $v(t)$ is constant. Rather, in this case the tip of the needle will approximately follow a circular arc with curvature κ when the medium is very firm and the needle is very flexible. The specific value of the constant κ depends on parameters such as the angle of the bevel, how sharp the needle is, and properties of the tissue. In practice κ is fit to experimental observations of the needle bending in a particular medium during insertions with $\omega(t) = 0$ and constant $v(t)$. Using this as a baseline, and building in arbitrary $\omega(t)$ and $v(t)$, a nonholonomic kinematic model then predicts the time evolution of the position and orientation of the needle tip [12, 17].

One of the most important tasks related to this flexible needle is that we should control the needles in order to get the desired tip position with or without the desired final direction, because the needles are used for drug injection or biopsy at a specific location. The path obtained by planning algorithms will be used for the actual control of the needle. In this paper, we focus on the path planning for the flexible needle.

A stochastic model for the steering of flexible needles with bevel tips has been developed in [12, 13]. It adopted the unicycle nonholonomic kinematic model [17]. Furthermore, it includes white noises weighted by coloring constants to capture the nondeterministic behavior of the needle insertion. This method, which is reviewed in detail in Section 2.2, is modified in this paper in such a way that allows for the closed-form evaluation of probability densities. The benefit of the closed-form method developed here is that it enables fast path planning. In order to evaluate the parameters that serve as the input to this closed-form probability density, the kinematic covariance propagation method developed in [14] is used. This probability density function is used for the path planning that was developed in [12, 13]. Like the work by Mason and Burdick [11], this path planning algorithm is an extension of the path-of-probability (POP) algorithm presented in [8]. Duindam et al. presented a path planning method for 3D flexible needle steering [7]. Although they consider 3D needle steering with obstacles, only positions of the targets are aimed for in that work. Our method gives a path that hits the goal position with the "desired direction" in a 3D environment without obstacles.

There exist some methods for steering nonholonomic systems [10]. Since some of them use the concept of optimal control, it is computationally intensive. Furthermore, since the needle system is not small-time locally controllable [12], the small changes in the goal pose can lead to large changes in the optimal path. We also note that Brockett's theorem says that some nonholonomic systems can not be stabilized to a desired pose using a continuous feedback. The POP algorithm used for the needle path planning in this paper has benefits compared to the existing methods based on optimal path following or optimal control: (1) At each time step we can make a choice about what control input to use, independent of the previous step, which means that this control is discontinuous and the limitations imposed by Brockett's theorem do not apply; (2) The path that we generate is not the path of minimal

length, or optimal, and so the solution is not as sensitive to small changes in the desired position as methods based on optimal control. Of course, this means that our paths may be a little bit longer, but we believe that they are also more robust to perturbations.

2 Mathematical Methods

2.1 Review of Rigid-Body Motions

The special orthogonal group, SO(3), is the space of rotation matrices contained in $\mathbb{R}^{3\times 3}$, together with the operator of matrix multiplication. Any element of SO(3) can be written using the Euler angles as [6]

$$R = R_z(\alpha)R_x(\beta)R_z(\gamma),$$

where α, β and γ are the ZXZ Euler angles, $0 \le \alpha, \gamma \le 2\pi$, $0 \le \beta \le \pi$ and

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}.$$

The Euclidean motion group, SE(3), represents rigid-body motions in 3D space. It is the semi-direct product of $\mathbb{R}^3$ with SO(3). The elements of SE(3) can be written as [6]

$$g = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}, \tag{1}$$

where $R \in \mathrm{SO}(3)$, $\mathbf{t} \in \mathbb{R}^3$ and $\mathbf{0}^T$ denotes the transpose of the 3D zero vector.

Given a time-dependent rigid-body motion $g(t)$, the quantity

$$g^{-1}\dot{g} = \begin{pmatrix} R^T\dot{R} & R^T\dot{\mathbf{t}} \\ \mathbf{0}^T & 0 \end{pmatrix} \in se(3) \tag{2}$$

(where a dot represents the time derivative) is a spatial velocity as seen in the body-fixed frame, where $se(3)$ is the Lie algebra associated with SE(3). We identify $se(3)$ with $\mathbb{R}^6$ in the usual way via the mappings $^\vee : se(3) \to \mathbb{R}^6$ and $\widehat{\ } : \mathbb{R}^6 \to se(3)$, given by

$$\xi = (g^{-1}\dot{g})^\vee = \begin{pmatrix} (R^T\dot{R})^\vee \\ R^T\dot{\mathbf{t}} \end{pmatrix} = \begin{pmatrix} \omega \\ \mathbf{v} \end{pmatrix} \in \mathbb{R}^6$$

and

$$\widehat{\xi} = \widehat{\begin{pmatrix} \omega \\ \mathbf{v} \end{pmatrix}} = \begin{pmatrix} \widehat{\omega} & \mathbf{v} \\ \mathbf{0}^T & 0 \end{pmatrix} \in se(3),$$

where

$$\widehat{\omega} = \widehat{\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}.$$

The vector ξ contains both the angular and translational velocity of the motion $g(t)$ as seen in the body-fixed frame of reference.

Let $\mathbf{e}_i$, $i = 1,\ldots,6$ denote the standard basis for $\mathbb{R}^6$. The basis given by a set of matrices $E_i = \widehat{\mathbf{e}}_i$, $i = 1,\ldots,6$ produce elements of SE(3), when linearly combined and exponentiated. Specifically we have [6]

$$E_1 = \begin{pmatrix} 0&0&0&0 \\ 0&0&-1&0 \\ 0&1&0&0 \\ 0&0&0&0 \end{pmatrix}; \quad E_2 = \begin{pmatrix} 0&0&1&0 \\ 0&0&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \end{pmatrix}; \quad E_3 = \begin{pmatrix} 0&-1&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix};$$

$$E_4 = \begin{pmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}; \quad E_5 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{pmatrix}; \quad E_6 = \begin{pmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{pmatrix}.$$

The element of SE(3) can be obtained by the exponential mapping as [6, 13]

$$g = g(x_1,x_2,\ldots,x_6) = \exp\left(\sum_{i=1}^{6} x_i E_i\right).$$

Therefore the vector $\mathbf{x} = (x_1\ x_2\ \ldots\ x_6)^T$ can be obtained from $g \in \mathrm{SE}(3)$ by

$$\mathbf{x} = (\log g)^{\vee}.$$

If $X \in se(3)$ is an arbitrary element of the form

$$X = \begin{pmatrix} \Omega & \mathbf{v} \\ \mathbf{0}^T & 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{x} = (X)^{\vee} = \begin{pmatrix} \omega \\ \mathbf{v} \end{pmatrix},$$

then $Ad(g)$ (the adjoint) is defined by the expression

$$(gXg^{-1})^{\vee} = Ad(g)\mathbf{x}, \quad \text{where} \quad Ad(g) = \begin{pmatrix} R & 0 \\ TR & R \end{pmatrix}. \tag{3}$$

The matrix T is skew-symmetric, and $T^{\vee} = \mathbf{t}$, when $g \in \mathrm{SE}(3)$ is given as in (1).

2.2 Nonholonomic Stochastic Needle Model

In a reference frame attached to the needle tip with the local z axis denoting the tangent to the "backbone curve" of the needle, and x denoting the axis orthogonal to the direction of infinitesimal motion induced by the bevel (i.e., the needle bends in

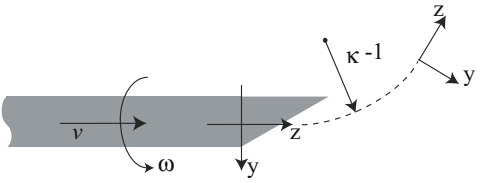


Fig. 1. The definition of parameters and frames in the nonholonomic needle model [12, 17].

the $y-z$ plane), the nonholonomic kinematic model for the evolution of the frame at the needle tip was developed in [12, 17] as:

$$\xi = (g^{-1}\dot{g})^{\vee} = \begin{bmatrix} \kappa & 0 & \omega(t) & 0 & 0 & v(t) \end{bmatrix}^T, \tag{4}$$

where $\omega(t)$ and $v(t)$ are the rotation and insertion speeds, respectively. The frames and parameters for the needle are shown in Fig. 1.

If everything were certain, and if this model were exact, then $g(t)$ could be obtained by simply integrating the ordinary differential equation in (4). However, in practice a needle that is repeatedly inserted into a medium such as gelatin (which is used to simulate soft tissue [17]) will demonstrate an ensemble of slightly different trajectories.

A simple stochastic model for the needle is obtained by letting [12, 13]:

$$\omega(t) = \omega_0(t) + \lambda_1 w_1(t), \text{ and } v(t) = v_0(t) + \lambda_2 w_2(t).$$

Here $\omega_0(t)$ and $v_0(t)$ are what the inputs would be in the ideal case, $w_1(t)$ and $w_2(t)$ are uncorrelated unit Gaussian white noises, and λ_i are constants.

Thus, a nonholonomic needle model with noise is

$$(g^{-1}\dot{g})^{\vee}dt = \begin{bmatrix} \kappa & 0 & \omega_0(t) & 0 & 0 & v_0(t) \end{bmatrix}^T dt + \begin{bmatrix} 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_2 \end{bmatrix}^T \begin{bmatrix} dW_1 \\ dW_2 \end{bmatrix} \tag{5}$$

where $dW_i = W_i(t+dt) - W_i(t) = w_i(t)dt$ are the non-differentiable increments of a Wiener process $W_i(t)$. This noise model is a stochastic differential equation (SDE) on SE(3). As shorthand, we write this as

$$(g^{-1}\dot{g})^{\vee}dt = \mathbf{h}(t)dt + Hd\mathbf{W}(t). \tag{6}$$

Corresponding to this SDE is the Fokker-Planck equation that describes the evolution of the probability density function of the ensemble of tip positions and orientations at each value of time, t [12, 13]:

$$\frac{\partial \rho(g;t)}{\partial t} = -\sum_{i=1}^{d} h_i(t)\tilde{E}_i^r \rho(g;t) + \frac{1}{2}\sum_{i,j=1}^{d} D_{ij}\tilde{E}_i^r \tilde{E}_j^r \rho(g;t) \tag{7}$$

where $D_{ij} = \sum_{k=1}^{m} H_{ik} H_{kj}^T$ and $\rho(g;0) = \delta(g)$. In (7) the "right" Lie derivative $\tilde{E}_i^r$ is defined for any differentiable function $f(g)$ as

$$\tilde{E}_i^r f(g) = \left(\frac{d}{dt} f(g \circ \exp(tE_i)) \right) \Big|_{t=0}. \tag{8}$$

For a small amount of diffusion, the solution for the Fokker-Planck equation, (7), can be approximated by a shifted Gaussian function [13, 15]:

$$\rho(g(\mathbf{x});t) = (2\pi)^{-3} |\det(\Sigma)|^{-1/2} \exp\left(-\frac{1}{2} (\mathbf{x} - \mu(t))^T \Sigma^{-1} (\mathbf{x} - \mu(t)) \right), \tag{9}$$

where $g = \exp(\hat{\mathbf{x}})$, and μ and Σ are the mean and the covariance of the probability density function, $\rho(g;t)$, respectively. This approximation is based on the fact that for small diffusion the Lie derivative is approximated as [13]

$$\tilde{E}_i^r f(g) \approx \frac{\partial f}{\partial x_i}.$$

Using this, the Fokker-Planck equation (7) becomes a diffusion equation in $\mathbb{R}^6$. Therefore, we have the solution for the diffusion equation as (9).

2.3 Second Order Error Propagation

If a unique value $\mu \in \mathrm{SE}(3)$ exists for which

$$\int_{\mathrm{SE}(3)} [\log(\mu^{-1}(t) \circ g)]^{\vee} \rho(g;t) dg = \mathbf{0},$$

then $\mu(t)$ is called the mean of a pdf $\rho(g,t)$. In addition, the covariance about the mean is defined as [16]

$$\Sigma(t) = \int_{\mathrm{SE}(3)} \log(\mu^{-1}(t) \circ g)^{\vee} [\log(\mu^{-1}(t) \circ g)^{\vee}]^T \rho(g;t) dg.$$

Suppose that for small values of t, the quantities $\mu(t)$ and $\Sigma(t)$ corresponding to $\rho(g;t)$ can be obtained (even if $\rho(g;t)$ is not known in closed form). Then these can be propagated over longer times. In other words, due to the Markovian nature of the above model, solutions can be "pasted together" using the fact that the following convolution equalities hold:

$$\rho(g;t_1 + t_2) = \rho(g;t_1) * \rho(g;t_2),$$

where convolution on $SE(3)$ is defined as in [5]. Even if these convolutions are too time-consuming to compute explicitly, the fact that these expressions hold means that propagation formulas for the mean and covariance can be used.

Wang and Chirikjian [16] derived the formulas for the second order propagation. If a pdf, $\rho_i(g)$ has mean μ_i and covariance Σ_i for $i = 1,2$, then with second order accuracy, the mean and covariance of $(\rho_1 * \rho_2)(g)$ are respectively [6]

$$\mu_{1*2} = \mu_1 \circ \mu_2 \qquad \text{and} \qquad \Sigma_{1*2} = A + B + F(A,B), \tag{10}$$

where $A = Ad(\mu_2^{-1})\Sigma_1 Ad^T(\mu_2^{-1})$, $B = \Sigma_2$ and $F(A,B)$ is given in Appendix. Consequently, we can obtain the mean and covariance for a relatively large t with given mean and covariance for a small t by this propagation formulas. These can then be substituted into (9) to obtain a closed-form estimate of the probability density that the needle will reach any particular pose at any value of time.

3 Path Planning for Flexible Needles

For path planning, we use the algorithm that appeared in [12, 13]. This algorithm was adapted from the path-of-probability algorithm in [8]. A similar trajectory planning method can be also found in [11].

In this algorithm, we find the whole path by serially pasting together several intermediate paths. Fig. 2 shows the concept of this algorithm. We aim to find a path that starts at g_0 and ends at g_{goal} using M intermediate steps. The homogeneous transformation matrix, $g_i \in \mathrm{SE}(3)$ $(i = 1,2,...,M)$, represents the position and orientation of the ith frame with respect to $(i-1)$th frame as shown in Fig. 2. Suppose that the $(i-1)$ intermediate steps $(g_1, g_2, \cdots, g_{i-1} \in \mathrm{SE}(3))$ have been already determined. The intermediate step, g_i is determined to maximize the probability that the remaining steps reach the goal. In Fig. 2, the shaded ellipses depict the probability density function when we consider the remaining $(M-i)$ steps. In other words, when we consider $(M-i)$ intermediate steps after g_i, the final pose will be in the dark area with higher probability than the bright area. Comparing the two simplified cases in Fig. 2, if the previous intermediate steps $(g_1, g_2, \cdots, g_{i-1})$ are the same for both cases, we should choose g_i as shown in Fig. 2b, because it guarantees the higher probability that the final pose reaches the goal pose.

The determination of the intermediate steps can be formulated as

$$g_i = \arg\max_{g \in S} \rho\left((g_0 \circ g_1 \cdots g_{i-1} \circ g)^{-1} \circ g_{goal}; \tau_i\right), \tag{11}$$

where τ_i is the remaining time to hit the goal and S is the set of possible intermediate poses. Now let us adapt this to the needle insertion problem. If t_{total} is the fixed time for the insertion from g_0 to g_{goal} and we have M intermediate steps, each intermediate step takes $\Delta t = t_{total}/M$. Therefore, we can define $\tau_i = (M-i)t_{total}/M$ in (11). Technically, this formula can not be used for determining the final intermediate step, g_M, because there is no remaining path. The final step can be determined to minimize the difference between the resulting final pose of the path and the goal pose using metrics such as in [4, 9].

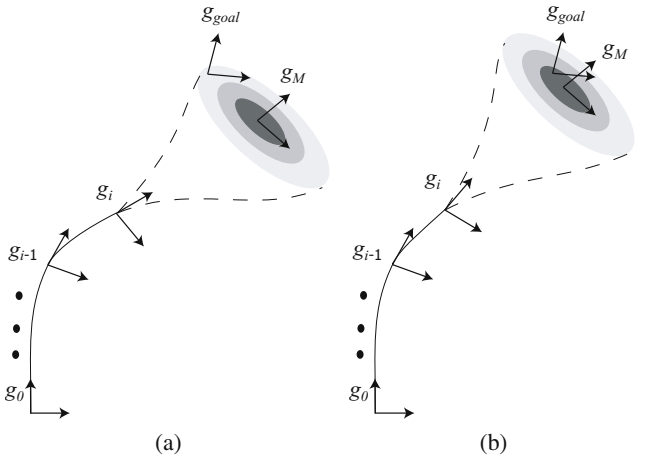


Fig. 2. The path-of-probability algorithm at the ith step. (a) Evaluation of one candidate move, g_i, with low resulting probability of reaching goal, (b) an intermediate step, g_i, resulting in high probability of reaching the goal.

Since the flexible needle that we consider is usually controlled by rotating along the axis tangential to the needle curve under the constant insertion speed, the above algorithm can be modified into the following: The intermediate step, g_i can be determined by

$$\theta_i = \underset{\theta\in[0\,2\pi)}{\arg\max}\,\rho((g_0 \circ g_1 \cdots g_{i-1} \circ \tilde{R}(\theta) \circ \mu(\Delta t))^{-1} \circ g_{goal}; (M-i)t_{total}/M), \quad (12)$$

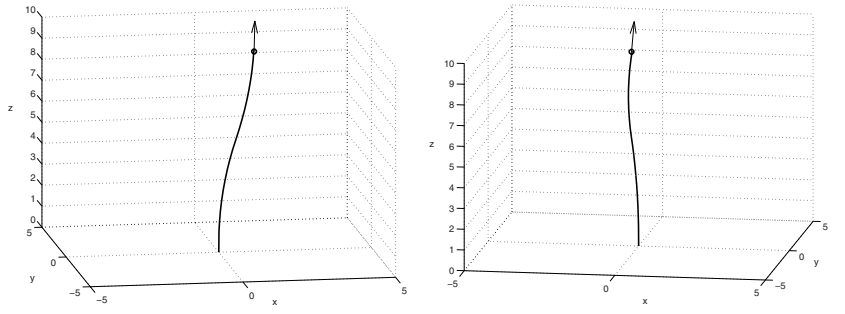
$$g_i = \tilde{R}(\theta_i) \circ g_{\Delta t} \quad (13)$$

where

$$\tilde{R}(\theta) = \begin{pmatrix} R_z(\theta) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

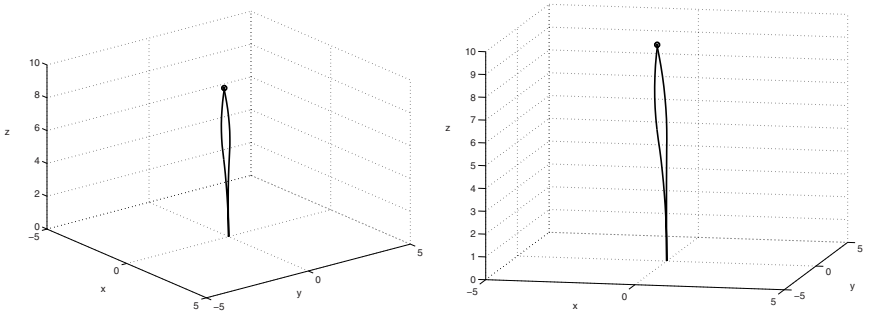
and $g_{\Delta t}$ is one sample path obtained by integrating the SDE (6) up to time $t = \Delta t = t_{total}/M$ (which is our model for what a real needle would do over this period of time), and $\mu(\Delta t)$ is the sample mean of the SDE (6) at the time $t = \Delta t = t_{total}/M$. The resulting whole path is given as $g = g_0 \circ g_1 \cdots g_M$. This insertion is achieved by repeating the twisting without insertion and the insertion without twisting.

It is important to understand why $\mu(\Delta t)$ and $g_{\Delta t}$ are used in (12) and (13), respectively. When determining θ_i using (12), the ith insertion has not been actually performed. In order to evaluate the probability, we need an estimate for the simple insertion after the rotation. Since our needle insertion system has the stochastic behavior, the mean path, $\mu(\Delta t)$, is the reasonable choice for the estimate. After obtaining θ_i, we need to define g_i carefully. The ith intermediate step, g_i, is that we twist the needle by θ_i and then insert it with constant insertion speed without twisting. Since the simple insertion is the "actual" insertion, this should not be the



(a) (b)

Fig. 3. Results of path planning for the flexible needles. The thin arrow shows the desired direction of the needle and the circle shows the target position. (a) Needle path for the goal, $(\alpha, \beta, p_x, p_y, p_z) = (0, 0.1, 1, -1, 9.8)$ (b) Needle path for the goal, $(\alpha, \beta, p_x, p_y, p_z) = (\pi/4, 0.1, 0, -1.5, 9.8)$



(a) (b)

Fig. 4. Results of path planning for the flexible needles. (a) Needle paths for the goals, $(\alpha, \beta, p_x, p_y, p_z) = (0, 0.1, 1, -1, 9.8)$ and $(\pi/2, 0.1, 1, -1, 9.8)$ (b) Needle paths for the goals, $(\alpha, \beta, p_x, p_y, p_z) = (\pi/4, 0.1, 0, -1.5, 9.8)$ and $(-\pi/4, 0.1, 0, -1.5, 9.8)$

mean path. Rather, one sample path obtained by integrating the SDE (6) up to time $t = \Delta t = t_{total}/M$ is more realistic choice as in (13).

In practice, $\mu(\Delta t)$ can be approximated by noise-free path that can be obtained by integrating the deterministic model (4) up to $t = \Delta t$, if $\|D\Delta t\|$ is small. D was defined in (7). In addition, the first covariance $\Sigma(\Delta t)$ that will be put in the propagation formula can be approximated by $D\Delta t$, because the diffusion by the noise term in SDE is small. Using the propagation formulas in (10) with the mean and covariance at $t = \Delta t$, we can compute the mean and covariance at $t = 2\Delta t, 3\Delta t, \cdots, M\Delta t$. Plugging these means and covariances into (9), we obtained the closed-form probability density function, $\rho(g;t)$. We use this pdf for the POP path planning algorithm.

In the real flexible needle system, if we twist the needle without insertion after the needle tip hits a target, the whole shape of the needle does not change. Therefore, out of 6 degrees of freedom of the target, we can naturally specify 5 degrees of freedom of the target pose using the desired position and direction. The remaining

rotational DOF (twisting around the needle backbone curve) can be considered as a free parameter which will be determined to guarantee the reacheability of the needle insertion system.

The target pose of the needle can be written as

$$g_{target} = \begin{pmatrix} R_z(\alpha)R_x(\beta)R_z(\gamma) & \mathbf{p} \\ \mathbf{0}^T & 1 \end{pmatrix},$$

where $\mathbf{p}$ is a 3D position vector. First we specify the 5 degrees of freedom: α, β and $\mathbf{p}$ using the desired position and direction of the needle tip. Then, for various values of γ, we repeatedly perform the path planning algorithm until we have a reasonable path.

Fig. 3 shows the paths obtained by the suggested method. We used the parameter values, $\kappa = 0.0449$, $\lambda_1 = 0.08$ and $\lambda_2 = 0.015$. We also set $\omega_0(t) = 0$ and $v_0(t) = 1$ in (5). We used 20 intermediate steps. Using a current PC (Intel Core Duo processor 2.66GHz, 1GB memory) the propagation (10) takes 0.33(sec) for 20 intermediate steps. For given α, β, γ and $\mathbf{p}$, the POP algorithm (12) and (13) gives a result in 0.37(sec), when we use 50 candidates for θ in (12), which are equally-spaced from 0 to 2π(rad). In the example in Fig. 3a, we tested with the 18 candidates for γ, equally-spaced from 0 to 2π(rad) and we could find the path with $\gamma = 4.89$(rad). Fig. 4 shows that for the same target position, we can have different paths applying different desired directions.

4 Conclusions and Discussions

In this work, we proposed a path planning method for the flexible needle. This method mainly uses the path-of-probability algorithm which requires the probability density function. Based on the stochastic model for the needle insertion, we approximated the probability density function for the needle tip pose with a shifted Gaussian distribution and obtained the mean and covariance for the probability density function using the second order error propagation theory. Using the path-of-probability algorithm, the needle paths that reach the goal position along the desired direction were obtained. Since the propagation formulas need the mean and covariance for the short length as inputs and compute the mean and covariance for the relatively long length as outputs, we only have to sample the needle path for the short length. Therefore we could avoid extensive sampling and long-range integration which are time-consuming.

This new method has two advanced features. First it uses the second order propagation formulas which is more accurate than the first order one that was used in [13]. Second, it can deal with the desired final direction of the needle tip, which was ignored in [7, 12, 13]. Using the second feature, we can generate a needle path avoiding an obstacle in an indirect way. Fig. 4 shows the possibility of obstacle avoidance.

We approximated the probability density function as a Gaussian assuming small diffusion. Thus evaluation of the probability density function is less reliable in a

distant area from the mean. This aspect eventually affects the performance of our path planning method. Specifically, if the desired position and orientation of the needle are away from the mean path, the path planning method does not always guarantee to give a reasonable path. One solution would be that we should start with a stochastic model which reflects the given target pose. In this paper, we considered only one case where the noise-free path is a simple circular arc.[1] If we consider another noise-free path that can be obtained by changing $\omega_0(t)$ and $v_0(t)$ in (5), the path planning method will work for more various target poses. Future research should include the method of determining $\omega_0(t)$ and $v_0(t)$ which reflect the given target pose for better performance of the path planning method.

Acknowledgements. This work was supported by NIH Grant R01EB006435 "Steering Flexible Needles in Soft Tissue."

Appendix

We review the second order propagation formulae. The entire work including derivation appears in [16].

If a pdf, $\rho_i(g)$, has mean μ_i and covariance Σ_i for $i = 1,2$, then to second order, the mean and covariance of $(\rho_1 * \rho_2)(g)$ are respectively [6]

$$\mu_{1*2} = \mu_1 \circ \mu_2 \text{ and } \Sigma_{1*2} = A + B + F(A,B),$$

where $A = Ad(\mu_2^{-1})\Sigma_1 Ad^T(\mu_2^{-1})$, $B = \Sigma_2$. Here

$$F(A,B) = C(A,B)/4 + (A''B + (A''B)^T + B''A + (B''A)^T)/12$$

A'' is computed as

$$A'' = \begin{pmatrix} A_{11} - \mathrm{tr}(A_{11})I_3 & 0_3 \\ A_{12} + A_{12}^T - 2\mathrm{tr}(A_{12})I_3 & A_{11} - \mathrm{tr}(A_{11})I_3 \end{pmatrix},$$

where A_{ij} are 3×3 block matrices in

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}.$$

B'' is defined in the same way with B replacing A everywhere in the expression. The blocks of C are computed as

$$C_{11} = -D_{11,11}$$

[1] Mathematically, the noise-free path and the mean path are not the same in the stochastic model for the flexible needle. However, we can treat them as the same in practice when the diffusion is small.

$$C_{12} = -(D_{21,11})^T - D_{11,12} = C_{21}$$
$$C_{22} = -D_{22,11} - D_{21,21} - (D_{21,12})^T - D_{11,22}$$

where $D_{ij,kl} = D(A_{ij}, B_{kl})$, and the matrix-valued function $D(A', B')$ is defined relative to the entries in the 3×3 blocks A' and B' as

$$d_{11} = -a'_{33}b'_{22} + a'_{31}b'_{32} + a'_{23}b'_{23} - a'_{22}b'_{33}, \qquad d_{12} = a'_{33}b'_{21} - a'_{32}b'_{31} - a'_{13}b'_{23} + a'_{21}b'_{33},$$
$$d_{13} = -a'_{23}b'_{21} + a'_{22}b'_{31} + a'_{13}b'_{22} - a'_{12}b'_{32}, \qquad d_{21} = a'_{33}b'_{12} - a'_{31}b'_{32} - a'_{21}b'_{13} + a'_{21}b'_{33},$$
$$d_{22} = -a'_{33}b'_{11} + a'_{31}b'_{31} + a'_{13}b'_{13} - a'_{11}b'_{33}, \qquad d_{23} = a'_{23}b'_{11} - a'_{21}b'_{31} - a'_{13}b'_{12} + a'_{11}b'_{32},$$
$$d_{31} = -a'_{32}b'_{12} + a'_{31}b'_{22} + a'_{22}b'_{13} - a'_{21}b'_{23}, \qquad d_{32} = a'_{32}b'_{11} - a'_{31}b'_{21} - a'_{12}b'_{13} + a'_{11}b'_{23},$$
$$d_{33} = -a'_{22}b'_{11} + a'_{21}b'_{21} + a'_{12}b'_{12} - a'_{11}b'_{22}.$$

References

1. Alterovitz, R., Goldberg, K., Okamura, A.: Planning for Steerable Bevel-tip Needle Insertion Through 2D Soft Tissue with Obstacles. In: IEEE Int. Conf. on Robot. and Auto., pp. 1652–1657 (2005)
2. Alterovitz, R., Siméon, T., Goldberg, K.: The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty. In: Proc. Robot. Sci. and Syst. (2007)
3. Brockett, R.W.: Asymptotic stability and feedback stabilization. Differential Geometric Control Theory, 181–208 (1983)
4. Chirikjian, G.S., Zhou, S.: Metrics on Motion and Deformation of Solid Models. ASME J. Mech. Des. 120(2), 252–261 (1998)
5. Chirikjian, G.S., Ebert-Uphoff, I.: Numerical Convolution on the Euclidean Group with Applications to Workspace Generation. IEEE Trans. on Robot. and Auto. 14(1), 123–136 (1998)
6. Chirikjian, G.S., Kyatkin, A.B.: Engineering Applications of Noncommutative Harmonic Analysis. CRC Press, Boca Raton (2000)
7. Duindam, V., Alterovitz, R., Sastry, S., Goldberg, K.: Screw-Based Motion Planning for Bevel-Tip Flexible Needles in 3D Environments with Obstacles. In: IEEE Int. Conf. on Robot. and Auto., pp. 2483–2488 (2008)
8. Ebert-Uphoff, I., Chirikjian, G.S.: Inverse Kinematics of Discretely Actuated Hyper-Redundant Manipulators Using Workspace Densities. In: IEEE Int. Conf. on Robot. and Auto., pp. 139–145 (1996)
9. Kuffner, J.J.: Effective Sampling and Distance Metrics for 3D Rigid Body Path Planning. In: IEEE Int. Conf. on Robot. and Auto., pp. 3993–3998 (2004)
10. Laumond, J.P. (ed.): Robot Motion Planning and Control. LNCIS, vol. 229. Springer, New York (1998)
11. Mason, R., Burdick, J.W.: Trajectory Planning Using Reachable-State Density Functions. In: IEEE Int. Conf. on Robot. and Auto., pp. 273–280 (2002)
12. Park, W., Kim, J.S., Zhou, Y., Cowan, N.J., Okamura, A.M., Chirikjian, G.S.: Diffusion-based motion planning for a nonholonomic flexible needle model. In: IEEE Int. Conf. on Robot. and Auto., pp. 4600–4605 (2005)
13. Park, W., Liu, Y., Zhou, Y., Moses, M., Chirikjian, G.S.: Kinematic State Estimation and Motion Planning for Stochastic Nonholonomic Systems Using the Exponential Map. Robotica, 419–434 (2008)

14. Wang, Y., Chirikjian, G.S.: Second-Order Theory of Error Propagation on Motion Groups. WAFR, New York City (2006)
15. Wang, Y., Chirikjian, G.S.: Error Propagation on the Euclidean Group with Applications to Manipulator Kinematics. IEEE Trans. on Robot. 22(4), 591–602 (2006)
16. Wang, Y., Chirikjian, G.S.: Nonparametric Second-Order Theory of Error Propagation on Motion Groups. Int. J. of Robot. Res. 27, 1258–1273 (2008)
17. Webster III, R.J., Kim, J.S., Cowan, N.J., Chirikjian, G.S., Okamura, A.M.: Nonholonomic modeling of needle steering. Int. J. of Robot. Res. 25, 509–525 (2006)

Part X
Planning

Path Planning among Movable Obstacles: A Probabilistically Complete Approach

Jur van den Berg, Mike Stilman, James Kuffner, Ming Lin, and Dinesh Manocha

Abstract. In this paper we study the problem of path planning among movable obstacles, in which a robot is allowed to move the obstacles if they block the robot's way from a start to a goal position. We make the observation that we can decouple the computations of the robot motions and the obstacle movements, and present a *probabilistically complete* algorithm, something which to date has not been achieved for this problem. Our algorithm maintains an explicit representation of the robot's configuration space. We present an efficient implementation for the case of planar, axis-aligned environments and report experimental results on challenging scenarios.

1 Introduction

In this paper we consider the problem of path planning among movable obstacles. This involves an environment with static and *movable* obstacles, and the task is for a robot to plan a path from some start position s to some goal position g, whereby the robot can move the movable obstacles out of its way. The robot and the movable obstacles may not collide with other obstacles.

This problem is more complex than typical robot path planning among only static obstacles. The computational challenge is similar to games like Sokoban [6] where it is easy to design puzzle scenarios that are difficult even for experienced human players. Not surprisingly, the problem is known to be NP-hard [19].

Jur van den Berg, Ming Lin, and Dinesh Manocha
Department of Computer Science, University of North Carolina at Chapel Hill
e-mail: {berg,lin,dm}@cs.unc.edu

Mike Stilman
School of Interactive Computing, Georgia Tech
e-mail: mstilman@cc.gatech.edu

James Kuffner
School of Computer Science, Carnegie Mellon University
e-mail: kuffner@cs.cmu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 599–614.
springerlink.com

Due to the complexity of our problem, previous works have focused on heuristics [5, 10, 15], and have given completeness results only for subclasses of the problem [1, 13, 14, 19]. No complete algorithms are known that cover the entire problem domain. In contrast, for other path planning problems generally applicable algorithms have been proposed that rely on *probabilistic completeness*— a weaker form of completeness that given infinite time guarantees to find a solution to any problem for which a solution exists [8, 9].

In this paper, we present a *probabilistically complete* algorithm that covers the entire domain of planning among movable obstacles. Our approach is based on the observation that we can *decouple* the computation of the robot's motion from the computation of the obstacle movements, if we maintain an explicit representation of the robot's free configuration space and keep track of which connected component the robot configuration resides in. We present an efficient data structure to maintain a representation of the robot's configuration space for the specific case in which both the robot and obstacle geometry can be represented by translating axis-aligned rectangles. We implemented our algorithm and data structure and present experimental results on challenging problem scenarios.

The rest of this paper is organized as follows. In the next section, we give an overview of previous work. In Section 3, we formally define our path planning problem. We present our approach and prove its probabilistic completeness in Section 4. In Section 5 we describe the implementation of our algorithm and a data structure to maintain the robot's configuration space, and discuss results in Section 6. We conclude the paper in Section 7.

2 Related Work

Achieving completeness in planning among movable obstacles has proven extremely challenging. The problem was shown to be NP-hard by Wilfong [19] and addressed with heuristic methods by Chen and Hwang [5]. Stilman and Kuffner [14] introduced (resolution-) completeness to this domain by showing that a subclass of problems called L_1 could be solved within a practical amount of time. The class was broadened to monotone problems in [15]. These approaches are particularly relevant to practical scenarios where an efficient method is required to identify blocking obstacles and restore connectivity in the robot's free space. However, they cannot solve movable obstacle problems outside the given subclasses.

Constructing an efficient, generally complete algorithm is difficult even for the standard path planning problem of a single robot moving among static obstacles [3]. Recently, probabilistic completeness has become an alternative standard for path planning problems. Sampling-based planners such as PRM [8] and RRT [9] have proven to be very successful in a domains ranging from single and multiple robots [12, 17] to dynamic environments with non-holonomic constraints [7]. This success prompted Nieuwenhuisen et. al. [10] and Stilman et. al. [16] to apply sampling based planning in the movable obstacle domain. However, in both cases the expansion of search trees for individual obstacle movements was bounded to ensure

proper backtracking over alternative obstacle choices. In order to ensure probabilistic completeness an algorithm must explore all possibilities and allow these search trees to grow indefinitely.

We now propose an algorithm that allows indefinite exploration over all obstacle movements. The algorithm is proven to be probabilistically complete. Not only is this result new for the domain of planning among movable obstacles, but also for related domains such as rearrangement planning [2, 4, 11] or manipulation planning [1, 13] where the goal is specified in terms of goal configurations for the obstacles, rather than for the robot. While we do not directly address this variant of the problem, the observations in this paper can be applied to the design of probabilistically complete algorithms that span these domains as well.

3 Problem Definition

The problem we discuss in this paper is defined as follows. We are given a robot R and a two- (or three-) dimensional workspace containing a set of static obstacles O and a set of n (rigid) movable obstacles $\{M_1, \ldots, M_n\}$. We denote the configuration space of R, i.e. the set of all possible configurations of the robot, by C_R (e.g. if R is a "free-flying" robot in the plane, then $C_R = \mathbb{R}^2 \times [0, 2\pi)$), and we similarly denote the configuration space of each of the movable obstacles M_i by C_{M_i}. A movable obstacle cannot move by itself, but can be moved by R if R first *grasps* the obstacle.

Given a start configuration $s \in C_R$ and a goal configuration $g \in C_R$ for the robot, and initial configurations $(c_1, \ldots, c_n) \in C_{M_1} \times \cdots \times C_{M_n}$ for the movable obstacles, the task is to find a collision-free path for the robot R from s to g. The robot is allowed to move the movable obstacles, but only one at a time, and only if the robot is grasping the obstacle. During the movement of an obstacle, both the robot and the obstacle should be collision-free with respect to other obstacles. We generally

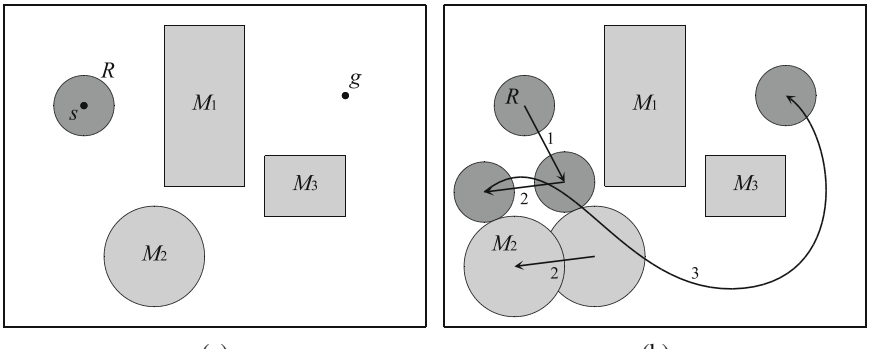


Fig. 1. (a) The initial situation of an example problem with three moving obstacles M_1, M_2 and M_3. The dark grey disc R is the robot. **(b)** An alternating sequence of navigation actions (1 and 3) and manipulation actions (2) that solves the problem.

define that the robot can grasp (and move) a movable obstacle when it is *touching* that obstacle.

The problem is thus defined as finding a *sequence of actions*, alternating between *navigation actions*, in which a robot moves by itself from a configuration on the boundary of one moving obstacle to a configuration on the boundary of another moving obstacle, and *manipulation actions*, in which the robot rigidly attaches itself to a movable obstacle and moves as a composite body from one position to the other. The first and last actions of the sequence are navigation actions that begin in the robot's start configuration s, and end in the robot's goal configuration g, respectively. The navigation actions in the sequence may include *regrasps* of the same obstacle, in which the robot moves to another configuration on the boundary of the moving obstacle to grasp it there. In Fig. 1 we show a sequence of actions that solve an example problem.

4 Approach

The problem has traditionally been approached by finding an alternating sequence of navigation actions and manipulation actions [1, 10, 13, 14]. This formulation, however, makes it difficult to devise a (probabilistically) complete planner. This is because each of the navigation and manipulation actions lie in a sub-dimensional "slice" of the composite configuration space $C_R \times C_{M_1} \times \cdots \times C_{M_n}$ of the robot and the obstacles. There is an infinite number of such slices, and each of these slices have zero probability to receive a sample in a probabilistic planner. Previous works have circumvented this problem by constraining the problem to a finite set of possible obstacle positions and grasps [1], or by dealing with only one movable obstacle [13].

In this paper, we discuss the general continuous problem with any number of movable obstacles. The key to our approach is that the problem should not be defined in terms of finding an alternating sequence of navigation and manipulation actions, but that one should abstract from the precise motions of the robot, and focus on the movements of the obstacles.

In our approach we are looking for a sequence of obstacle movements. The precise robot motions that lead to such obstacle movements are not explicitly computed; *we only make sure that the robot is somehow able to validly execute those movements*. In order to test whether movements of the movable obstacles are executable by the robot, we maintain an explicit representation of the *free configuration space* of the robot. Each of the static and movable obstacles induce a *C-obstacle* in the robot's configuration space, consisting of robot configurations in which the robot is in collision with that obstacle. The free configuration space is the space of configurations in which the robot is collision-free. This free configuration space consists of multiple *connected components*, whose boundaries consists of boundaries of C-obstacles (see Fig. 2a). If the robot is on the boundary of the C-obstacle of one of the movable obstacles, it is touching that obstacle. As defined above, it is then able to move that movable obstacle. So, if the robot is in a free connected component N, it is able to move the movable obstacles whose C-obstacles are adjacent to N. When

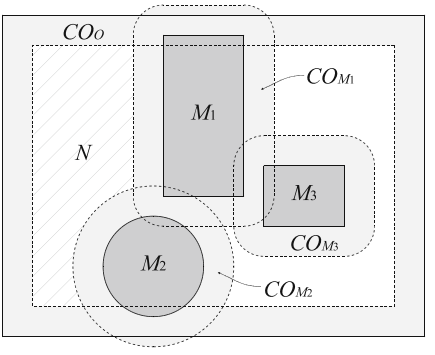


Fig. 2. The configuration space of the robot in the situation of Fig. 1a. The C-obstacles are shown light gray. There are two connected components in the robot's free configuration space. The dashed region labeled N is the one the robot is in.

an obstacle is moved, the configuration space of the robot and the connected component N change their shape, but as long as the C-obstacle of the movable obstacle remains adjacent to N, the robot is able to execute that movement. So, it is necessary only to keep track of which connected component the robot is in, rather than the exact configuration of the robot. This observation is central to our approach.

The task is now to find a sequence of obstacle movements that results in a situation in which the robot's goal configuration g is in the same connected component N as the robot. Once we have found such a sequence, we can (relatively easily) find the actual motions of the robot that execute these movements as a post-processing step.

In the remainder of this section, we will formalize the above observation and introduce the state space of the problem (Section 4.1). We then present a simple random-search algorithm (Section 4.2), and show that this algorithm is probabilistically complete (Section 4.3). Note that we do not make any assumption about the nature and dimensionality of the configuration spaces of both the robot and the movable obstacles.

4.1 State Space

Let us denote the robot R configured at $c_R \in C_R$ by $R(c_R)$, and similarly a movable obstacle M_i configured at $c_i \in C_{M_i}$ by $M_i(c_i)$.

Each of the movable obstacles generates a C-obstacle in the configuration space C_R of the robot (see Fig. 2). Given a specific configuration $c_i \in C_{M_i}$ of a movable obstacle M_i, its C-obstacle is given by $CO_{M_i}(c_i) = \{c_R \in C_R \,|\, M_i(c_i) \cap R(c_R) \neq \emptyset\}$. Similarly, the static obstacles generate a C-obstacle CO_O in C_R. Now, given specific configurations $(c_1, \ldots, c_n) \in C_{M_1} \times \cdots \times C_{M_n}$ of all movable obstacles, the *free configuration space* of the robot, i.e. the set of all configurations of the robot for which it is collision-free, is given by $C_R^{\text{free}}(c_1, \ldots, c_n) = C_R \setminus (CO_O \cup \bigcup_i CO_{M_i}(c_i))$. Note that the shape of the free space of the robot changes when an obstacle is moved.

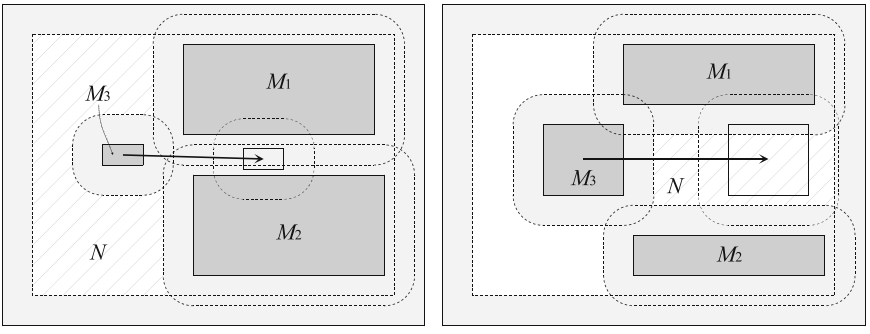


Fig. 3. Invalid obstacle movements. **(a)** The movement of obstacle M_3 is not executable by the robot, because at some point M_3's C-obstacle will not be adjacent to the connected component N anymore. **(b)** The movement of obstacle M_3 causes the connected component N of the robot to disappear, so there does not exist a collision-free motion for the robot to execute this movement.

At any time, the robot's free configuration space consists of one or more *connected components* (see Fig. 2), and the robot must be residing in one of them. We denote the connected component in which the robot resides by N.

In configurations on the *boundary* of N, the robot is *touching* some static or movable obstacle. If it is touching a movable obstacle M_i, the robot's configuration is on the boundary of the C-obstacle of M_i as well, and in that case the robot is able to move M_i. This leads to the following observation.

Definition 4.1 (Manipulable obstacle). *Given the configurations $(c_1,\ldots,c_n)$ of the movable obstacles and the connected component N of the robot's free configuration space that contains the robot, we define a movable obstacle M_i to be* manipulable *if its C-obstacle is adjacent to N, i.e. $\partial CO_{M_i} \cap \partial N \neq \emptyset$, where ∂ refers to the boundary of a set.*

Lemma 4.1. *Given initial configurations $(c_1,\ldots,c_n)$ of the movable obstacles and the connected component N of the robot's free configuration space that contains the robot, the movement of a movable obstacle M_i over a path $\pi : [0,1] \rightarrow C_{M_i}$, with $\pi(0) = c_i$, is valid and can be executed by the robot if (see Fig. 3 for examples of* in*valid obstacle movements):*

- *The movable obstacle M_i is* collision-free *with respect to the other obstacles at all times during the movement, i.e. $(\forall t \in [0,1] :: M_i(\pi(t)) \cap O = \emptyset \wedge (\forall j \neq i :: M_i(\pi(t)) \cap M_j(c_j) = \emptyset))$.*
- *The movable obstacle M_i is* manipulable *at all times during the movement, i.e. $(\forall t \in [0,1] :: \partial CO_{M_i}(\pi(t)) \cap \partial N \neq \emptyset)$. (Note that the shape of N changes during the movement of M_i over π.)*

Proof. Let the robot initially be in some configuration $c_R \in N$. As N shares part of its boundary with the boundary of CO_{M_i}, there exists some collision-free path within

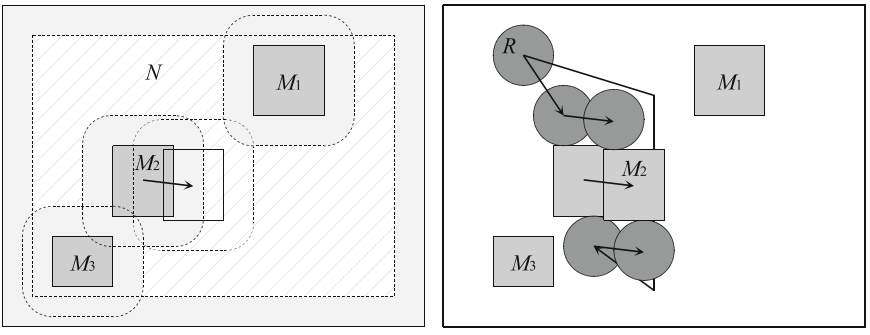


Fig. 4. (a) A situation in which the movement of obstacle M_2 splits the connected component N. **(b)** Two motions of the robot executing the same movement of M_2, but ending up in either of the two connected components formed by the split.

N for the robot to arrive at a point p on the boundary of CO_{M_i}. Let us rigidly attach this point p to CO_{M_i}. Now, the robot is able to move M_i along π. As M_i moves the point p moves, so p might leave the boundary of N at some moment. At the instant that this happens, the robot must regrasp, and find a new point p' that is both on the boundary of N and on the boundary of CO_{M_i}. As there is a part of the boundary of CO_{M_i} that is also on the boundary of N (see the second requirement), such a point p' must exist. As both p and p' are in N at that moment, there exists a free path for the robot that arrives at p'. Now the robot can continue moving M_i, and the above process can repeat until M_i arrives at configuration $\pi(1)$. □

As mentioned, the connected component N of the robot changes its shape during the movement of an obstacle. In some cases the obstacle movement may lead to a *split* of the connected component N into *two* new connected components (see Fig. 4a). In such a case, the robot may be in either of the two newly formed connected components after the split (see Fig. 4b), so we have to choose which component the robot will next be in.

Based on Lemma 4.1, we can define the *state space* our problem "lives" in. A *state* x is defined as a tuple $\langle c_1, \ldots, c_n, N \rangle$, where $c_1, \ldots, c_n$ are the configurations of the movable obstacles, and N is the connected component of the robot's free configuration space in which the robot resides (note that the definition of a state does not include any information regarding the specific configuration of the robot). The *state space* X is consequently defined as the set of all states. Given the robot's goal configuration g, the *goal region* $X_{\text{goal}} \subset X$ is given as the set of all states $x \in X$ for which $g \in N$. The initial state $x_{\text{init}} \in X$ is given by the initial configurations of the movable obstacles, and the connected component containing the robot's start configuration s.

We define an *action* u as a tuple $\langle M_i, \pi, \chi \rangle$, in which movable obstacle M_i is moved over path $\pi : [0,1] \rightarrow C_{M_i}$, and choices as given in χ are made with respect to the robot's connected component in cases of component splits encountered during

Algorithm 1. RANDOMTREE$(x_{\text{init}}, X_{\text{goal}})$

1: $T \leftarrow \{x_{\text{init}}\}$.
2: **while true do**
3: Pick a random state $x \in T$ from the tree.
4: $x' \leftarrow$ EXPAND(x).
5: $T \leftarrow T \cup \{x'\}$.
6: **if** $x' \in X_{\text{goal}}$ **then**
7: Path found! Terminate.
8: **end if**
9: **end while**

Algorithm 2. EXPAND$(x) : X$

1: Pick a random movable obstacle M_i that is manipulable in x.
2: Pick a random configuration c_i' from M_i's configuration space C_{M_i}.
3: **return** MOVEOBSTACLE(x, M_i, c_i').

Algorithm 3. MOVEOBSTACLE$(x = \langle c_1, \ldots, c_n, N \rangle, M_i, c_i') : X$

1: **while** M_i is manipulable **and** M_i is collision-free **and** M_i is not at c_i' **do**
2: Move M_i toward c_i', and keep track of the robot's connected component N.
3: **if** N splits into two components during the movement of M_i **then**
4: $N \leftarrow$ a component randomly chosen among the two formed by the split.
5: **end if**
6: **end while**
7: **return** the resulting state x'.

M_i's movement over π. An action is valid if the movement of M_i over π is valid according the requirements of Lemma 4.1. Applying an action u to a state $x \in X$ results in a new state $x' \in X$. Below, we present a simple algorithm that finds a sequence of valid actions that when applied to the initial state x_{init} gives a final state in X_{goal}.

4.2 Algorithm

Our algorithm randomly builds a tree of states that are connected by actions. The tree is rooted in the initial state x_{init}. In each iteration, we randomly pick a state from the tree, and *expand* it by applying a randomly chosen action to that state. The newly created state is then added to the tree. This repeats until a state has been reached that is in X_{goal} (see Algorithm 1). In the algorithm, we continuously keep track of an explicit representation of the robot's configuration space.

We only consider actions that move an obstacle along a *straight line* in the obstacle's configuration space (see Algorithm 2). The obstacle is moved toward a randomly chosen configuration as long as the movement is valid, or until the picked

configuration is reached (see Algorithm 3). We next prove that our algorithm is *probabilistically complete*.

4.3 Probabilistic Completeness

In the following, a problem solution is defined as a sequence of k *straight-line* actions $u_1,\ldots,u_k$, where $u_j = \langle M_{i_j}, \pi_j, \chi_j \rangle$ and each π_j is a straight-line path, that transform the initial state into a state in X_{goal}. Notice that any sequence can be approximated with one consisting of straight-line paths. A solution $u_1,\ldots,u_k$ has *clearance* ε if any alternative sequence $u'_1,\ldots,u'_k$ —where $u'_j = \langle M_{i_j}, \pi'_j, \chi'_j \rangle$ such that the endpoint of each π'_j deviates no more than $\frac{\varepsilon}{k}$ from the endpoint of π_j (i.e. $\|\pi'_j(1) - \pi_j(1)\| < \frac{\varepsilon}{k}$ for all $j \in 1..k$) *and* the correct choices χ'_j are made in case of component splits— is also a solution to the problem. Applying all these alternative sequences $u'_1,\ldots,u'_k$ to the initial state x_{init} gives a sequence $X_0 = \{x_{\text{init}}\}, X_1, \ldots, X_k$ of sets of states, such that $X_k \subset X_{\text{goal}}$. The following establishes probabilistic completeness for the random tree planner.

Theorem 4.1. *If there exists a solution with clearance $\varepsilon > 0$ then the the probability that* RANDOMTREE *will find a solution approaches 1 as the number of states in the tree approaches ∞.*

Proof. Assume that the random tree contains state $x_{j-1} \in X_{j-1}$ after some finite number $z-1$ of iterations. In the next iteration, each state in the tree has a probability $1/z$ to be selected for expansion (see line 3 of Algorithm 1). If x_{j-1} is chosen as the state to expand, there exists a second probability greater than some $q > 0$ that an action $\langle M_{i_j}, \pi'_j, \chi'_j \rangle$ with $\|\pi'_j(1) - \pi_j(1)\| < \frac{\varepsilon}{k}$ is chosen that results in a state $x_j \in X_j$ (see lines 1-2 of Algorithm 2; c'_i needs to be picked such that $\|c'_i - \pi_j(1)\| < \frac{\varepsilon}{k}$). Hence, the probability of 'success', i.e. that the next step in the solution sequence is constructed, in the z'th iteration is q/z.

Now, let random variable Y_z denote the number of successes we have had after z iterations. The expected value and the variance of Y_z are given by:

$$E(Y_z) = \sum_{i=1}^{z} \frac{q}{i} = q(\psi_0(z+1) + \gamma) \tag{1}$$

$$\text{Var}(Y_z) = \sum_{i=1}^{z} \left[\frac{q}{i}(1 - \frac{q}{i})^2 + (1 - \frac{q}{i})(\frac{q}{i})^2 \right] = E(Y_z) + q^2(\psi_1(z+1) - \tfrac{\pi^2}{6}) \tag{2}$$

where $\psi_n(x)$ is the n'th polygamma function, and γ the Euler-Mascheroni constant (the closed form for $\text{Var}(Y_z)$ was obtained using Maple).

To construct a solution sequence to a state $x_k \in X_k \subset X_{\text{goal}}$, we need k times success. The expected number of successes $E(Y_z)$ approaches infinity as the number of iterations z approaches infinity (i.e. $\lim_{z\to\infty} E(Y_z) = \infty$), so $E(Y_z) - k$ is positive for sufficiently large z. In these cases, the probability $\Pr(Y_z < k)$ that after z iterations a solution sequence has *not* been found is upper bounded by the Chebyshev inequality:

$$\Pr(Y_z < k) \;<\; \Pr(|E(Y_z) - Y_z| > E(Y_z) - k) \;\leq\; \frac{\mathrm{Var}(Y_z)}{(E(Y_z) - k)^2} \tag{3}$$

$\Pr(Y_z < k)$ approaches zero as the number of iterations z approaches infinity, as

$$\lim_{z \to \infty} \Pr(Y_z < k) \;\leq\; \lim_{z \to \infty} \frac{\mathrm{Var}(Y_z)}{(E(Y_z) - k)^2} \;=\; 0 \tag{4}$$

Hence, the probability $1 - \Pr(Y_z < k)$ that a solution *has* been found approaches 1 as the number of states in the tree approaches infinity. □

5 Implementation

The challenging part of the above algorithm is to explicitly maintain the robot's configuration space, and detect events (such as connected component splits) critical for the algorithm. Below, we present a data structure for the specific case of all obstacles and the robot being *axis-aligned rectangles* that can *translate* in the *plane*. This data structure enables us to efficiently perform the checks of lines 1 and 3 of Algorithm 3, in an *exact* and *continuous* manner, so we do not have to rely on approximations taking small discrete steps.

5.1 Data Structure

We maintain two data structures: the *workspace*, in which we make sure that movable obstacles will not collide with other movable or static obstacles, and the *configuration space* of the robot, in which we keep track of the connected component the robot is in.

As both the robot and the obstacles are axis-aligned rectangles that translate, the C-obstacles the movable obstacles induce are also axis aligned rectangles. Now, we look at all vertical and horizontal lines which are incident to the boundaries of the workspace obstacles and the C-obstacles, and use these lines to form which we call a *rectangular map* for the workspace and the configuration space of the robot, respectively (see Fig. 5).

For both the workspace and the configuration space of the robot, we store two lists of lines: one list for horizontal lines and one list for vertical lines. Both of these lists are *ordered*, the vertical lines from left to right and the horizontal lines from bottom to top. With the lines, we store their coordinate, to which obstacle it belongs, and whether it is the bottom or top line (horizontal lines), or the left or right line (vertical lines) of the particular obstacle.

For the configuration space of the robot, we maintain some additional information in the data structure. The rows and columns in between the lines are overlapping zero or more obstacles. If, for a horizontal row, the bottom line of an obstacle is below the row, and the top line of the obstacle is above the row, the row contains the particular obstacle. We store these associations with the rows and columns. In Fig. 5b, we encoded these associations using three bits, one bit for each obstacle. If first

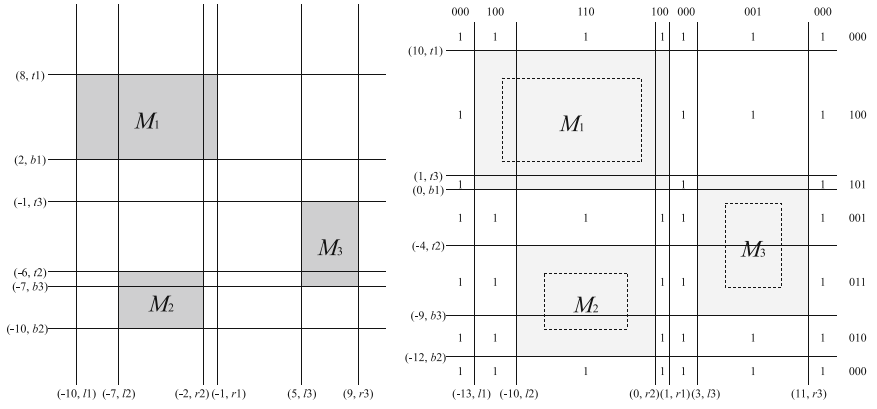


Fig. 5. (a) The rectangular map of the workspace. The coordinate and the type (l_i, r_i, b_i and t_i mean the left, right. bottom and top line of obstacle M_i, respectively) are stored with the horizontal and vertical lines. **(b)** The rectangular map of the configuration space of the robot. Here, we additionally store with the rows and columns what obstacles they overlap (see right and top), and for each empty cell in the map whether or not it belongs to the robot's connected component.

bit is 1, then the row (or column) contains the obstacle M_1. If the second bit is 1, it also contains obstacle M_2, etc. Now, to determine the status of a cell in the rectangular map, we can simply "and" the bits stored at the row and the column of that particular cell. If the result is 0, the cell is free, otherwise, the bits determine by which obstacles the cell is occupied.

In addition, we maintain in the configuration space data structure in which connected component the robot is. Given the initial position of the robot, we can quickly find in which cell of the (configuration space) map it is. Then, we "flood fill" the empty cells of the rectangular map from the cell containing the robot's position, and store with each free cell a flag indicating whether or not the cell belongs to the robot's connected component. In the example of Fig. 5b, all free cells belong to the robot's connected component, as there is only one connected component.

5.2 Events

Given the initial configuration space and workspace as constructed as explained above, we can manipulate the environment by moving the movable obstacles. While doing so, we need to keep track of the changes made in the configuration space data structure and the workspace data structure. We can only move an obstacle if it is manipulable. This is when in the rectangular map of the robot's configuration space, a cell in the connected component of the robot is adjacent to a cell occupied by the C-obstacle of the movable obstacle. Suppose we have selected a movable obstacle to be moved along some straight line. While moving the obstacle, the coordinates

of the horizontal lines and the vertical lines associated with the obstacle in both the configuration space and the workspace change. As long as the ordering of the vertical and horizontal lines remains the same, we only need to update the coordinates stored at the vertical and horizontal lines. However, at the moment two vertical lines or two horizontal lines swap position in either the workspace or the configuration space, we need to (1) check if that swap is allowed (does the obstacle lose its manipulability? does the obstacle collide with another obstacle?), and if so (2) update the the data structure.

Given a proposed straight-line movement of an obstacle M_i, we can compute using the coordinates of the lines what the first *event* is that will be encountered, i.e. the first occasion in which we potentially swap two lines. Below we iterate over all possible events the can be encountered, and show for each of them how we check whether the event is allowed. We only discuss events that occur when M_i moves directly to the *right* (all other directions can be handled symmetrically). So, each event involves the left or the right line of obstacle M_i, and the left or the right line of another obstacle, say M_j, in either the workspace or the configuration space (C-space) of the robot.

- *Workspace*: *left* line of M_i, *left* line of M_j. The event is always allowed.
- *Workspace*: *left* line of M_i, *right* line of M_j. The event is always allowed.
- *Workspace*: *right* line of M_i, *left* line of M_j. The event potentially causes M_i to start colliding with M_j. The event is allowed if the bottom line of M_i is above the top line of M_j, or if the top line of M_i is below the bottom line of M_j.
- *Workspace*: *right* line of M_i, *right* line of M_j. The event is always allowed.
- *C-space*: *left* line of M_i, *left* line of M_j. The event potentially causes M_i to stop being manipulable (similar to the situation of Fig. 3a). The event is *not* allowed if all free cells of the connected component of the robot that are adjacent to the C-obstacle of M_i are below the top line of M_j, above the bottom line of M_j, and left of the left line of M_j.
- *C-space*: *left* line of M_i, *right* line of M_j. The event is always allowed.
- *C-space*: *right* line of M_i, *left* line of M_j. The event potentially causes the connected component of of the robot to disappear (similar to the situation of Fig. 3b). The event is *not* allowed if all free cells of the connected component of the robot that are adjacent to the C-obstacle of M_i are below the top line of M_j, above the bottom line of M_j, and right of the right line of M_i. The event may also cause a connected component *split* (similar to the situation of Fig. 4a); we will discuss this below.
- *C-space*: *right* line of M_i, *right* line of M_j. The event is always allowed.

If the event we encountered is allowed, we need to update the data structure. If the event happened in the workspace, we only need to swap the involved lines in the ordered list of lines stored in the data structure, and update the coordinate of the lines of the moved obstacle M_i.

If the event happened in the configuration space of the robot, we first swap the involved lines in the configuration space data structure, and update the coordinate of the lines of M_i. Second we need to update the information stored with the column

Algorithm 4. EXPAND$(x = \langle c_1, \ldots, c_n, N \rangle) : X$

1: **for** K times **do**
2: Pick a random movable obstacle M_i that is manipulable in x.
3: Pick a random direction $\theta \in \{0, \pi/2, \pi, 3\pi/2\}$, and a random distance r.
4: $x \leftarrow$ MOVEOBSTACLE$(x, M_i, c_i + (r\cos\theta, r\sin\theta))$.
5: **end for**
6: **return** x.

(or row) that is in between the two lines that have swapped. Given the bit-representation, we simply need to "x-or" the stored value with the bits of both of the involved obstacles M_i and M_j.

Further, we need to update the component information of the free cells; i.e. set the flag whether or not they belong to the connected component of the robot. Only the cells in the column between the two swapped lines may have changed status from "occupied by a C-obstacle" to "free" or vice versa. Initially, we set the flag of all cells in that column to 0 (i.e. not belonging to the robot's connected component). Then we use a flood fill from free cells flagged 1 that neighbor the column to set all the flags of the free cells belonging to the robot's connected component. However, if the event involves the right line of M_i, and the left line of M_j, the connected component of the robot may have split into two components (which would both be flagged 1 after the above flood-fill). Whether this is the case can be checked using another flood fill. If the connected component has indeed split, the robot can in the new situation be in either of the newly created components. In our algorithm, we randomly pick one, and set the flags of the cells in the other component to 0.

After the event has been handled, we can compute what the next event is that is encountered when moving M_i. This repeats until M_i has reached its destination, or until an event is encountered that is not allowed.

5.3 Algorithm

We implemented the algorithm of Section 4.2 using the data structure presented above. For each state in the tree, we store both the rectangular map of the workspace and the rectangular map of the configuration space of the robot. As this is quite memory-intensive —the space complexity of the data structure is $O(n^2)$— we slightly changed Algorithm 2 (see Algorithm 4). Instead of storing the state after each obstacle movement, we let the state be expanded by a random sequence of K obstacle movements, where K is a parameter of the algorithm. This does not affect the probabilistic completeness: Theorem 4.1 also holds for a sequence of sequences of actions. Further, also without loss of probabilistic completeness, we let the obstacles only move along axis-aligned paths (see line 3 of Algorithm 4).

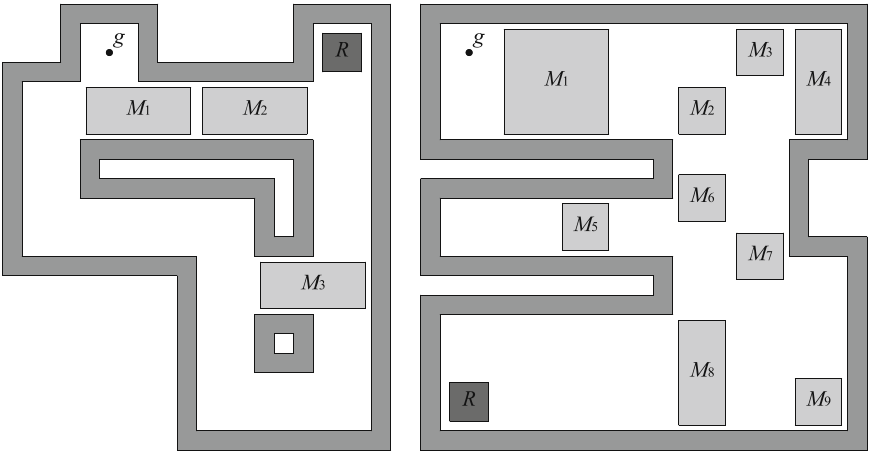


Fig. 6. The initial situation of two scenarios A and B we perform experiments on. The movable obstacles are light gray, the static obstacles are dark gray, and the robot is even darker gray. The robot is shown in its start configuration.

6 Results

As a proof-of-concept, we report results of experiments performed on two scenarios (see Fig. 6). Scenario A only contains three movable obstacles, but is a difficult problem mainly because of the "lock" created by obstacle M_3. This problem cannot be solved by the algorithm of [15] since it belongs to the class of non-monotone problems. The planner of [10] does not take into account indirect obstacle interactions, and therefore would not consider moving obstacle M_2 before moving through the lock of obstacle M_3. Scenario B is more complex in the sense that it contains more obstacles. It is an axis-aligned version of the problem experimented on in [15]. The big obstacle M_1 must be moved out of the way of the robot, but before this is possible other obstacles have to be moved out of the way of M_1 first.

We performed our experiments on a 1.66 GHz Intel T5500 processor with 1 GByte of memory. Our algorithm solves scenario A in 0.01 seconds, and scenario B in 0.38 seconds. The solutions contain 214 and 23899 obstacle movements, respectively.

The solutions produced by our algorithm are not optimal; they include large amounts of unnecessary obstacle movements. This is because of the purely random nature of our algorithm. However, the solutions are found very fast (although the comparison is not entirely fair, [15] reports a running time of

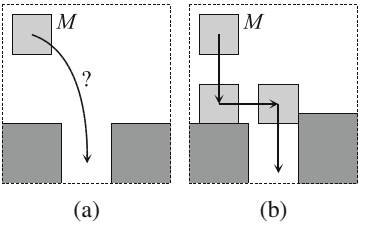


Fig. 7. (a) A narrow passage for movable obstacle M formed by two static obstacles. **(b)** M bumps against the obstacles to find its way through the narrow passage.

2.08 seconds on scenario B). This is explained by the efficiency of the rectangular map data structure, and our algorithm's inherent advantage in handling narrow passages: As the movable obstacles only move along axis-aligned paths, they implicitly use *compliance* to the static obstacles (see Fig. 7). Even if the boundaries of two (static) obstacles are collinear (Fig. 7a), there is an explicit ordering of the lines in the rectangular map data structure, so one is above the other. This is topologically equivalent to the situation of Fig. 7b, which the movable obstacles exploit to find their way through narrow passages.

7 Conclusion and Future Work

In this paper, we have discussed the problem of path planning among movable obstacles. We have made the observation that if we maintain an exact representation of the configuration space of the robot and the connected component the robot is in, we can *decouple* the computation of the obstacle movements, and the robot motions that lead to these obstacle movements. This approach to the problem enabled us to devise the first *probabilistically complete* algorithm for this domain.

We have presented a data structure called the *rectangular map* to maintain an exact representation of the robot's configuration space in case all obstacles and the robot are translating axis-aligned rectangles. We have implemented the algorithm and the data structure, and used it to solve problems that could not be solved by previous work.

The requirement to maintain an explicit representation of the robot's configuration space limits the practical applicability of our algorithm to robots with two or three degrees of freedom. Note, however, that the number of degrees of freedom of the movable obstacles is not constrained. Future work includes the implementation of a data structure to maintain the robot's configuration space in the more general case of polygonal and circular obstacles that can both translate and rotate. The arrangement package of CGAL [18] may provide most of the functionality required. Another possibility to address this limitation is to maintain the robot's connected component and its connectivity using sampling-based techniques, without sacrificing probabilistic completeness. This remains subject of future study.

The fact that our algorithm is probabilistically complete shows that we have characterized the problem correctly, but it does not necessarily say much about the performance of the algorithm. In fact, the algorithm that we have presented performs a rather uninformed brute force search. Enhancing the algorithm with *heuristics* to focus the search, such as the ones used in [10, 14, 15], might drastically improve the performance without losing probabilistic completeness. It may improve the quality of the produced solutions as well.

References

1. Alami, R., Laumond, J.-P., Siméon, T.: Two manipulation planning algorithms. In: Proc. Workshop on Algorithmic Foundations of Robotics, pp. 109–125 (1995)
2. Ben-Shahar, O., Rivlin, E.: Practical pushing planning for rearrangement tasks. IEEE Trans. on Robotics and Automation 14(4), 549–565

3. Canny, J.: The complexity of robot motion planning. MIT Press, Cambridge (1987)
4. Chadzelek, T., Eckstein, J., Schömer, E.: Heuristic motion planning with movable obstacles. In: Proc. Canadian Conf. on Computational Geometry, pp. 211–219 (2000)
5. Chen, P., Hwang, Y.: Practical path planning among movable obstacles. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 444–449 (1991)
6. Demaine, E., Demaine, M., O'Rourke, J.: Pushpush and Push-1 are NP-hard in 2D. In: Proc. Canadian Conf. on Computational Geometry, pp. 211–219 (2000)
7. Hsu, D., Kindel, R., Latombe, J.-C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. of Robotics Research 21(3), 233–255 (2002)
8. Kavraki, L., Švestka, P., Latombe, J.-C., Overmars, M.: Probabilistic roadmaps for path planning in high dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
9. LaValle, S., Kuffner, J.: Rapidly-exploring random trees: progress and prospects. In: Proc. Workshop on Algorithmic Foundations of Robotics (2000)
10. Nieuwenhuisen, D., van der Stappen, F., Overmars, M.: An effective framework for path planning amidst movable obstacles. In: Proc. Workshop on Algorithmic Foundations of Robotics (2006)
11. Ota, J.: Rearrangement of multiple movable objects. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1962–1967 (2004)
12. Sánchez, G., Latombe, J.-C.: Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2112–2119 (2002)
13. Siméon, T., Laumond, J.-P., Cortés, J., Sahbani, A.: Manipulation planning with probabilistic roadmaps. Int. J. of Robotics Research 23(7-8), 729–746 (2004)
14. Stilman, M., Kuffner, J.: Navigation among movable obstacles: real-time reasoning in complex environments. Int. J. of Humanoid Robotics 2(4), 479–504 (2005)
15. Stilman, M., Kuffner, J.: Planning among movable obstacles with artificial constraints. In: Proc. Workshop on Algorithmic Foundations of Robotics (2006)
16. Stilman, M., Schamburek, J.-U., Kuffner, J., Asfour, T.: Manipulation planning among movable obstacles. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 3327–3332 (2007)
17. Švestka, P., Overmars, M.: Coordinated path planning for multiple robots. Robotics and Autonomous Systems 23(3), 125–152 (1998)
18. Wein, R., Fogel, E., Zukerman, B., Halperin, D.: 2D Arrangements. CGAL User and Reference Manual, ch. 20 (2007)
19. Wilfong, G.: Motion planning in the presence of movable obstacles. Annals of Mathematics and Artificial Intelligence 3, 131–150 (1991)

Multi-modal Motion Planning in Non-expansive Spaces

Kris Hauser and Jean-Claude Latombe

Abstract. The motion planning problems encountered in manipulation and legged locomotion have a distinctive multi-modal structure, where the space of feasible configurations consists of overlapping submanifolds of different dimensionalities. Such a feasible space does not possess expansiveness, a property that characterizes whether planning queries can be solved with traditional sample-based planners. We present a new sample-based multi-modal planning algorithm and analyze its completeness properties. In particular, it converges quickly when each mode is expansive relative to the submanifold in which it is embedded. We also present a variant that has the same convergence properties, but works better for problems with a large number of modes by considering subsets that are likely to contain a solution path. These algorithms are demonstrated in a legged locomotion planner.

1 Introduction

Probabilistic roadmap (PRM) planners (Chapter 7 of [4]) are state-of-the-art approaches for motion planning in high-dimensional configuration spaces under geometrically complex feasibility constraints. They approximate the connectivity of the feasible set F using a network of randomly sampled configurations connected by straight line paths. It is widely known that PRMs can be slow when F has poor *expansiveness* [11], or, informally, contains "narrow passages" [9]. In certain *non-expansive* spaces, where F consists of overlapping submanifolds of varying dimensionality, the narrow passages are arbitrarily thin, and PRMs do not work at all.

This type of *multi-modal* structure happens in motion planning problems found in manipulation and legged locomotion. Here, each submanifold in F corresponds to a *mode*, a fixed set of contact points maintained between the robot and its environment. The planner must choose a discrete sequence of modes (a sequence of

Kris Hauser and Jean-Claude Latombe
Department of Computer Science, Stanford University, Stanford CA 94305
e-mail: {khauser,latombe}@cs.stanford.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 615–630.
springerlink.com

contacts to make and break), as well as continuous single-mode paths through them (the joint space motions to achieve those changes of contacts).

In problems where each mode is low-dimensional, such as planar manipulation of multiple objects [1, 15], the primary challenge lies in the combinatorial complexity of mode sequencing. But in problems with high-dimensional modes, the geometric complexity of single-mode planning poses an additional challenge. Complete single-mode planning is intractable, so PRMs are often used for such planning. This approach has made it possible to solve several specific problems in manipulation with grasps and regrasps [14, 16] and legged locomotion [2, 8]. But what happens to the overall planner's reliability when it is based on a large number of unreliable single-mode PRM queries (thousands or more)? A PRM planner cannot report that no path exists, so when a single-mode query fails, we cannot tell if the query is truly infeasible, or the PRM planner just needed more time. So far, little attention has been paid to the theoretical performance guarantees of these algorithms when applied to general multi-modal problems.

This paper presents MULTI-MODAL-PRM, a general-purpose multi-modal planning algorithm for problems with a finite number of modes, and investigates its theoretical completeness properties. MULTI-MODAL-PRM builds a PRM across modes by sampling configurations in F *and* in the transitions between modes. We prove that, like classical PRM planners, MULTI-MODAL-PRM will eventually find a feasible path if one exists, and convergence is fast as long as each mode is favorably expansive when restricted to its embedded submanifold.

We also present a more practical variant, INCREMENTAL-MMPRM, which searches for a small candidate subset of modes which are likely to contain a solution path, and then restricts MULTI-MODAL-PRM to these modes. INCREMENTAL-MMPRM has the same asymptotic completeness properties as MULTI-MODAL-PRM, but is usually much faster for problems with a large number of modes (which is common). We demonstrate the application of INCREMENTAL-MMPRM in a legged locomotion planner [8], showing that it is indeed reliable.

2 Multi-modal Planning

This section defines multi-modal problems, describes how PRM planners can be used to solve these problems, and outline the pitfalls that make many existing PRM-based planners incomplete.

2.1 PRM Planners and Non-expansive Spaces

PRM planners approximate the connectivity of F, the feasible subset of a robot's configuration space, using a roadmap of configurations (referred to as milestones) connected by simple paths (usually straight-lines). The concept of expansiveness was introduced to characterize how quickly a roadmap converges to an accurate representation [11]. So this paper can be self-contained, we review the basic algorithm and its properties in the Appendix.

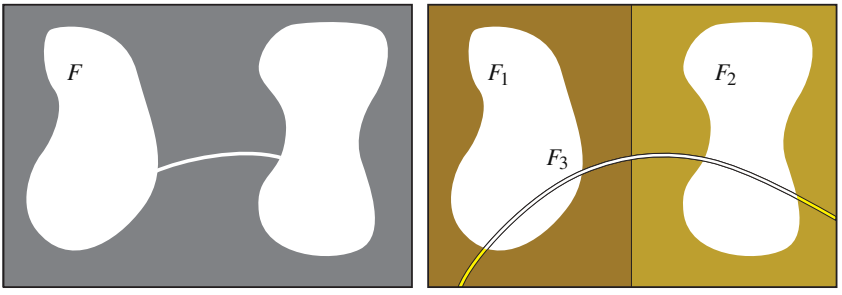


Fig. 1. (a) A non-expansive, multi-modal feasible space F. (b) F can be decomposed into three modes that are individually expansive.

Formally, F is *expansive* if there exist constants $\varepsilon, \alpha, \beta > 0$ such that F is $(\varepsilon, \alpha, \beta)$-expansive (see Appendix). Otherwise, F is non-expansive. In all expansive spaces, the probability that a PRM fails to solve a planning query decreases to zero exponentially as the roadmap grows. However, the convergence rate can become arbitrarily slow as ε, α, and β approach zero. An expansive space is poorly expansive (i.e., ε, α, and β are low) if it contains narrow passages. Fortunately, smoothed analysis shows that narrow passages are unstable with respect to small perturbations of the input geometry, and are therefore unlikely to occur (except by design) [3].

However, some non-expansive spaces do exist, not by chance, but rather for structural reasons. If F contains a cusp, it is non-expansive, but PRMs might still work well everywhere away from the cusp, since removing a tiny region around the cusp makes F expansive without changing its connectivity. But if F contains regions of varying dimensionality, then PRMs have probability zero of answering most queries. For example, F may consist of 2D regions connected by 1D curves (Figure 1a). In these spaces, the PRM convergence bounds take the meaningless value of 1.

Varying dimensionality is *inherent* in the structure of multi-modal problems. In these problems, the submanifolds whose union forms F can be enumerated from the problem definition, e.g., by considering all possible combinations of contacts. However, their number may be huge.

2.2 Multi-modal Problem Definition

The robotic system moves between a finite set of modes, Σ. Each mode $\sigma \in \Sigma$ is assigned a mode-specific feasible space F_σ, and $F = \bigcup_{\sigma \in \Sigma} F_\sigma$ (Figure 1b). The feasibility constraints of a mode can be divided into two classes.

- *Dimensionality-reducing* constraints, often represented as functional equalities $C_\sigma(q) = 0$. They define the submanifold C_σ as the set of configurations that satisfy these constraints.

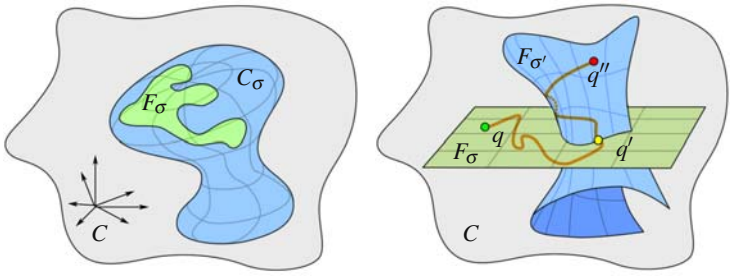


Fig. 2. (a) At a mode σ, motion is constrained to a subset F_σ of a submanifold C_σ with lower dimension than C. (b) To move from configuration q at stance σ to q'' at an adjacent stance σ', the motion must pass through a transition configuration q'.

- *Volume-reducing* constraints, often represented as inequalities $D_\sigma(q) > 0$. These may cause F_σ to be empty. Otherwise, F_σ has the same dimension as C_σ but lower volume (Figure 2a).

For example, in legged locomotion, σ is a fixed set of footfalls. F can be embedded in a configuration space C, which consists of parameters for a free-floating robot base and the robot's joint angles. Enforcing contact at the footfalls imposes multiple closed-loop kinematic constraints, which in turn define C_σ (a submanifold of uniform lower dimension, except at singularities). Collision avoidance and stability constraints are volume-reducing, and restrict F_σ to a subset of C_σ.

2.3 Planning between Two Modes

PRMs can be used for single-mode planning restricted to C_σ. This requires adapting configuration sampling and path segment feasibility testing to the submanifold, since the sampler must have nonzero probability of generating a configuration in F_σ, and a path segment on C_σ may not be a straight line. Two approaches are parameterizing C_σ with an atlas of charts [5], or using numerical methods to move configurations from the ambient space onto C_σ [13]. A PRM planner will plan quickly as long as F_σ is expansive (with C_σ taken as the ambient space).

Suppose the robotic system is at configuration q in mode σ. To switch to a new mode σ', it must plan a path in F_σ that ends in a configuration q' in $F_\sigma \cap F_{\sigma'}$ (Figure 2b). The region $F_\sigma \cap F_{\sigma'}$ is called the *transition* between σ and σ', and q' is called a *transition configuration*. Transitions are at least as constrained as σ or σ' because they must simultaneously satisfy the constraints of both stances. Hence, they often have zero volume relative to C_σ or $C_{\sigma'}$. Thus, transition configurations should be sampled explicitly from $C_\sigma \cap C_{\sigma'}$. Like single-mode sampling, transition sampling can be performed using explicit parameterization or numerical techniques [6].

The existence of q' is a necessary condition for a single-mode path to connect q and σ', and is also a good indication that a feasible path exists. Sampling q' is also typically faster than planning a single-mode path. These two observations are instrumental in the implementation of INCREMENTAL-MMPRM. However the existence of q' is not sufficient for a path to exist, because q and q' might lie in different connected components of F_σ.

2.4 Planning in Multiple Modes

Because PRMs usually work well for single-mode planning, multi-modal planning is usually addressed by computing several single-mode plans and concatenating them into a multi-modal plan. Given a finite set of modes $\{\sigma_1, \ldots, \sigma_m\}$, a multi-modal planner explores a *mode graph* G. Each vertex in G represents a mode, and an edge connects each pair of *adjacent* modes. We assume we are given a cheap computational *adjacency test*, which tests a necessary condition for the existence of a path moving between any pair of modes σ and σ'. For example, an adjacency test in legged locomotion tests whether σ' adds exactly one footfall within the reach of σ. The test is fast, but prunes out many unnecessary edges from G.

Because G tends to be extremely large, one natural strategy explores only a small part of G. For example, a search-like algorithm incrementally builds a tree T of configurations reachable from the start configuration of the robot. Each expansion step picks a configuration q at mode σ in T, enumerates each adjacent mode σ', then uses a PRM to plan a feasible single-mode path from q to a transition configuration q' in $F_\sigma \cap F_{\sigma'}$. If the outcome is successful, q' is added to T. These steps are repeated until the goal is reached.

2.5 Completeness Challenges in Multi-modal Planning

Special challenges arise when combining several single-mode PRM queries to solve a multi-modal problem. MULTI-MODAL-PRM addresses the following issues, any of which may cause a planner to fail to find a path.

Because any single-mode query might be infeasible, the PRM planner must be terminated with failure after some cutoff time. Set the cutoff too low, and the planner may miss critical paths; too high, and the planner wastes time on infeasible queries. Some prior work tries to avoid infeasible queries [2, 6, 12], allowing the cutoff to be set high. Another approach avoids cutoffs by interleaving computation among queries [6, 14, 15].

Because transitions $F_\sigma \cap F_{\sigma'}$ may have zero volume in F_σ or $F_{\sigma'}$, transition configurations should be sampled explicitly from $C_\sigma \cap C_{\sigma'}$. Furthermore, more than one configuration q' may need to be sampled in each transition $F_\sigma \cap F_{\sigma'}$, because a configuration may lie in a component that is disconnected in F from the start configuration q, or one that is disconnected in F' from a target configuration q''.

3 Multi-Modal-PRM

This section presents the general MULTI-MODAL-PRM algorithm, and gives an overview of its theoretical completeness properties.

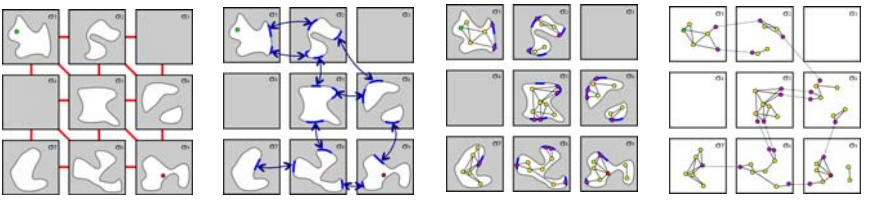
3.1 Algorithm

MULTI-MODAL-PRM builds PRMs across all modes, connecting them at explicitly sampled transition configurations (Figure 3). Suppose there are m modes, $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$. For each $\sigma_i \in \Sigma$ maintain a roadmap R_{σ_i} of F_{σ_i}. Define the sampler SAMPLE-MODE(σ_i) as follows. Uniformly sample a configuration q from C_{σ_i}. If q is in F_{σ_i}, q is returned; if not, SAMPLE-MODE returns failure. Similarly, define SAMPLE-TRANS(σ_i, σ_j) to rejection sample from $F_{\sigma_i} \cap F_{\sigma_j}$. MULTI-MODAL-PRM is defined as follows:

MULTI-MODAL-PRM(q_{start}, q_{goal}, N)
Add q_{start} and q_{goal} as milestones to the roadmaps corresponding to their modes (σ_{start} and σ_{goal}).
Repeat N times:

1. For each mode σ_i, sample a configuration q using SAMPLE-MODE(σ_i). If it succeeds, add q to R_{σ_i} as a new milestone, and connect it to existing milestones.
2. For each pair of adjacent modes σ_i and σ_j, sample a configuration q using SAMPLE-TRANS(σ_i, σ_j). If it succeeds, add q to R_{σ_i} and R_{σ_j}, and connect it to all visible milestones in R_{σ_i} and R_{σ_j}.

Build an aggregate roadmap R by connecting roadmaps at matching transition configurations.
If q_{start} and q_{goal} are connected by a path in R, terminate with success. Otherwise, return failure.



(a) (b) (c) (d)

Fig. 3. Illustrating MULTI-MODAL-PRM on an abstract example problem. (a) A mode graph with nine modes, with adjacent modes connected by lines. (b) The transition regions, with arrows indicating how they map between modes. (c) Building roadmaps. Light dots are milestones sampled from modes, dark ones are sampled from transitions. (d) The aggregate roadmap. Transition configurations connected by dashed lines are identified.

3.2 Summary of Theoretical Results

Section 4 will prove that, under certain conditions, the probability that MULTI-MODAL-PRM incorrectly returns failure when a feasible path actually exists is less than ce^{-dN}, where c and d are positive constants and N is the number of iterations. This means that as more time is spent planning, the probability of failure decreases quickly to zero. The constants c and d do not explicitly depend on the dimensionality of the configuration space, or the total number of modes. Furthermore, since a constant number of samples ($m+n$, where n is the number of adjacencies) are drawn per iteration, MULTI-MODAL-PRM also converges exponentially in the total number of samples drawn.

This bound holds if the following conditions are met:

1. The set of modes $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ is finite.
2. If F_{σ_i} is nonempty, then it is expansive.
3. If F_{σ_i} is nonempty, SAMPLE-MODE(σ_i) succeeds with nonzero probability.
4. If $F_{\sigma_i} \cap F_{\sigma_j}$ is nonempty, SAMPLE-TRANS(σ_i, σ_j) samples each connected component of $F_{\sigma_i} \cap F_{\sigma_j}$ with nonzero probability.

4 Proof of Completeness Properties

This section proves that MULTI-MODAL-PRM is probabilistically complete and exponentially convergent in the number of iterations. The proof shows that three processes exponentially converge: 1) BASIC-PRM under rejection sampling; 2) roadmaps connecting transition components; and 3) roadmaps along any sequence of modes which contains a feasible multi-step path.

4.1 Exponential Convergence

We say a process is *exponentially convergent* in N if the probability of failure is lower than ae^{-bN} for positive constants a and b. A useful *composition principle* allows us to avoid stating the coefficients a and b, which become cumbersome. If two quantities are subject to exponentially decreasing upper bounds, their sums and products are themselves subject to exponentially decreasing bounds. The product is trivial, and $a_1e^{-b_1N} + a_2e^{-b_2N}$ is upper bounded by ae^{-bN} for $a = (a_1 + a_2)$ and $b = \min b_1, b_2$. So if two processes are exponentially convergent in N, then the probability that both of them succeed, or either one succeeds, also converges exponentially in N.

4.2 PRMs Converge under Rejection Sampling

The first lemma states that PRMs converge exponentially, not just in the number of milestones (as was proven in [11]), but also in the number of rejection samples drawn. Let p be the probability that a random sample from C lies in F. From N

configurations uniformly sampled at randomfrom C, let the milestones M be those that are feasible.

Lemma 4.1. *If F is expansive and $p > 0$, the probability that a roadmap R constructed from M connects two query configurations q_1 and q_2 converges to 1 exponentially in N.*

Proof. Since the configuration samples are independent, the size of M is binomially distributed. Hoeffding's inequality gives an upper bound to the probability that M has n or fewer milestones:

$$Pr(|M| \leq n) \leq \exp(-2(Np-n)^2/N).$$

If $|M| \leq n$, the probability of failure is at most 1. On the other hand, if $|M| > n$, the probability that R fails to connect q_1 and q_2 is less than ce^{-dn}, where c and d are the constants in Theorem 7.1 (see Appendix). Since these events are mutually exclusive, the overall probability of failure ν is bounded as follows:

$$\begin{aligned} \nu \leq & Pr(|M| \leq n) \cdot 1 + Pr(|M| > n) c \exp(-dn) \\ \leq & \exp(-Np^2/2) + c\exp(-Ndp/2) \end{aligned} \tag{1}$$

where we have set $n = pN/2$. Using the composition principle, this bound is exponentially decreasing in N. □

4.3 Convergence of Connecting Transition Components

Here we consider the probability of finding a path in F between arbitrary connected subsets A and B, by building a roadmap using N rejection samples in A, B, and F.

Let M and p be defined as in Lemma 4.1. Let us rejection sample A and B respectively by drawing samples uniformly from supersets A' and B', with probability of success p_A and p_B. From N configurations sampled from A', let M_A be the set of milestones in A. Define M_B similarly.

Lemma 4.2. *Assume F is expansive, and p, p_A, and p_B are nonzero. Let R be the roadmap constructed from all milestones M, M_A, and M_B. If A and B are in the same connected component in F, then the probability that R contains a path between A and B converges to 1 exponentially in N.*

Proof. View any pair of milestones q_A in M_A and q_B in M_B as PRM query configurations. Then R contains a path between A and B when M_A is nonempty (event X), M_B is nonempty (event Y), and the roadmap formed from M connects q_A and q_B (event Z).

The probability that N rejection samples from A' fails to find a milestone in A is at most $(1-p_A)^N \leq e^{-Np_A}$. So event X is exponentially convergent. The same holds for Y. Finally, event Z is exponentially convergent in N by Lemma 4.1 and since q_A and q_B are in the same connected component. The lemma holds by applying the composition principle to X, Y, and Z. □

4.4 Convergence of Multi-Modal-PRM

Lemma 4.3. *Let the assumptions at the end of Section 3 hold. Between any two configurations, if* any *feasible multi-step path exists, there exists* some *feasible path that makes a finite number of mode switches.*

Proof. Expansiveness implies ε-goodness, which implies that each connected component of the feasible space has volume at least $\varepsilon > 0$. Let ε_0 be such that each mode is ε_0-good. Then, each mode can only contain $1/\varepsilon_0$ components, with m/ε_0 components overall. □

Theorem 4.1. *Let the assumptions at the end of Section 3 hold. If q_s and q_g can be connected with a feasible multi-step path, then the probability that* MULTI-MODAL-PRM *finds a path converges to 1 exponentially in the number of iterations N.*

Proof. If q_s and q_g can be connected with a feasible path, there is a feasible path with a finite number of mode switches. Suppose this path travels through mode σ_k, starting at transition connected component A and ending at B. SAMPLE-MODE and SAMPLE-TRANS have properties allowing Lemma 4.2 to be applied to roadmap R_{σ_k}. Therefore, the probability that R_{σ_k} connects A and B exponentially converges to 1. The theorem follows from repeating this argument for all modes along the path, and using the composition principle a finite number of times. □

Dimensionality of and the total number of modes m do not explicitly affect the coefficients in the convergence bound (although the total cost per iteration is at least linear in m). The modes' expansiveness measures and the parameters p and p' have a straightforward effect on the convergence rate: when expansiveness or the parameters increase, the bound moves closer to zero. The bound also moves closer to zero if fewer modes are needed to reach the goal, or if the goal can be reached via multiple paths.

5 An Incremental Variant

Even executing a few iterations of MULTI-MODAL-PRM is impractical if the number of modes is large. Typical legged locomotion queries have over a billion modes, but only tens or hundreds of steps are needed to go anywhere within the range of visual sensing. The INCREMENTAL-MMPRM variant uses heuristics to produce a small subset of modes that are likely to contain a path to the goal. Limiting planning to these modes make planning much faster. But heuristics are not always right, so INCREMENTAL-MMPRM is designed to gracefully degrade back to MULTI-MODAL-PRM if necessary.

5.1 Overview

INCREMENTAL-MMPRM alternates between *refinement* and *expansion*. At each round r, it restricts itself to building roadmaps over a candidate set of modes Σ_r. We

set Σ_0 to the empty set, and all single-mode roadmaps (except the start and goal) are initially empty. The algorithm repeats the following for rounds $r = 1, \ldots, Q$:

1. *Expansion*. Add new modes to Σ_{r-1} to produce the next candidate mode set Σ_r.
2. *Refinement*. Incrementally build roadmaps in Σ_r by performing n_r mode and transition samples (e.g., one iteration of MULTI-MODAL-PRM).

The performance of INCREMENTAL-MMPRM depends mainly on the expansion heuristic. The heuristic does not affect asymptotic convergence, as long as the candidate mode set grows until it cannot expand any further (at which point INCREMENTAL-MMPRM behaves exactly like MULTI-MODAL-PRM). But practically, it has a large impact on running time. The heuristic below can be applied to any multi-modal planning problem, and improves running time by orders of magnitude.

5.2 Expansion: Search among Feasible Transitions

In virtually all multi-modal problems we have studied, one may consider a huge number of modes and transitions, but *most* of them are actually infeasible, i.e., have empty feasible spaces. Trying to sample such spaces would cause the planner to waste a considerable amount of time. Instead, iff the expansion step produces candidate modes that are likely to contain a feasible path, overall planning speed will be improved, even if expansion incurs some additional computational expense.

Search among feasible transitions (SAFT) uses the existence of a feasible transition as a good indication that a feasible path exists as well (as in [2]). SAFT incrementally builds a mode graph G, but without expanding an edge until it samples a feasible transition configuration. To avoid missing transitions with low volume, SAFT interleaves transition sampling between modes (as in [6]). It maintains a list A of "active" transitions. Each transition T in A has an associated priority $p(T)$, which decreased as more time is spent sampling T. The full algorithm is as follows:

SAFT-INIT (performed only once at round 0)
Add the start mode σ_{start} to G. Initialize A to contain all transitions out of σ_{start}.

SAFT-EXPAND(r) (used on expansion round r)
Repeat the following:

1. Remove a transition T from A with maximum priority. Suppose T is a transition from σ to σ'. Try to sample a configuration in $F_\sigma \cap F_{\sigma'}$.
2. On failure, reduce the priority $p(T)$ and reinsert T into A.
3. On success, add σ' to the mode graph G. For each transitions T' out of σ', add T' to A with initial priority $p(T')$.

Repeat until G contains a sequence of modes (not already existing in Σ_{r-1}) connecting σ_{start} to σ_{goal}. Add these modes to Σ_{r-1} to produce Σ_r.

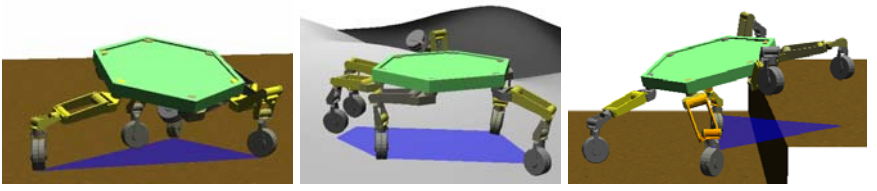


Fig. 4. Three terrains for testing the locomotion planner.

As in [6], SAFT-EXPAND may find paths more quickly if the initial $p(T)$ estimates the probability that T is nonempty. Other heuristics, such as a bias toward easier steps, could also be incorporated into $p(T)$ [8].

5.3 Refinement: Strategies to Improve Connectivity

The running time of INCREMENTAL-MMPRM is also substantially affected by the refinement strategy. First, if the sample count parameter n_r is too high, the planner might waste time on a candidate set of modes that contains no feasible path; too low, and the planner will expand the candidate mode set unnecessarily. We set n_r proportional to the number of modes, with some minor tuning of the proportionality constant.

Second, given a fixed n_r, the planner should allocate single-mode planning computations to maximize its chance of finding a feasible path. If a mode's roadmap is already highly connected, then more samples are unlikely to improve connectivity, so we bias sampling toward modes with poorly connected roadmaps. Furthermore, if the aggregate roadmap R already contains paths connecting two modes separated by σ, additional planning in σ is unlikely to improve connectivity. Thus, we bias sampling toward modes that could potentially connect large components of R.

Also, rather than use BASIC-PRM for single-mode planning, we use the SBL variant, which is much faster in practice [17]. SBL grows trees rooted from transition configurations, and delays checking the feasibility of straight line paths.

6 Experiments in a Legged Locomotion Planner

We use legged locomotion as a case study to demonstrate that MULTI-MODAL-PRM improves reliability over a prior incomplete approach. See [8] for the complete problem specification and implementation.

Our previous studies used a two-phase algorithm, here called SINGLE-TRANS. The first phase performs exploration of the mode graph (and is nearly identical to SAFT). It searches the mode graph for a sequence of footsteps to take, and a single feasible transition configuration for each step. In a second phase, single-mode paths are planned between subsequent transition configurations. If single-mode planning

Table 1. Locomotion planning statistics on various problems with a fixed mode sequence, averaged over 10 runs. Standard deviations in parentheses.

Problem	Method	% successful	Time (s)
Flat	Single trans	0%	501 (88.3)
	MMPRM	90%	409 (180)
	I-MMPRM	100%	106 (36.6)
Hills	Single trans	0%	149 (29.5)
	MMPRM	100%	181 (103)
	I-MMPRM	100%	99.5 (28.3)
Stair	Single trans	0%	258 (36.9)
	MMPRM	100%	507 (197)
	I-MMPRM	100%	347 (138)

fails, it returns to exploration. SINGLE-TRANS solved many challenging problems for a four-limbed robot [2] and a bipedal robot [7].

We attempted to apply SINGLE-TRANS to NASA's ATHLETE robot, a six-legged lunar vehicle with 36 revolute joints. We found that even on seemingly simple problems, the single-mode planning phase often failed, forcing the algorithm to return often to exploration. Each exploration usually takes several minutes, leading the overall planner to have a long running time, with high variability between runs. By contrast, our experiments here show that MULTI-MODAL-PRM is usually able to find a path after only one exploration phase.

We compare SINGLE-TRANS (Single Trans), MULTI-MODAL-PRM (MMPRM), and INCREMENTAL-MMPRM (I-MMPRM) using the connection strategy outlined in Section 5.3 on the three test terrains of Figure 4: flat ground (Flat), a smoothly undulating terrain (Hills), and a stair step with a height of 0.5 times the diameter of ATHLETE's chassis (Stair). To avoid variability of running time due to the the choice of footsteps, we restrict each planner to a single sequence of modes Σ which is known to contain a feasible path (so in essence, the mode graph is a single string of modes). These sequences were computed by running INCREMENTAL-MMPRM and extracting the modes along the path. Σ ranges in size from 14 to 37 modes. Note that our use of the restricted mode graph obscures the major benefit of I-MMPRM (again, restricting planning to a small subset of the mode graph). It is nonetheless necessary for comparison purposes, because MMPRM would not be able to even fit the original mode graph in memory. We tested each method 10 times on identical Σ but with different random seeds, and terminated the planner if no path was found after 30,000 total samples.

Table 1 reports the results. The single-transition method fails in every case. The methods based on MULTI-MODAL-PRM successfully find a path within the iteration limit nearly every time. I-MMPRM is faster than MMPRM, especially when the steps have varying degrees of difficulty. On flat ground, Σ contained one particularly difficult step (depicted in Figure 4), and hence I-MMPRM was over four times faster.

The failure of the single-transition method is consistent with the explanation that, by accident, transition configurations were sampled in disconnected components, which of course cannot be connected with single-mode paths. The MULTI-MODAL-PRM-based methods succeed by sampling multiple configurations in each transition (typically around 10 before finding a solution).

In similar experiments with a bipedal humanoid robot [7], SINGLE-TRANS worked fairly consistently, with an 80-90% success rate. MMPRM and I-MMPRM were again consistently successful, but needed very few configurations per transition (typically 1 or 2 before finding a solution). This modest reliability increase came at a moderate computational overhead; I-MMPRM averaged about twice as long as SINGLE-TRANS at finding a feasible path.

7 Conclusion

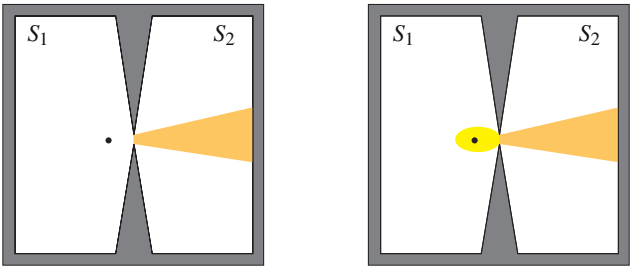
This paper presented MULTI-MODAL-PRM, a new sample-based multi-modal planning algorithm for problems with a finite number of modes. We proved that MULTI-MODAL-PRM has probability of failure exponentially converging to 0 in the number of samples drawn, if the feasible space of each mode is expansive relative to its embedded submanifold. We also presented a variant, INCREMENTAL-MMPRM, which restricts planning to an incrementally growing set of candidate modes, and is orders of magnitude faster than MULTI-MODAL-PRM in problems with large numbers of modes. We demonstrated the reliability of these techniques in experiments in a locomotion planner.

When applied to the ATHLETE six-legged robot, MULTI-MODAL-PRM dramatically improves the planner's reliability over a simpler incomplete method. When applied to a humanoid biped, MULTI-MODAL-PRM modestly improves reliability with moderate overhead. These results suggest that disconnected feasible spaces occurred more frequently in ATHLETE than in the biped. Future work might aim to discover the causes of this phenomenon. We suspect characteristics of ATHLETE's kinematics (e.g., singularities) or its larger number of legs in contact might be contributing factors.

Most systems with contact are posed as having a continuous (uncountably infinite) number of modes, which are then discretized for planning [8, 15, 16]. The completeness results in this paper hold only if the set of discretized modes contains a solution path. If no such path exists, the modes may have been discretized poorly. In future work, we hope to investigate the completeness and convergence rate of multi-modal planners that sample modes from continuous sets of modes.

Appendix

This appendix reviews the basic PRM algorithm and its theoretical completeness properties in expansive spaces.



(a) (b)

Fig. 5. A poorly expansive space. (a) The visibility set $V(q)$ in region S_2, of a point q in S_1. (b) $\text{LOOKOUT}_\beta(S_1)$ is the set of points that see at least a β fraction of S_2.

A.1 Basic Algorithm

PRM planners address the problem of connecting two configurations q_{start} and q_{goal} in the feasible space F, a subset of configuration space C. Though it is prohibitively expensive to compute an exact representation of F, feasibility tests are usually cheap. So to approximate the connectivity of F, PRM planners build a *roadmap* R, a network of feasible configurations (called *milestones*) connected with straight-line segments. A basic algorithm operates as follows:

BASIC-PRM(q_{start}, q_{goal}, n)
Add q_{start} and q_{goal} to R as milestones.
Repeat n times:

1. Sample a configuration q uniformly from C, and test its feasibility. Repeat until a feasible sample is found.
2. Add q to R as a new milestone. Connect it to nearby milestones q' in R if the line segment between q and q' lies in F.

If R contains a path between q_{start} and q_{goal}, return the path.
Otherwise, return 'failure'.

If a PRM planner produces a path successfully, the path is guaranteed to be feasible. If it fails, then we cannot tell whether no path exists or the cutoff n was set too low.

A.2 Performance in Expansive Spaces

BASIC-PRM and several variants have been shown to be probabilistically complete, that is, the probability of incorrectly returning failure approaches 0 as n increases. One particularly strong completeness theorem proves that PRMs converge exponentially, given that F is *expansive* [11].

The notion of expansiveness expresses the difficulty of constructing a roadmap that captures the connectivity of F. The success of PRMs in high dimensional spaces

is partially explained by the fact that expansiveness is not explicitly dependent on the dimensionality of F. Let $\mu(S)$ measure the volume of any subset $S \subseteq F$ (with $\mu(F)$ finite), and let $V(q)$ be the set of all points that can be connected to q with a straight line in F. The *lookout* set of a subset S of F is defined as the subset of S that can "see" a substantial portion of the complement of S (see Figure 5). Formally, given a constant $\beta \in (0,1]$ and a subset S of a connected component F' in F, define

$$\text{LOOKOUT}_\beta(S) = \{q \in S \quad | \quad \mu(V(q) \setminus S) \geq \beta\mu(F' \setminus S)\}$$

For constants ε, α, $\beta \in (0,1]$, F is said to be $(\varepsilon,\alpha,\beta)$-*expansive* if:

1. For all $q \in F$, $\mu(V(q)) \geq \varepsilon\mu(F)$.
2. For any connected subset S, $\mu(\text{LOOKOUT}_\beta(S)) \geq \alpha\mu(S)$.

The first property is known as ε-goodness, and states that each configuration "sees" a significant fraction of F. The second property can be interpreted as follows. View S as the visibility set of a single roadmap component R'. Let F' be the component of feasible space in which S lies. Then, with significant probability (at least $\alpha\mu(S)$), a random configuration will simultaneously connect to R' and significantly reduce the fraction of F' not visible to R' (by at least β).

The primary convergence result of [11] can be restated as follows:

Theorem 7.1. *If F is* $(\varepsilon,\alpha,\beta)$-expansive*, then the probability that a roadmap of n uniformly, independently sampled milestones fails to connect q_{start} and q_{goal} is no more than ce^{-dn} for some positive constants c and d.*

The constants c and d are simple functions of ε, α, and β. If F is favorably expansive (ε, α, and β are high), the bound is close to zero, and BASIC-PRM will find a path between q_{start} and q_{goal} relatively quickly. If, on the other hand, F is poorly expansive (ε, α, and β are low), then PRM performance might be poor for certain query configurations q_{start} and q_{goal}. A complementary theorem proven in [10] states that a PRM planner will succeed with arbitrarily low probability for any fixed n in spaces with small α and β.

References

1. Alami, R., Laumond, J.-P., Siméon, T.: Two manipulation planning algorithms. In: Goldberg, K., Halperin, D., Latombe, J.-C., Wilson, R. (eds.) Alg. Found. Rob., pp. 109–125. A K Peters, Wellesley (1995)
2. Bretl, T., Lall, S., Latombe, J.-C., Rock, S.: Multi-step motion planning for free-climbing robots. In: WAFR, Zeist, Netherlands (2004)
3. Chaudhuri, S., Koltun, V.: Smoothed analysis of probabilistic roadmaps. In: Fourth SIAM Conf. of Analytic Algorithms and Comp. Geometry (2007)
4. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge (2005)
5. Cortés, J., Siméon, T.: Sampling-based motion planning under kinematic loop-closure constraints. In: WAFR, Zeist, Netherlands (2004)

6. Hauser, K., Bretl, T., Latombe, J.-C.: Learning-assisted multi-step planning. In: IEEE Int. Conf. Rob. Aut., Barcelona, Spain (2005)
7. Hauser, K., Bretl, T., Latombe, J.-C.: Non-gaited humanoid locomotion planning. In: IEEE Humanoids, Tsukuba, Japan (2005)
8. Hauser, K., Bretl, T., Latombe, J.-C., Wilcox, B.: Motion planning for a six-legged lunar robot. In: WAFR, New York (2006)
9. Hsu, D., Kavraki, L.E., Latombe, J.-C., Motwani, R., Sorkin, S.: On finding narrow passages with probabilistic roadmap planners. In: WAFR, Natick, MA, pp. 141–153 (1998)
10. Hsu, D., Latombe, J., Kurniawati, H.: On the probabilistic foundations of probabilistic roadmap planning. Int. J. Rob. Res. 25(7), 627–643 (2006)
11. Hsu, D., Latombe, J.-C., Motwani, R.: Path planning in expansive configuration spaces. In: IEEE Int. Conf. Rob. Aut., pp. 2219–2226 (1997)
12. Koga, Y., Latombe, J.-C.: On multi-arm manipulation planning. In: IEEE Int. Conf. Rob. Aut., San Diego, CA, pp. 945–952 (1994)
13. LaValle, S.M., Yakey, J.H., Kavraki, L.E.: A probabilistic roadmap approach for systems with closed kinematic chains. In: IEEE Int. Conf. Rob. Aut., Detroit, MI (1999)
14. Nielsen, C.L., Kavraki, L.E.: A two level fuzzy prm for manipulation planning. In: IEEE/RSJ Int. Conf. Int. Rob. Sys., pp. 1716–1721 (2000)
15. Nieuwenhuisen, D., van der Stappen, A.F., Overmars, M.H.: An effective framework for path planning amidst movable obstacles. In: WAFR, New York (2006)
16. Sahbani, A., Cortés, J., Siméon, T.: A probabilistic algorithm for manipulation planning under continuous grasps and placements. In: IEEE/RSJ Int. Conf. Int. Rob. Sys., Lausanne, Switzerland, pp. 1560–1565 (2002)
17. Sánchez, G., Latombe, J.-C.: On delaying collision checking in PRM planning: Application to multi-robot coordination. Int. J. of Rob. Res. 21(1), 5–26 (2002)

Toward SLAM on Graphs

Avik De, Jusuk Lee, Nicholas Keller, and Noah J. Cowan

Abstract. We present an algorithm for SLAM on planar graphs. We assume that a robot moves from node to node on the graph using odometry to measure the distance between consecutive landmark observations. At each node, the robot follows a branch chosen at random, without reporting which branch it follows. A low-level process detects (with some uncertainty) the presence of landmarks, such as corners, branches, and bumps, but only triggers a binary flag for landmark detection (i.e., the robot is oblivious to the details or "appearance" of the landmark). Under uncertainties of the robot's odometry, landmark detection, and the current landmark position of the robot, we present an E-M-based SLAM algorithm for two cases: (1) known, arbitrary topology with unknown edge lengths and (2) unknown topology, but restricted to "elementary" 1- and 2-cycle graphs. In the latter case, the algorithm (flexibly and reversibly) closes loops and allows for dynamic environments (adding and deleting nodes).

1 Introduction

Navigation of inexpensive mobile robots with limited computational capabilities, imprecise sensing, and crude odometry presents a number of interesting challenges. Here, we approach the problem of Simultaneous Localization and Mapping (SLAM, cf. [10]) in this setting. We assume that as a robot moves through the environment, a low level control algorithm allows the robot to follow physical structures (walls, doorways, etc.). These physical structures are presumed to give rise to a natural graph structure where nodes of the graph are intermittent features detected by the robot's sensory system, including doorways, corners, or bumps on the wall (Fig. 1). This sensory system was inspired by an artificial antenna [17] from which the tactile feedback received is close range, intermittent, and sparse. This means that the robot needs only run its mapping and localization algorithm occasionally (when a feature

Avik De, Jusuk Lee, Nicholas Keller, and Noah J. Cowan
Department of Mechanical Engineering, Johns Hopkins University
e-mail: {avik.de,jsl,ncowan}@jhu.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 631–645.
springerlink.com

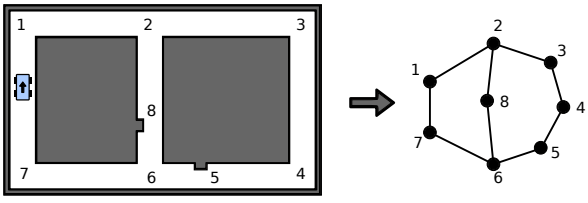


Fig. 1. An illustration of how an environment with landmarks is treated purely as a graph by the robot.

is detected), but that only "corridor-like" environments with walls are considered in this paper. Here, the observation is simply the odometrically measured distance traversed since the last detected node, and we allow for the possibility of missing nodes or detecting false positives.

In short, we propose a solution to the SLAM problem given sparse sensory data (binary and intermittent) and a low dimensional state space. The topological approach lets us abstract the application (for example an indoor environment) from the basics of the algorithm.

Almost all existing SLAM approaches use some statistical technique due to the inherent uncertainty in noisy robot motion and/or observation. Csorba [7] developed the theory behind a modified Extended-Kalman-Filter (EKF) based SLAM algorithm; the EKF method has been improved and used extensively such as in [12]. By the nature of an EKF, the motion model and observation noise are independent: the sensory noise is a function of the sensor physics, and is independent of the robot's motion noise. By contrast, our approach considers the motion and observation models as deeply related: the observation model is the "time to collision" for the motion model. In this paper, we use a Wiener process motion model which gives rise to an Inverse Gaussian sensory model; however this method can work with a variety of movement and sensory models as long as there is a probability density function describing the observations and an estimator, such as maximum likelihood, for its parameters.

Monte Carlo or particle filter approaches like FastSLAM [19] were designed to be computationally efficient, mapping up to thousands of landmarks while using the EKF for landmark location estimation. In our framework of intermittent observations, such a huge number of landmarks would be rare.

We used an E-M (Expectation–Maximization) based mapping approach which has been explored previously by others using various approaches. Unlike Thrun [23] who considered a discrete brute-force minimization of a cost function over a grid-based map, we considered our map lengths to be continuous and derive a formula for an approximate solution. GraphSLAM [24] optimizes a specially constructed graph with robot poses and landmarks as nodes to get the map posterior and is meant to work "off-line." We considered an "on-line" approach where the robot dynamically builds the map as it receives observations. Another approach uses a Kalman filter [22] which attempts to keep track of the full states *in between* nodes

(or observations) by taking the limit of a "dummy" observation variance to infinity. Utilizing the fact that our sensory information (edge length) is only available on every node contact and is directly related to the motor noise, we posed the problem to recovering the topological state of the robot; this has the added benefit of being more robust to problems such as loop closure.

Most SLAM implementations represent the map as a metric object but several researchers have taken a topological approach. Choset and Nagatani [6] treat the higher dimensional robot navigation space as a topology by using a Generalized Voronoi Graph (GVG) and perform localization using graph matching. Our simple sensor models generate sparse data that lends itself well to graph representation, and we attempt to simultaneously map and localize the robot on the graph using only odometry and landmark detection without appearance information. In [20], the authors demonstrate mapping using Bayesian methods and a prior over graphs. To search over the space of graphs, they use Monte Carlo sampling by starting with a random topology, proposing a modification based on a proposal distribution and then picking the new one if it improves a pre-set cost function. Bailey [3] proposes a graph theoretic approach to data association. A recent approach [11] performs SLAM on graphs using "energy" of the graph as a metric for choosing the best fitting topology and Extended Information Filter (EIF) for mapping. For all of these "model selection" issues, like data association and evaluating how good a topological fit is, we use an information theoretic criterion which rationalizes selection based on entropy considerations. A few methods [16, 18] combine both metric and topological information, composed of local feature-based metric maps connected by edges in a topology, using Kalman Filter or FastSLAM based methods. Our approach does not require metric mapping because we estimate lengths in the map as edge parameters.

The main contribution made by this paper is a multi-part algorithm that solves SLAM on planar graphs (assuming "elementary" 1- or 2-cycle topologies) including a novel loop closure approach using a model selection criterion (Sections 3–5). We verify our results and test applications of the algorithm through numerical experiments (Section 6), and address unsolved problems and opportunity for improvement (Section 7).

2 Preliminaries

2.1 Notation

We represent the topology of landmarks as a graph G with N nodes and M edges. We denote the set of all nodes as $\mathscr{X} = \{1, \ldots, N\}$. Each node $i \in \mathscr{X}$ has degree κ_i [13] and a landmark detection probability q_i which is assumed to be known *a priori* (our long-term goal is to use sensor characteristics to estimate this—see Discussion). The edge lengths between adjacent nodes are denoted by $\theta = \{\theta_1, \ldots, \theta_M\}$, and the robot's estimates are $\hat{\theta} = \{\hat{\theta}_1, \ldots, \hat{\theta}_M\}$.

As mentioned in Section 1, our SLAM algorithm runs at discrete instances $k \in \mathbb{Z}$ where each increment in k occurs when a landmark is detected. We use $x_k \in \mathscr{X}$ to denote the *node* position of the robot in the graph at instance k. The robot's (odometry) observation y_k is a variable representing the distance traveled between the events of detecting nodes at instances $k-1$ and k. The history of odometry observations is denoted $y_1^k \equiv \{y_1, \ldots, y_k\}$.

Let $\mathscr{S}$ denote the discrete set of states the robot can be in where $s_k \equiv (x_k, e_k) \in \mathscr{S}$ and e_k is the "entry" edge to node x_k. Note that $|\mathscr{S}| = \sum_{i=1}^{N} \kappa_i$. This choice of state takes into account the position and orientation of the robot in G at any instance k. We also denote the transition made by the robot at time k as t_k, the corresponding observation as y_k, the start state as $\mathrm{L}(t_k)$, and the end state as $\mathrm{R}(t_k)$. Relating to our existing notation, $\mathrm{R}(t_k) = s_k = \mathrm{L}(t_{k+1})$. Let the number of edges included in a transition be the "length" of the path $|t|$.

2.2 Odometry Measurement Error: Inverse Gaussian

When using Bayes' rule or performing parameter estimation we need to analytically express the posterior likelihood of an observation $P_\theta(y)$. This expression can be thought of as the distribution over the first passage time to a fixed distance in a random walk. This distribution is known in the literature as the Inverse Gaussian (IG) or Wald distribution [5].

We assume the robot's motion in between nodes to be a Wiener process with a variance σ^2 and a constant and strictly positive drift velocity v. Then the distribution of the first passage time is a probability density function [5]:

$$\mathscr{N}^{-1}(\tau; \mu, \lambda) = \sqrt{\frac{\lambda}{2\pi\tau^3}} \exp\Big(-\frac{\lambda(\tau-\mu)^2}{2\mu^2\tau}\Big), \tag{1}$$

where $\mu = L/v$, $\lambda = L^2/\sigma^2$, $\tau = y_k/v$ (passage time) and L is the actual edge length associated with the observation y_k. For our purposes it makes more sense to write the pdf as a function of y_k, v and L:

$$P_\theta(y_k) \sim \mathscr{N}^{-1}(y_k; L, v, \sigma^2) = \frac{L v^{3/2} y_k^{-3/2}}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y_k - L)^2 v}{2\sigma^2 y_k}\right). \tag{2}$$

For this distribution, $P(y_k < 0) = 0$, and as noise decreases the shape looks more and more Gaussian.

3 Mapping with Known, Arbitrary Topology

Mapping an unknown environment along with localization make up the SLAM problem. We treat mapping first, and then localization in Section 5.

The "map" in our case consists of the graph G and its associated edge lengths $\theta(G) = \{\theta_1, \ldots, \theta_M\}$. In this section we assume G is known (and can be a general graph without any restrictions) and we find an estimate $\hat{\theta}(G)$:

$$\hat{\theta}(G) = \arg\max_{\theta} P(y_1^k | \theta; G). \tag{3}$$

In Section 4 we address the problem when G is unknown.

3.1 Perfect Data Association

When each observation can be perfectly associated to an edge length, we can do a simple ML estimate of the parameter. For this section, let θ_i be the edge length associated with the observations y_1^k. Then

$$\hat{\theta}_i = \arg\max_{\theta_i} \left(\ln P(y_1^k | \theta_i) \right).$$

Using (2) and maximizing the likelihood function above gives $\hat{\theta}$ as a function of the sample harmonic mean $\langle y \rangle$:

$$\hat{\theta}_i = \frac{1}{2} \left(\langle y \rangle + \sqrt{\langle y \rangle^2 + 4\sigma^2 \langle y \rangle / v} \right), \text{ where } \frac{k}{\langle y \rangle} = \sum_{i=1}^{k} \frac{1}{y_i}. \tag{4}$$

3.2 Imperfect Data Association: E-M Approach

When data association is not deterministic (for example if the robot does not perfectly detect nodes), the observed data y_1^k are "incomplete" and we cannot directly maximize $P_\theta(y_1^k)$. The E-M algorithm [8] introduces "hidden variables" z_1^k which are chosen such that $P_\theta(y_1^k, z_1^k)$ or the "complete data" is specifiable by some distribution.

The E step computes an expectation (and effectively averages over) the hidden variables z_1^k and lets us maximize a likelihood $P_\theta(y_1^k)$:

$$\begin{aligned} Q(\theta, \theta') &= E_{Z|Y;\theta'} \left[\ln P_\theta(y_1^k, z_1^k) \right] \\ &= \sum_{l=1}^{k} \sum_{z_l} (\ln P_\theta(y_l | z_l) + \ln P_\theta(z_l)) P_{\theta'}(z_l | y_1^k). \end{aligned} \tag{5}$$

The natural choice of each hidden variable is $z_l = t_l$, but this raises the issue that the space of paths t is infinite and the sum seems intractable. This problem can be solved by breaking up the sum by the possible length of the path:

$$\sum_t f(t) = \sum_{s \in \mathscr{S}} \sum_{t:\mathrm{R}(t)=s} f(t) = \sum_{s \in \mathscr{S}} \left(\sum_{\substack{t:\mathrm{R}(t)=s, \\ |t|=1}} f(t) + \sum_{\substack{t:\mathrm{R}(t)=s, \\ |t|=2}} f(t) + \ldots \right).$$

The infinite sum above is suited for a breadth-first search (BFS) which is a tree traversal technique. Any planar graph G can be expanded to an infinite tree if we allow nodes to appear multiple times in this tree, by looking at connectivities in the incidence matrix of G. We can make an approximation and truncate this tree at $|t|_{\max}$ levels deep, making the set of possible paths (denoted $\mathscr{T}$) finite and the sum above computable.

By keeping track of detection probabilities q_i of each node in the tree and the sum of the edge lengths from the root node, the BFS can tell us $P(t)$ and $P(y_l|t)$. Since we only care about transitions with non-zero probability, we let $|\mathscr{T}|$ be the number of paths with $P(t) > 0$.

E-Step. We use the "forward" and "backward" algorithms from Hidden Markov Model (HMM) theory to compute $P_{\theta'}(t_l|y_1^k)$ efficiently. Let

$$c_j^{(l)} \triangleq P_{\theta'}(t_l|y_1^k) = \frac{\alpha_{l-1}(\mathrm{L}(t_l))P(t_l)P(y_l|t_l)\beta_{l+1}(\mathrm{R}(t_l))}{\sum_{s'}\alpha_k(s')}, \tag{6}$$

where j indexes into the (finite) set of paths returned by BFS, and

$$\alpha_l(s) = \sum_{t:\mathrm{R}(t)=s} P(y_l|t)P(t)\alpha_{l-1}(\mathrm{L}(t)) \tag{7}$$

$$\beta_l(s) = \sum_{t:\mathrm{L}(t)=s} P(y_l|t)P(t)\beta_{l+1}(\mathrm{R}(t)), \tag{8}$$

with $\alpha_0(s) = P(S_0 = s)$ and $\beta_{k+1}(s) = 1\ \forall\ s \in \mathscr{S}$.[1]

M-Step. We can define a $M \times |\mathscr{T}|$ matrix $\mathbf{D}$ with $d_{ij} = \begin{cases} 1 & \text{if path } j \text{ contained edge } i, \\ 0 & \text{otherwise.} \end{cases}$

Also let L_j be the length of path t_j. To maximize we take the first derivative, and that gives us

$$\begin{aligned}\frac{\partial Q}{\partial \theta_i} &= \sum_{l=1}^{k}\sum_{j:d_{ij}=1} c_j^{(l)}\frac{\partial}{\partial \theta_i}\ln P_\theta(y_l|t_j) = \sum_{l=1}^{k}\sum_{j:d_{ij}=1} c_j^{(l)}\frac{\partial}{\partial L_j}\ln P_\theta(y_l|t_j) \\ &= \sum_{l=1}^{k}\sum_{j:d_{ij}=1} c_j^{(l)}\left(\frac{1}{L_j} + \frac{v}{\sigma^2} - L_j\frac{v}{\sigma^2 y_l}\right),\end{aligned} \tag{9}$$

by the chain rule. For $\frac{\partial L_j}{\partial \theta_i}$, we know $L_j = \sum_{i:d_{ij}=1}\theta_i$ where the condition $d_{ij} = 1$ ensures that the derivative is not zero, and we assume that each L_j passes through θ_i no more than once.

Using all the θ_i we get M quadratic equations in M variables which are hard to solve analytically. Gradient ascent methods may fail because the likelihood function (5) may have local maxima as shown by Fig. 4.

[1] These computations are standard in HMM literature and derivations should be easily found in texts such as [14].

We can get an analytical solution by making the following approximation. If we assume that the edge lengths are large compared to the motion model noise, the first term in the sum in (9) can be ignored.[2] Then we get

$$\frac{\partial Q}{\partial \theta_i} = \frac{v}{\sigma^2} \sum_{l=1}^{k} \sum_{j:d_{ij}=1} c_j^{(l)} \left(1 - \frac{L_j}{y_l}\right). \tag{10}$$

After manipulating the sums, we get a linear system of equations; if we let $U_j = \sum_{l=1}^{k} c_j^{(l)}$, $V_j = \sum_{l=1}^{k} \left(c_j^{(l)}/y_l\right)$, $\mathbf{A} = \{a_{ij}\}, a_{ij} = \sum_{m:d_{im}d_{jm}=1} V_m$, $\mathbf{b} = (b_1, \ldots, b_M)^T$, $b_i = \sum_{m:d_{im}=1} U_m$, and $\theta = (\theta_1, \ldots, \theta_M)^T$, we can solve for the estimates of θ by solving $\mathbf{A}\theta = \mathbf{b}$.

Given reasonable odometry, such as would be expected with a wheeled robot on an office floor, we expect this approximate solution to be fairly close to the real solution, and, when using gradient ascent, within the region of convergence of the true solution. In future work we will establish this for our particular experimental platform.

4 Mapping with Unknown Topology (1- or 2-Cycles)

The space of planar graphs is large, so we restrict our attention to a specific, narrow class of topologies, and enumerate all possible topologies from that class—"elementary" planar graphs consisting *only* of one or two cycles, without leaves or self-loops. The former is clearly defined as a "cycle graph" in literature, and the latter is a union of two cycle graphs (at an edge chain or single vertex) which is connected.

4.1 Model Selection Using Information Theory

To select the most parsimonious model $\hat{G}$ to fit the data y_1^k we use the Akaike Information Criterion (AIC) [1] which is defined as

$$\mathrm{AIC} = 2\mathscr{K} - 2\ln(\mathscr{L}), \tag{11}$$

where $\mathscr{K}$ is the number of model parameters (it is M in our case because there are M edges in G) and $\mathscr{L}$ is the maximized likelihood (5) for that G. A lower AIC indicates a better model. Now the algorithm picks $\hat{G}$ as

$$\hat{G} = \arg\min_{G} \left(2M(G) - 2\ln P(y_1^k | G, \hat{\theta}(G))\right), \tag{12}$$

where $\hat{\theta}(G)$ was found in the previous section.

[2] To achieve (10), we assume $\frac{\sigma^2}{vL_j} \ll 1$. In our numerical trials (Section 6.1) this ratio is in the order of 10^{-3}. The ratio depends on the chosen robot's dynamics and may be determined by performing random trials or characterizing the dynamics [4].

4.2 1-Cycle Graphs

For simple graphs with 1 cycle we are just estimating M, and that completely specifies the graph. This is equivalent to the problem of "closing the loop," because the robot must make a decision about when it has re-visited the start node.

Our mapping procedure finds length estimates according to (3) for a preset range of M (for implementation feasibility), and then uses (12) to find $\hat{G}$.

4.3 2-Cycle Graphs

Graphs with two cycles and other more complex graphs contain *branches*, or nodes with $\kappa_i > 2$. We use this nomenclature because if we call one edge the "entry edge," there are more than 1 "exit edges," only one of which the robot will use to exit the node.

In our simple framework, we will ignore all control input and assume that the robot picks from the exit edges uniformly at random. This can easily be incorporated into the BFS mentioned before in the calculation of $P(t)$.

4.3.1 Enumerating 2-Cycle Topologies

One type of 2-cycle topology has two nodes of degree 3 which are connected by 3 edge chains. We will refer to these edge chains as *superedges*.

For a given total number of edges M, the problem reduces to finding the number of edges in each superedge, or the number of ways in which M can be written as a sum of 3 positive integers ($p+1$ positive integers for a p-cycle graph) which is the number of solutions to $\gamma_1 + \gamma_2 + \ldots + \gamma_{p+1} = M, \gamma_i > 0$, which is the same as the number of solutions to $\phi_1 + \phi_2 + \ldots + \phi_{p+1} = M - (p+1), \phi_i = \gamma_i - 1, \phi_i \geq 0$.

This last problem is almost the same as that of finding "partitions of an integer" which has been studied extensively such as in [2]. In the absence of a simple formula for the number of partitions, we can provide a very conservative upper bound using the "stars and bars" combinatorial argument which will include solutions that are permutations of one another. The upper bound is $\binom{M-1}{p}$ solutions. To actually find these partitions a very simple recursive function that enforces $\phi_i \leq \phi_{i-1}$ or similar to eliminate permuted solutions can be implemented as a computer program.

The other type of 2-cycle graph can be thought of being two distinct 1-cycle graphs joined together at a degree 4 node. These can be enumerated by finding the ways in which M can be expressed as a sum of 2 integers each greater than 1. We disallow a superedge of length 1 because that would imply presence of a self-loop.

4.3.2 The Start State Problem

The 1-cycle graphs has a symmetry that any choice of starting position $P(x_0 = x) = \delta(x, x^\star)$, $x^\star \in \mathscr{X}$ would yield a correct map up to a cycling of the edges. However, for more general graphs, this is not true. One way to circumvent this problem is to

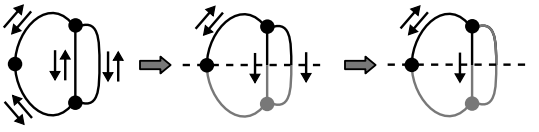


Fig. 2. Eliminating symmetric start states in a 2-cycle graph. (1) Draw a plane of symmetry between the two nodes of degree 3. (2) Discard all starting states lying on edges completely in the lower half. (3) For edges being cut by the symmetry plane, discard one of the two starting states on that edge. (4) If more than one superedge has the same number of edges, ignore duplicates.

use a uniform distribution for $P(s_0)$, which is the most general and intuitive but also less optimal for the algorithm because at the start localization results will be poor.

An alternative approach which we take is to enumerate all the possible distinct starting states for a given topology while taking symmetries into account. The number of elements in this set is usually much smaller than $|\mathscr{S}|$; a visual demonstration of how this elimination occurs is shown in Fig. 2. A start state s_0 can be represented completely by an edge and a direction (with the edge being e_0 and the direction being towards x_0) and is represented by an arrow in the figure. Then we perform E-M calculations for each of those starting states and pick the one that gives maximum likelihood. We use a similar method as in (3) but maximize for θ and s_0 together:

$$\hat{\theta}(G) = \arg\max_{\theta, s_0} P(y_1^k | \theta, s_0; G). \tag{13}$$

5 Localization

The localization problem asks to find a distribution over the current state s_k given the history of observations y_1^k. If we use E-M for mapping as described in Section 3.2, localization is performed implicitly in the E-step. After the computation of (6),

$$P(s_k | y_1^k) = \sum_{t:R(t)=s_k} P_{\theta'}(t | y_1^k). \tag{14}$$

6 Numerical Tests

6.1 *Mapping Lengths with E-M*

Figure 3 shows a simple trial run of the algorithm in Section 3.2 in which the robot attempts to map the edge lengths and localize in a known topology with imperfect association of observations with edges. We used a uniform distribution for $P(s_0)$, and the initial length guesses are shown in the leftmost column of the figure. We chose $|t|_{\max} = 5$ for this trial (and others in this section), and for this choice $\sum_j c_j^{(l)}$ was 1 without need for renormalization.

At every "branch point" the exit edge is picked uniformly at random and node detection is imperfect, resulting in imperfect data association. The given map has a

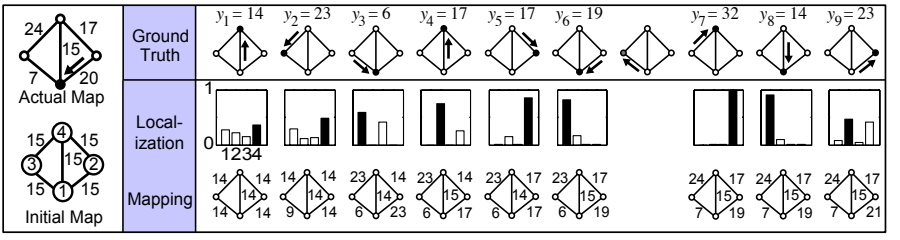


Fig. 3. The first few iterations from a numerical trial of the E-M mapping algorithm for imperfect detection. The filled circles in the "ground truth" row are the true position of the robot and the filled bars in the localization bar plots are the peaks of the pmf $P(x_k)$. Between observations 6 and 7, the robot failed to detect a node.

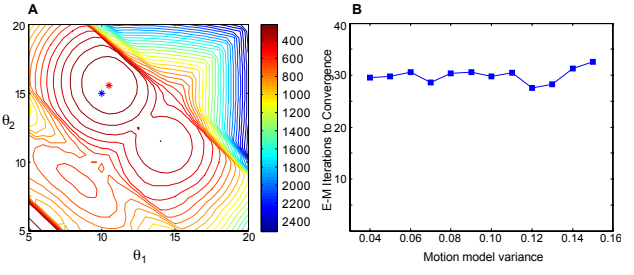


Fig. 4. A: Contours of the likelihood function (5) showing multiple maxima. The blue star is the true value of the parameters and the red star is the solution found using the "noise approximation" in (10). The red star is close to the peak of the global maximum showing that the approximation was valid. B: When the E-M algorithm finds the correct solution, the number of E-M iterations required in total is roughly independent of the motion noise.

symmetry so that if it is, for example, "flipped" about the vertical or horizontal, we still get the same graph. The ground truth and developed maps have been aligned for the reader's convenience.

It is evident that localization results are poor initially because of the developing map and the initial uniform distribution, but it performs better after a few iterations. After each iteration localization is performed using the current map estimate, so the localization results may be drastically different from the previous belief after the map is updated. We can only hope to get good localization results after the developed map is perfect.

Figure 4A justifies the assumption made in (10) by showing that for sufficiently small noise, the peak of the approximated likelihood function (red star) closely matches that of the exact function (blue star) given in (9). As noise is increased, the red star diverges from the blue star and at each E-M iteration, the maximum obtained by solving (10) will not necessarily maximize the true likelihood function. Since the maximization is key to convergence of E-M, we suspect that sufficiently large noise will cause E-M to fail and degrade the performance of our algorithm.

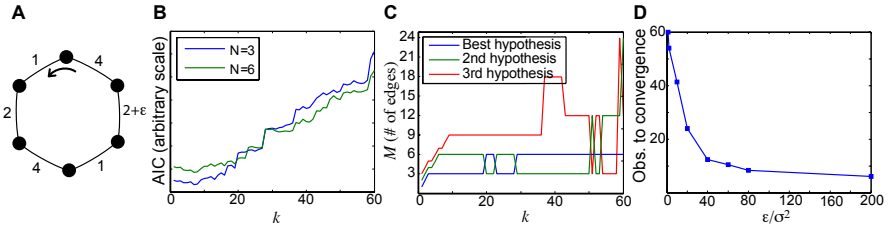


Fig. 5. An illustration of loop re-opening using the AIC model selection criterion. A: The simple (but almost ambiguous) map given to the robot. B: The AIC of the hypotheses for a 3-edge and 6-edge cycle evolving over time. C: The top three hypotheses at any period during the same trial. D: The average convergence time to the correct hypothesis as a function of ε/σ^2.

The figure also shows the presence of local extrema in the likelihood function making a gradient ascent method difficult. This trial was for a simple 2-edge cycle so that the likelihood function can be visualized.

In general for E-M, no bounds can be given on the rate of convergence, but Fig. 4B empirically shows that the total number of E-M iterations required to find the solution is roughly independent of the motion noise, which is important from an implementation point of view.

6.2 *Loop Closure (and Re-opening)*

With respect to "closing the loop," Fig. 5 demonstrates the advantage of using the method described in Section 4.1: the robot can modify a loop closure decision as long as we do not discard the history of observations. In this trial the robot is performing SLAM in an unknown topology with imperfect association.

The robot is given a simple cycle map of six edges with the first three edges almost of equal length to the last three (with one of the three being perturbed by an ε). It initially picks $M = 3$ as the best hypothesis after the trial starts, but after going around more times and receiving more observations it corrects its hypothesis and chooses $M = 6$ as the most parsimonious model. This behavior can be explained at a higher level by the following argument.

When the robot does not have much information about the map, the "perturbation" can be attributed to noise, so that the lower order model is sufficiently good at predicting the observations. From the asymptotic normality of ML estimators [9], we know that $\text{var}(\hat{\theta})$ falls as $1/\sqrt{k}$. So as k gets bigger the lower order model gives a much poorer fit to the data than the higher order model.

The time required for the algorithm to decide to favor the higher order (but correct) model over the lower order model depends on the perturbation, and this "convergence time" is plotted in part (D). The minimum number of observations to support a $M = 6$ model is six, and so there is a horizontal asymptote at 6 as ε gets larger and larger. The motion model variance was $\sigma^2 = 0.05$ for these trials.

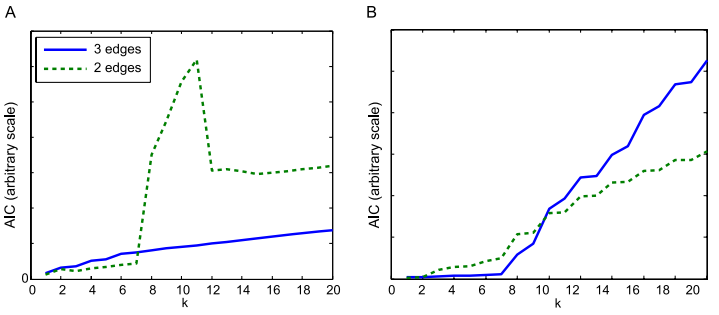


Fig. 6. The A: "new landmark" problem where a node is added to a 2-edge cycle map and B: "disappearing landmark" problem where a node is deleted from a 3-edge cycle, as handled by our algorithm. In both the plots, the change in the environment occurs at iteration 7. For the disappearing landmarks problem, one possibility is that the robot attributes the missing landmark to imperfect detection and continues to favor the higher-order model (which helps in case the landmark *re*-appears). In this trial we set detection probabilities $q_i \approx 1$ so that this did not happen.

6.3 Dynamic Environment

Another problem which pertains to model selection for us is the problem of "disappearing landmarks" where a previously existing landmark is taken away, and the similar problem of "new landmarks" where a newer landmark is inserted in the environment. By comparing the AIC of hypotheses which have M close to the best model the robot can make "soft" or modifiable decisions about the nature of its environment (Fig. 6).

Note in Fig. 6A that the 2-edge model adjusts its length estimates to considerably lower its AIC around iteration 12, but it still cannot match the 3-edge model. In Fig. 6B it takes about three iterations for the 2-edge hypothesis to beat the 3-edge hypothesis.

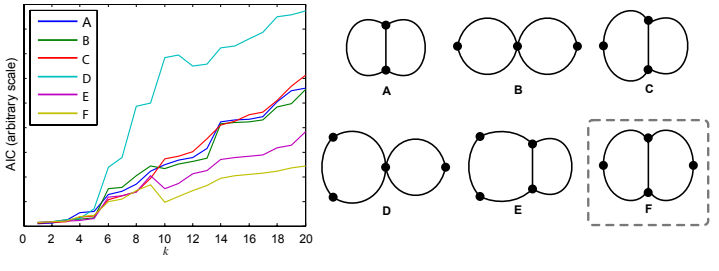


Fig. 7. The robot picks the best (2-cycle) topology fit for some observations using AIC. The true topology was "F," i.e. the rightmost one in the second row.

6.4 Topology Enumeration and Selection: 2-Cycle Graphs

We used the AIC again to help the robot pick the best topology according to (12). Due to calculation costs, the robot searched only the space of 2-edge-connected planar graphs with two cycles, having $3 \leq M \leq 5$. The candidate topologies are shown in the right half of Fig. 7. For each of these "edge-partition" hypotheses, there were a number of possible starting states to be taken into account (see Section 4.3.2). The robot assumed perfect detection for this particular trial.

7 Discussion

This paper presents an algorithm to perform SLAM on elementary graphs. We divide the mapping into two parts, known and unknown topology. Our algorithm solves the former in the most general case using E-M, and presents techniques of finding the topology from a very small subset of planar graphs. We pick this subset to be graphs with one or two cycles, but this could be easily extended to other families of graphs as long as they can be parametrized and enumerated.

This method, while still targeting simple environments and using minimal sensory information, can address some well known SLAM problems [10]. We present numerical experiments demonstrating how the algorithm solves the "loop closure" problem without requiring appearance information in a non-parametric way by using an information theoretic model selection criterion. We also present numerical experiments illustrating the solution to the problem of dynamic environments by maintaining multiple hypotheses.

To apply our framework to the real world, several issues would have to be addressed. First, we need a better way to characterize various landmarks and their corresponding detection probabilities. Although the antenna sensor is capable of capturing finer details of a landmark, we have chosen to use only sparse sensor input (a binary flag for detection) which gives rise to problems such as "misses" and "false positives" (Section 6.3) which correspond to the sensor signal being below or above (resp.) a pre-set threshold. This approach would take many trials to give an accurate detection probability and the detection probability for one landmark would most likely be different from the rest of the landmarks. In the future, we plan to incorporate a richer sensor model of the antenna to (a) correct "misses" or "false positives" for cases in which the signal is close to the threshold, (b) predict q_i based on how "close" the signal magnitude is to the threshold the first time a landmark is observed, and (c) incorporate landmark appearances in our algorithm which may make data association simpler in more complex environments.

Second, some parts of the algorithm, such as maintaining multiple hypotheses for a large number of possible topologies as well as enumerating to find graph structures, require large computations. Methods of graphical inference from data [15] as well as approximate methods that take assumptions about the structure, such as the "topology improvement algorithm" in [21], should be explored. We plan to compute the complexity bounds of our algorithm and compare the performance with our approach to other existing approaches such as EKF-SLAM in large environments.

A natural extension to the algorithm presented here would be to devise a method of mapping a general planar graph. Our algorithm targets indoor 1-dimensional environments (corridors), but we imagine using our algorithm with GVG's [6] to map higher dimensional environments; extensions to the algorithm to handle "leaves" will naturally have to be made. Furthermore, with the framework in this paper, we can perform SLAM on any parameter (e.g., landmark appearance) which can be associated with edges on a graph that has states as nodes, as long as we can define a probability distribution for observations and an estimator of its parameters.

Acknowledgments

The authors would like to thank the anonymous reviewers for helpful comments on elucidating approximations and the rationale behind our approach, and also Donniell Fishkind for helpful discussions on graph theory. Contributions: AD with input from NC and JL developed the algorithms, performed all numerical experiments, and wrote the manuscript; NK with JL and NC formulated a preliminary grid-based approach (not reported) to demonstrate the feasibility of 1D SLAM.

References

1. Akaike, H.: A new look at the statistical model identification. IEEE Trans. Autom. Control 19(6), 716–723 (1974)
2. Andrews, G.E.: The theory of partitions. Addison-Wesley, Reading (1976)
3. Bailey, T.: Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis, Department of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney (2002)
4. Borenstein, J., Feng, L.: Measurement and correction of systematic odometry errors in mobile robots. IEEE Trans. Robot. Autom. 12, 869–880 (1996)
5. Chhikara, R.S., Folks, J.L.: The inverse Gaussian distribution: theory, methodology, and applications. Marcel Dekker, Inc., New York (1989)
6. Choset, H., Nagatani, K.: Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization. IEEE Trans. Robot. Autom. 17, 125–137 (2001)
7. Csorba, M.: Simultaneous Localisation and Map Building. PhD thesis, Department of Engineering Science, University of Oxford (1997)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statistical Society. Series B (Methodological) 39(1), 1–38 (1977)
9. DuMouchel, W.: On the asymptotic normality of the maximum-likelihood estimate when sampling from a stable distribution. Annals of Statistics 1(5), 948–957 (1973)
10. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: Part I. IEEE Robot. Autom. Magazine 13(2), 99–108 (2006)
11. Folkesson, J., Christensen, H.I.: Closing the loop with graphical SLAM. IEEE Trans. Robot. 23(4), 731–741 (2007)
12. Gamini Dissanayake, M.W.M., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Autom. 17(3), 229–241 (2001)

13. Harary, F.: Graph theory. Addison-Wesley, Reading (1969)
14. Jelinek, F.: Statistical methods for speech recognition. MIT Press, Cambridge (1997)
15. Jordan, M.I. (ed.): Learning in Graphical Models. MIT Press, Cambridge MA (1999)
16. Kouzoubov, K., Austin, D.: Hybrid topological/metric approach to SLAM. In: IEEE Intl Conf. Robot. Autom., April 2004, vol. 1, pp. 872–877 (2004)
17. Lee, J., Sponberg, S.N., Loh, O.Y., Lamperski, A.G., Full, R.J., Cowan, N.J.: Templates and anchors for antenna-based wall following in cockroaches and robots. IEEE Trans. Robot. 24(1), 130–143 (2008)
18. Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I., Choset, H.: Hierarchical simultaneous localization and mapping. In: IEEE/RSJ Intl. Conf. Intell. Robots Syst., October 2003, vol. 1, pp. 448–453 (2003)
19. Montemerlo, M.: FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association. PhD thesis, School of Computer Science, Carnegie Mellon University (2003)
20. Ranganathan, A., Menegatti, E., Dellaert, F.: Bayesian inference in the space of topological maps. IEEE Trans. Robot. 22(1), 92–107 (2006)
21. Shi, L., Capponi, A., Johansson, K.H., Murray, R.M.: Network lifetime maximization via sensor trees construction and scheduling. In: Third International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks, Annapolis, MD, USA (June 2008)
22. Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M.I., Sastry, S.S.: Kalman filtering with intermittent observations. IEEE Trans. Autom. Control 49, 1453–1464 (2004)
23. Thrun, S.: Learning occupancy grid maps with forward sensor models. Autonomous Robots 15(2), 111–127 (2003)
24. Thrun, S., Montemerlo, M.: The graph SLAM algorithm with applications to large-scale mapping of urban structures. Intl. J. Robot. Res. 25(5-6), 403–429 (2006)

HybridSLAM: Combining FastSLAM and EKF-SLAM for Reliable Mapping

Alex Brooks and Tim Bailey

Abstract. This paper presents HybridSLAM: an approach to SLAM which combines the strengths and avoids the weaknesses of two popular mapping strategies: FastSLAM and EKF-SLAM. FastSLAM is used as a front-end, producing local maps which are periodically fused into an EKF-SLAM back-end. The use of FastSLAM locally avoids linearisation of the vehicle model and provides a high level of robustness to clutter and ambiguous data association. The use of EKF-SLAM globally allows uncertainty to be remembered over long vehicle trajectories, avoiding FastSLAM's tendency to become over-confident. Extensive trials in randomly-generated simulated environments show that HybridSLAM significantly out-performs either pure approach. The advantages of HybridSLAM are most pronounced in cluttered environments where either pure approach encounters serious difficulty. In addition, the HybridSLAM algorithm is experimentally validated in a real urban environment.

1 Introduction

The Simultaneous Localisation and Mapping (SLAM) problem has attracted immense attention in the mobile robotics literature. The problem involves building a map while computing a vehicle's trajectory through that map, based on noisy measurements of the environment and the vehicle's odometry.

This paper specifically addresses the problem of feature-based SLAM, where the environment is modelled as a discrete set of features, each described by a number of continuous state variables. The standard solution is to take a Bayesian approach, explicitly modelling the joint probability distribution over possible vehicle trajectories and maps. There are currently two popular approaches to modelling this distribution. The first is to linearise and represent the joint probability with a single

Alex Brooks and Tim Bailey
Australian Centre for Field Robotics, University of Sydney
e-mail: {a.brooks,t.bailey}@acfr.usyd.edu.au

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 647–661.
springerlink.com

high-dimensional Gaussian. This is the approach taken by EKF-SLAM [8] and its variants [2]. The second approach is to use a Rao-Blackwellised particle filter, representing the vehicle's trajectory using a set of particles and conditioning the map on the vehicle's trajectory. Examples of this approach include FastSLAM [16] and others (*e.g.* [9]).

Both approaches have their strengths and weaknesses, and it is not difficult to produce examples where one or the other will fail. In particular, the EKF is prone to failure where significant vehicle uncertainty induces linearisation errors [3], or where significant clutter induces ambiguity in data association. The latter issue is problematic because the standard EKF formulation requires that hard data association decisions be made, by selecting the most likely hypothesis. Once a bad association is made it cannot be un-done and can cause the filter to fail catastrophically [2].

The probability of making a potentially-catastrophic incorrect decision can be reduced by using batch association methods. Non-batch methods such as nearest-neighbour association consider each observation in isolation, under the assumption that landmarks are independent. Joint compatability (*e.g.* JCBB [11]) improves robustness by simultaneously considering all observations made from a single pose, but cannot consider relations between observations made from different poses. Sliding-window methods simultaneously consider all observations made from poses in a small window of recent history, but can be computationally intractable without the assumption of landmark independence for multiple observations from a single pose [12][6].

In contrast to the EKF, FastSLAM does not suffer from linearisation problems (because it does not linearise the vehicle pose), and is much more robust in situations of association ambiguity. For data association, each particle is allowed to make independent decisions, hence FastSLAM can maintain a probability distribution over all possible associations [16]. As more observations arrive, particles which made poor association decisions in the past tend to be removed in the resampling process, hence the majority of particles tend to converge to the correct set of associations. For the purposes of data association, FastSLAM automatically allows information to be integrated between observations at a *single* time step (as JCBB does), and between *multiple* time steps (as sliding-window methods do). The former occurs due to landmarks being conditionally independent given the vehicle path, the latter due to each particle's memory of past associations. Furthermore, FastSLAM is simple to implement relative to complicated batch association algorithms.

The disadvantage of FastSLAM is its inability to maintain particle diversity over long periods of time [4]. The fundamental problem is that the particle filter operates in a very high-dimensional space: the space of vehicle *trajectories* (not momentary poses). The number of particles needed is therefore exponential in the length of the trajectory. When a smaller number is used, the filter underestimates the total uncertainty, and eventually becomes inconsistent. While this may still produce good maps, problems are encountered when the full uncertainty is required, for example when large loops need to be closed.

For remembering long-term uncertainty, the EKF is far superior. By using a continuous (Gaussian) representation, uncertainty does not degrade purely as a function

of trajectory length. Linearisation errors are still a concern [3], but these are far less severe than FastSLAM's particle diversity problems.

This paper presents a hybrid mapping strategy which combines the strengths of both approaches. We advocate the use of FastSLAM as a front-end to an EKF-SLAM back-end. The FastSLAM front-end is used to build local maps and to disambiguate transient data association uncertainties. Before the trajectory becomes so long that particle diversity becomes problematic, a single Gaussian is computed from the FastSLAM posterior. This Gaussian local map is then fused into the global map. At the point of fusion, a hard decision must be made about the associations between local map features and global map features. However, a large local map can provide sufficient constraints to make the probability of a bad decision extremely low. The result is a SLAM algorithm which is robust to linearisation errors and data association ambiguities on a local scale, and can also close large loops on a global scale.

The remainder of the paper is structured as follows. Section 2 gives a brief review of FastSLAM and shows how a single Gaussian can be produced from a FastSLAM posterior, under the assumption of known associations. This assumption is removed in Section 3, which addresses the problem of producing a Gaussian posterior given unknown associations. Section 4 describes the algorithm for fusing local sub-maps produced by the front-end into the global map. Section 5 describes a set of experiments and discusses the results, and Section 6 concludes.

2 Conversion of Factored FastSLAM Distribution to a Single Gaussian

The aim of a SLAM algorithm is to compute the posterior

$$p(\mathbf{v}_{0:t}, \mathbf{M} | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{v}_0) \tag{1}$$

where $\mathbf{v}_{0:t} = \mathbf{v}_0, \ldots, \mathbf{v}_t$ denotes the path of the vehicle, the map $\mathbf{M} = \theta_1, \ldots, \theta_N$ denotes the positions of a set of landmarks, $\mathbf{Z}_{0:t}$ represents the set of all observations, $\mathbf{U}_{0:t}$ denotes the set of control inputs, and $\mathbf{v}_0$ denotes the initial vehicle state. The subscript $_{0:t}$ indicates a set of variables for all time steps up to and including t, while the subscript $_t$ is used to indicate the variable at time step t. Many SLAM algorithms (including the one described in this paper) marginalise out past poses, estimating only

$$p(\mathbf{v}_t, \mathbf{M} | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{v}_0) \tag{2}$$

Fusion into a global EKF-SLAM back-end requires the local mapping algorithm to produce this distribution in the form of a single multi-dimensional Gaussian. This section shows how a distribution of this form can be extracted from FastSLAM, under the restrictive assumption that the environment contains N uniquely-identifiable landmarks, all of which have been observed at least once. This assumption will be removed in Section 3.

2.1 The FastSLAM Posterior as a Gaussian Mixture Model (GMM)

The FastSLAM algorithm factors distribution 1 as follows [16]:

$$p(\mathbf{v}_{0:t}, \mathbf{M} | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{v}_0) = p(\mathbf{v}_{0:t} | \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}, \mathbf{v}_0) \prod_n p(\theta_n | \mathbf{v}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{U}_{0:t}) \tag{3}$$

where the first factor represents the vehicle's path and subsequent factors represent landmark positions given the vehicle's path. This factored distribution is represented as a set of P samples, with the p^{th} particle S_t^p consisting of a weight w_t^p, a vehicle pose $\mathbf{v}_t^p$ (past poses are marginalised out), and N Gaussian landmark estimates described by their mean $\mu_{n,t}^p$ and covariance $\Sigma_{n,t}^p$:

$$S_t^p = < w_t^p, \mathbf{v}_t^p, \underbrace{\mu_{1,t}^p, \Sigma_{1,t}^p}_{\text{landmark } \theta_1}, \ldots, \underbrace{\mu_{N,t}^p, \Sigma_{N,t}^p}_{\text{landmark } \theta_N} > \tag{4}$$

While it is convenient to represent individual landmark distributions as isolated low-dimensional Gaussians, each particle can equivalently be represented using

$$S_t^p = < w_t^p, \mathbf{x}_t^p, \mathbf{P}_t^p > \tag{5}$$

where $\mathbf{x}_t^p = [\mathbf{v}_t^p, \mu_{1,t}^p, \ldots, \mu_{N,t}^p]$ denotes the concatenation of the vehicle states with all landmark states, and $\mathbf{P}_t^p$ denotes a block-diagonal covariance matrix constructed from the vehicle covariance and the covariances of each landmark:

$$\mathbf{P}_t^p = \begin{bmatrix} \mathbf{P}_{vv,t}^p & & & \\ & \Sigma_{1,t}^p & & \\ & & \ddots & \\ & & & \Sigma_{N,t}^p \end{bmatrix}$$

In this case the vehicle covariance $\mathbf{P}_{vv,t}^p$ is zero because each particle has no uncertainty associated with the vehicle states.

Using this representation, the FastSLAM distribution over vehicle and map states is a Gaussian Mixture Model (GMM): each particle is a Gaussian component with weight, mean, and covariance given by Equation 5.

2.2 Conversion to a Single Gaussian

The parameters of a single Gaussian with mean $\mathbf{x}_t$ and covariance $\mathbf{P}_t$ can be computed from the GMM using a process known as moment matching [5]:

$$\mathbf{x}_t = \sum_p w_t^p \mathbf{x}_t^p \tag{6}$$

$$\mathbf{P}_t = \sum_p w_t^p \left[\mathbf{P}_t^p + (\mathbf{x}_t^p - \mathbf{x}_t)(\mathbf{x}_t^p - \mathbf{x}_t)^T\right] \tag{7}$$

In Equation 7, the first term in the square brackets is the covariance of the particle's individual map (due to sensor noise), while the second term is due to the variation between particles' maps (due to vehicle noise).

3 Conversion Using Unknown Associations

This section extends to the more realistic case where landmarks are not uniquely identifiable. Since FastSLAM particles are free to make data association decisions independently, both the number of landmarks and their ordering may differ between particles. However, since the particles are to be merged into a single high-dimensional Gaussian distribution (for fusion into the global map), it is necessary to track the correspondences between landmarks in different particles' maps. The aim is to produce a single common set of features from the particles' individual sets of features.

While the true set of landmarks is unknown, suppose each landmark in the environment is assigned a unique index. Each particle can be augmented with a set of correspondence variables of the form $\gamma_{i,t}^p$, indicating the correspondences between landmarks in the maps of individual particles and landmarks in the environment: $\gamma_{i,t}^p = n$ indicates that the i^{th} landmark in the p^{th} particle's map corresponds to the n^{th} landmark in the environment. Each particle is therefore of the form:

$$S_t^p = < w_t^p, \mathbf{v}_t^p, \underbrace{\mu_{1,t}^p, \Sigma_{1,t}^p, \gamma_{1,t}^p}_{\text{landmark } \theta_1^p} \cdots, \underbrace{\mu_{N^p,t}^p, \Sigma_{N^p,t}^p, \gamma_{N^p,t}^p}_{\text{landmark } \theta_{N^p}^p} > \tag{8}$$

Sections 3.1 and 3.2 describe procedures for (a) tracking these correspondence variables within a particle's map, and (b) generating a single Gaussian given a set of particles with non-identical correspondence variables.

3.1 Tracking Correspondences

The correspondences between features are tracked using a simple voting scheme. Consider a single observation. Each particle can vote to either

1. ignore the observation as spurious,
2. associate the observation with a particular existing map feature, or
3. create a new map feature and associate the observation with this.

Each particle votes in proportion to its weight, and the winner takes all: a single correspondence variable is computed based on the majority vote. All particles must use this correspondence variable for features modified or created by this observation, over-writing any previous correspondence variable.

This scheme forces the particles to form a consensus about the common set of features, but allows that consensus to be overturned at a later time if future

information alters the particle weights. In principal it is possible for particles to disagree wildly about the number of features, but in practice a large majority tends to form quickly, out-voting a small set of dissenting particles.

3.2 Producing a Gaussian Given Correspondences

The correspondence variable γ provides a mapping from features in each particle's individual map to the common set of features $\mathbf{M}$. Let the function δ denote the reverse mapping: $\delta_t(n,p) = i$ indicates that the n^{th} feature in the common set is represented by the i^{th} feature in the p^{th} particle's map.

Given these variables, each particle can be represented using a weight, mean, and covariance as in Equation 5 with the mean

$$\mathbf{x}_t^p = [\mathbf{v}_t^p, \mu^p_{\delta_t(1,p),t}, \ldots, \mu^p_{\delta_t(N,p),t}] \tag{9}$$

and the block-diagonal covariance matrix

$$\mathbf{P}_t^p = \begin{bmatrix} \mathbf{P}^p_{vv,t} & & & \\ & \Sigma^p_{\delta_t(1,p),t} & & \\ & & \ddots & \\ & & & \Sigma^p_{\delta_t(N,p),t} \end{bmatrix}$$

In other words, each particle's individual map can be represented by a single multi-dimensional mean and covariance, produced by arranging the individual map feature states in the order of the common features about which they are a hypothesis. A Gaussian distribution can then be produced using Equations 6 and 7.

This approach requires that two corner cases be addressed. Firstly, a particle's individual map may contain multiple features referring to the same common feature. In this case, δ simply selects one at random (another alternative would be to combine the features). Secondly, a common feature may have no corresponding features in a particular particle's map. In this case, the particle is ignored when computing the mean and covariance of that common feature.

In addition to computing the mean and covariance of the landmarks' positions, independent state estimates (such as the probability of existence of each landmark) can be computed using weighted sums analogous to Equation 6. Maintaining an estimate of the probability of existence of each feature means that features which are not generally agreed upon will have a low probability of existence. These features can be ignored when fusing the local map into the global map.

4 Sub-map Fusion

This paper uses a sub-mapping strategy similar to CLSF [17] and others [14]. Figure 1 illustrates the approach: the filter maintains both a Gaussian global map and a Gaussian local map. The (uncertain) pose of the local map's origin is stored in

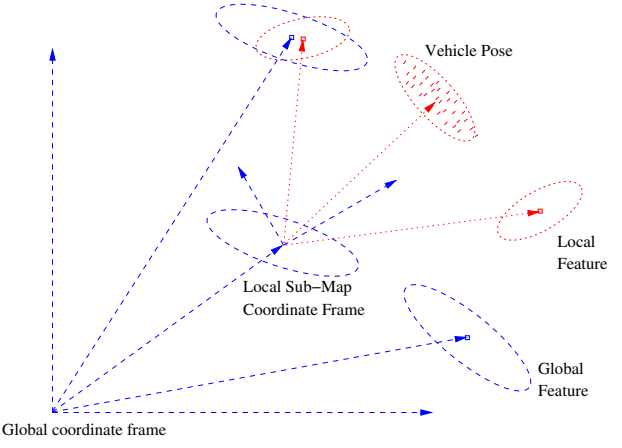


Fig. 1. An illustration of the sub-mapping strategy. In this case, the local map is converted to a Gaussian from a FastSLAM representation, in which the local vehicle pose is represented using particles.

the global map. The filter periodically fuses the local map into the global map to produce a single map in the global coordinate frame. After fusion, a new local map is started with the vehicle beginning at the local origin with no uncertainty and no landmarks.

Map fusion requires two steps:

1. local features are initialised in the global map, and
2. features are associated and fused.

Note that the vehicle pose in the local map is considered to be just another feature, and hence does not require any special treatment in the procedure that follows.

Let $\mathbf{x}^-$, $\mathbf{x}^+$, $\mathbf{P}^-$, and $\mathbf{P}^+$ refer to the global map mean and covariance immediately before and after fusion, and $\mathbf{x}_L$ and $\mathbf{P}_L$ refer to the local map mean and covariance. Local features (including the vehicle pose) are initialised in the global map by augmenting the state and covariances as follows:

$$\mathbf{x}^+ = \begin{bmatrix} \mathbf{x}^- \\ g(\mathbf{v}^-, \mathbf{x}_L) \end{bmatrix}$$

$$\mathbf{P}^+ = \begin{bmatrix} \mathbf{P}^-_{vv} & \mathbf{P}^-_{vm} & \mathbf{P}^{-T}_{vv}\nabla_v g^T \\ \mathbf{P}^{-T}_{vm} & \mathbf{P}^-_{mm} & \mathbf{P}^{-T}_{vm}\nabla_v g^T \\ \nabla_v g\mathbf{P}^-_{vv} & \nabla_v g\mathbf{P}^-_{vm} & \nabla_v g\mathbf{P}^-_{vv}\nabla_v g^T + \nabla_z g\mathbf{P}_L\nabla_z g^T \end{bmatrix}$$

where $g(\mathbf{v}^-, \mathbf{x}_L)$ transforms the local map into the global coordinate frame, using $\mathbf{v}^-$: the vehicle's global pose at the time of the previous fusion (now the origin of the local sub-map).

New features are then associated with old features, using joint compatibility [11]. Two features θ_1 and θ_2 are fused using a zero-noise observation $\mathbf{z}_f$ of the difference between features:

$$\mathbf{z}_f(\theta_1, \theta_2) = \theta_1 - \theta_2 \tag{10}$$

and setting $\mathbf{z}_f = \mathbf{0}$. This equates to observing that the two features are the same and leaves two identical features, one of which is removed. Finally, the old global vehicle pose estimate is replaced by the new estimate, and a new local map is started.

5 Experiments

Experiments examined the ability to operate amid clutter and to close large loops. The following filters were compared:

1. FastSLAM v2.0 [16] (FASTSLAM),
2. The sub-mapping algorithm described in Section 4 with EKF front- and back-ends (EKF), and
3. The same sub-mapping algorithm with a FastSLAM front-end and an EKF back-end (HYBRID).

In all experiments, the vehicle was equipped with a range-bearing sensor subject to false alarms and non-detections. The sensor's reliability was specified by two parameters:

- α: the probability of detecting a true feature, and
- β: the probability a false alarm. Simulation experiments generated false alarms with uniformly-distributed range and bearing, with a maximum number of false alarms equal to the number of visible true features.

In order to be able to ignore spurious features, all three filters maintain an estimate of the probability of existence of each feature. Let $p(\iota_{n,t})$ denote the estimated probability of existence of the n^{th} feature at time t. This estimate can be updated as in a traditional target-detection framework [13], using

$$p(\iota_{n,t}) \leftarrow p(\iota_{n,t}) \frac{\alpha}{p(\iota_{n,t})\alpha + (1 - p(\iota_{n,t}))\beta} \tag{11}$$

where the denominator gives the total probability of making an observation, either through false-alarm or true feature detection.

The two sub-mapping algorithms only fused features whose probability of existence exceeded 0.95, while FASTSLAM ran a periodic cull of old unlikely features. Sub-maps were fused at 10-second intervals. FastSLAM implementations used 500 particles. Note that while FastSLAM is often implemented with fewer particles (*e.g.* 100 [10]), a larger number can be tolerated in our approach because HYBRID's local FastSLAM map is always very small. Both FASTSLAM and the HYBRID front-end used Data Association Sampling (DAS) [16] for computing correspondences.

5.1 Simulation Environment

The experimental environments consisted of 50 randomly-generated simulated two-dimensional worlds of size 120m×120m, each containing 35 uniformly-distributed point features, as shown in Figure 2. The vehicle moved at 1m/s and up to 100°/s, observing features using a 360° range-bearing sensor with a maximum range of 25m. The vehicle followed a path near the extremities of the environment, such that one significant loop closure had to be executed, for a loop size of approximately 400m.

SLAM filters were run at the observation frequency of 10Hz. Vehicle linear and rotational velocity was noisy, perturbed with standard deviations equal to $0.25\dot{x}$ and $0.35\dot{\theta}$ respectively, where $\dot{x}$ and $\dot{\theta}$ are commanded linear and rotational velocities. Observation noise standard deviations were set to $0.05m$ and 1° in range and bearing respectively. Identical worlds, odometry, and observations were used for all filters.

5.2 Simulation Results in Cluttered Environments

A number of trials were conducted with varying settings for the sensor's reliability. Each test run was considered to have failed if the true vehicle pose remained more than 10 standard deviations from the filter's estimate for more than 20 iterations. The results are summarised in Table 1. They clearly show that the HYBRID filter out-performs both FASTSLAM and EKF filters. FASTSLAM's tendency to become overconfident meant that it was unable to succeed in a single run (see the following sections for more detail).

Comparing the two sub-mapping approaches, the benefits of using HYBRID versus EKF are apparent with no clutter but most pronounced in high levels of clutter.

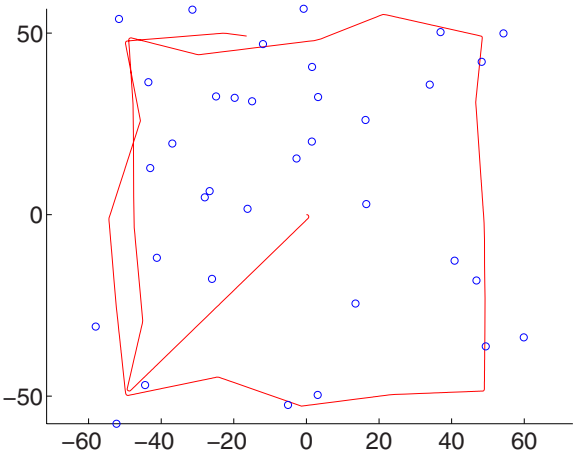


Fig. 2. An example scenario, showing the true vehicle path and landmark positions. The precise path and landmark locations are randomly generated.

Table 1. Success rate for each algorithm, evaluated over 100 trials in each case, for different values of α and β. Failure is defined as the vehicle mean being more than 10 standard deviations from the true vehicle pose for more than 20 iterations.

	α, β [p(truePositive),p(falsePositive)]			
Filter	1.0, 0.00	0.7, 0.02	0.4, 0.04	0.1, 0.06
FASTSLAM	0%	0%	0%	0%
EKF	79%	76%	72%	30%
HYBRID	87%	89%	83%	63%

While the level of clutter in the extreme case is particularly high, it is also unrealistically easy to filter because it is truly random. Clutter in the real world is not independent and identically distributed, but tends to be correlated, clustered around particular (possibly moving) objects.

5.2.1 Discussion

Failures occurred for a number of reasons. The FASTSLAM failures occurred due to over-confidence, resulting from an inability to remember uncertainty over long trajectories [4].

The two sub-mapping algorithms have two common long-term failure modes, specific to the EKF back-end. Firstly, when sufficient vehicle heading uncertainty accumulates, linearisation errors can cause the filter to become over-confident [3]. Secondly, data association errors were sometimes made when closing the loop. The latter issue is particularly problematic because the data fusion algorithm is relatively

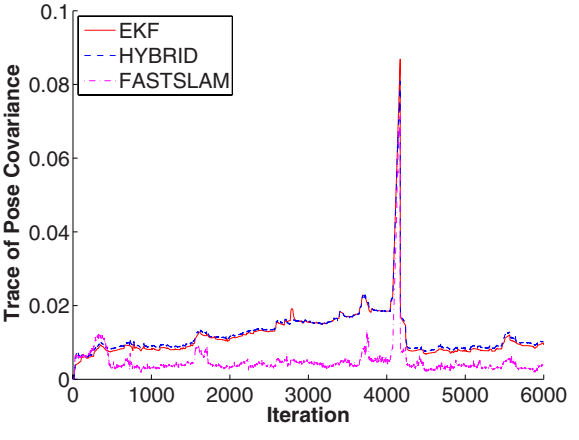


Fig. 3. Vehicle uncertainty over time in a scenario with no vehicle non-linearities. Uncertainty is measured using the trace of the covariance matrix. In order to produce a meaningful comparison with FASTSLAM, the sub-mapping algorithms' estimates are based on both the local and global maps at every iteration, even though the maps are only fused on every 100^{th} iteration.

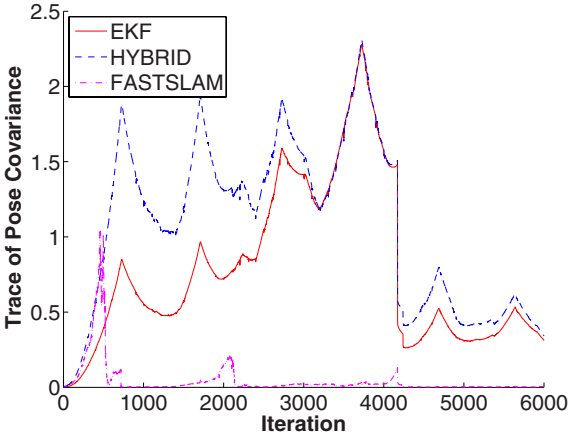


Fig. 4. Vehicle uncertainty over time in a more realistic scenario, with vehicle non-linearities. Sub-map fusion causes small drops in uncertainty every 100 iterations. The spike after iteration 4000 is due to the vehicle passing an area devoid of landmarks before closing the loop.

simplistic, content to close a loop on a single matching feature. The back-end data-association problem could be solved in a variety of ways, for example by running the local mapper for longer when a loop closure is imminent. Both problems were exacerbated as the reliability of the sensor was decreased, since the total amount of sensor information available to the filter was reduced, increasing vehicle uncertainty.

The differences between the two sub-mapping filters occurred due to differences in short-term mapping in the front-end. Failures in the EKF filter occurred due to spurious observations being associated with landmarks, particularly during transient periods of high vehicle uncertainty (*i.e.* during turns). This is especially problematic when the true feature is not observed simultaneously, and therefore JCBB is unable to resolve the ambiguity.

EKF failures also occurred due to local (front-end) linearisation errors. Local linearisation errors are kept to a minimum because vehicle uncertainty in the local frame of reference is generally small, since local maps begin with no uncertainty. However local-map uncertainty can be significant, for example if the vehicle begins a new sub-map immediately prior to making a sharp turn amid sparse landmarks.

The HYBRID filter is much more robust to both these local sources of error. The probability of incorporating spurious observations is reduced by the front-end's ability to integrate data association information over time. Vehicle linearisation errors are eliminated because each particle always has zero vehicle uncertainty.

5.3 Simulation Results in the Absence of Clutter

Two additional experiments were conducted in the same simulated environment, to examine sources of error unrelated to clutter. The effects of clutter were eliminated by setting α to 1.0 and β to 0.0).

Fig. 5. The Segway RMP vehicle and the environment

In the first experiment, the vehicle is provided with an accurate compass, eliminating vehicle linearisation errors. The observation model is still linearised, however this introduces only minor errors relative to vehicle non-linearities. Since the Kalman Filter is the optimal estimator in the linear case, the EKF solution is close to optimal in this case, and therefore provides a benchmark against which to compare alternative algorithms.

In the absence of vehicle non-linearities and clutter, one would expect HYBRID and EKF to produce very similar results. To minimise differences due to the statistical noise introduced by HYBRID's front-end, we use 5000 particles.

Figure 3 compares the vehicle uncertainty for all three filters, measured using the trace of the vehicle pose covariance matrix. The figure shows that HYBRID can correctly track the uncertainty while FASTSLAM quickly becomes over-confident, even with 5000 particles.

In a second experiment, vehicle uncertainty was introduced by removing the compass. Since the vehicle is executing many uncertain turns (including one sharp turn near the beginning of the trajectory), non-linearity due to vehicle heading uncertainty is problematic. Figure 4 shows the vehicle uncertainty over the entire run.

In this case, the uncertainties of both sub-mapping solutions are certainly underestimated, due to vehicle linearisation issues. However, since the HYBRID solution avoids vehicle linearisation in local sub-maps, the larger uncertainties which it reports are closer to the optimal solution. HYBRID is clearly an improvement on EKF in this respect. The swift deterioration of the FASTSLAM estimate is clear in this scenario.

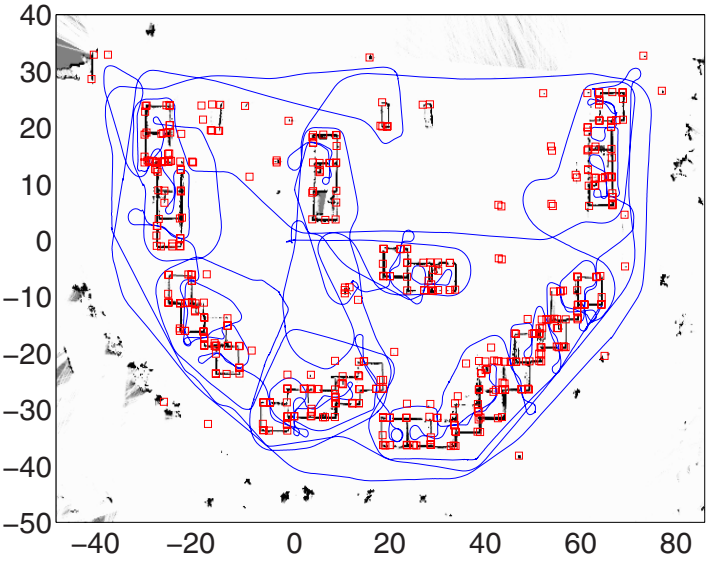


Fig. 6. A map of an urban environment produced by the HYBRID filter. Landmarks are shown as red squares, while the smoothed vehicle path is shown as a blue line. The occupancy grid map is overlaid for clarity.

5.4 Live Experiments

The HYBRID filter was used to generate a map in a real urban environment, using a Segway RMP equipped with a SICK laser, as shown in Figure 5. The environment was approximately 135m×90m in size, and was mapped over a 3.75km path. The final map consisted of 452 landmarks, including foreground points, corners, doorways, and 46 retro-reflective strips. Data association was complicated by the presence of difficult-to-model objects such as bushes and trees, people occasionally walking through the area, and the tilting nature of the platform which caused it to make frequent observations of the ground when accelerating. Observations of the ground result in spurious corner features from the intersection of the ground with walls. The final map is shown in Figure 6, with an occupancy grid overlaid for clarity.

6 Conclusion

This paper presented a hybrid approach to SLAM which combines the strengths of two popular existing approaches: FastSLAM and EKF-SLAM. Using EKF-SLAM as a global mapping strategy allows uncertainty to be remembered over long vehicle trajectories. Using FastSLAM as a local mapping strategy minimises observation-rate linearisation errors and provides a significant level of robustness to clutter and uncertain associations. In experiments involving a large number of randomly

generated trials, the hybrid mapping approach was shown to be superior to either pure approach. The approach was further validated in a live experiment.

HybridSLAM can be seen as a mixture between pure FastSLAM and pure EKF-SLAM, with the time between map fusions as the mixing parameter. Consider the two extremes: if map fusion occurs on every iteration, HybridSLAM is largely equivalent to EKF-SLAM (with no sub-mapping). If map fusion never occurs, HybridSLAM is identical to FastSLAM.

While a simple heuristic for the mixing parameter was used in this work (local-map age equal to 10 seconds), better strategies are certainly possible and likely to increase robustness. For example, it may be prudent to delay map fusion when the vehicle distribution is multi-modal or particularly non-Gaussian, or when there is significant disagreement about landmark identities. Conversely, it may be of benefit to fuse sooner if particle diversity begins to drop.

Another option is to re-use information from slightly before and after the point of map fusion in both temporally-adjacent local maps, for the purposes of data association only. Since uncertainty about data association is not tracked in the back-end, re-use of information in this way will not lead to over-confidence.

Future work will also addresss the back-end. For simplicity, this presentation showed a monolithic EKF back-end, which is known to suffer from non-linearities and high computation in large environments. In future work, the back end could be replaced by more efficient Gaussian representations such as sparse information form [15] or submap methods such as ATLAS [7] of NCFM [1]. The submap forms also address long-term non-linearity.

References

1. Bailey, T.: Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis, University of Sydney (2002)
2. Bailey, T., Durrant-Whyte, H.: Simultaneous localisation and mapping (SLAM): Part II - state of the art. Robotics and Automation Magazine 13(3), 108–117 (2006)
3. Bailey, T., Nieto, J., Guivant, J., Stevens, M., Nebot, E.: Consistency of the EKF-SLAM algorithm. In: Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 3562–3568 (2006)
4. Bailey, T., Nieto, J., Nebot, E.: Consistency of the FastSLAM algorithm. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 424–429 (2006)
5. Bar-Shalom, Y., Li, X., Kirubarajan, T.: Estimation with Applications to Tracking and Navigation. John Wiley and Sons, Chichester (2001)
6. Bibby, C., Reid, I.: Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In: Proc. Robotics: Science and Systems (2007)
7. Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., Teller, S.: An atlas framework for scalable mapping. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 1899–1906 (2003)
8. Durrant-Whyte, H., Bailey, T.: Simultaneous localisation and mapping (SLAM): Part I – the essential algorithms. Robotics and Automation Magazine 13(2), 99–110 (2006)
9. Eliazar, A., Parr, R.: DP-SLAM 2.0. In: Proc. IEEE Intl. Conf. on Robotics and Automation, vol. 2, pp. 1314–1320 (2004)

10. Montemerlo, M., Thrun, S.: Simultaneous localization and mapping with unknown data association using FastSLAM. In: Proc. IEEE Intl. Conf. on Robotics and Automation, pp. 1985–1991 (2003)
11. Neira, J., Tardos, J.: Data association in stochastic mapping using the joint compatibility test. IEEE Transactions on Robotics and Automation 17(6), 890–897 (2001)
12. Perera, L., Wijesoma, W., Adams, M.: Data association in dynamic environments using a sliding window of temporal measurement frames. In: Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 753–758 (2005)
13. Stone, L., Barlow, C., Corwin, T.: Bayesian Multiple Target Tracking. Artech House (1999)
14. Tardos, J., Newman, P., Leonard, J.: Robust mapping and localization in indoor environments using sonar data. Intl. Journal of Robotics Research 21(4), 311–330 (2002)
15. Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., Durrant-Whyte, H.: Simultaneous localization and mapping with sparse extended information filters. International Journal of Robotics Research 23(7-8), 693–716 (2004)
16. Thrun, S., Montemerlo, M., Koller, D., Wegbreit, B., Nieto, J., Nebot, E.: FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. Journal of Machine Learning Research (2004)
17. Williams, S., Dissanayake, G., Durrant-Whyte, H.: An efficient approach to the simultaneous localisation and mapping problem. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 1, pp. 406–411 (2002)

Discovering a Point Source in Unknown Environments

Martin Burger, Yanina Landa, Nicolay M. Tanushev, and Richard Tsai

Abstract. We consider the inverse problem of discovering the location of a source from very sparse point measurements in a bounded domain that contains impenetrable (and possibly unknown) obstacles. We present an adaptive algorithm for determining the measurement locations, and ultimately, the source locations. Specifically, we investigate source discovery for the Laplace operator, though the approach can be applied to more general linear partial differential operators. We propose a strategy for the case when the obstacles are unknown and the environment has to be mapped out using a range sensor concurrently with source discovery.

1 Introduction

This work is motivated by robotic applications in which a robot, sent into an unknown environment, is supposed to discover the location of a signal source and place it under its line-of-sight in an efficient manner. The unknown environment contains non-penetrable solid obstacles that should be avoided along the robot's path. In this environment, the properties of the signal, such as the signal strength, are assumed to satisfy certain partial differential equations (PDEs) with appropriate boundary conditions. The robot can gather measurements from two different sensors: a range sensor that gives distance from the robot to the surrounding obstacles, and a sensor

Martin Burger
Institute for Computational and Applied Mathematics,
Westfaelische Wilhelms Universitaet Muenster
e-mail: martin.burger@wwu.de

Yanina Landa
University of California, Los Angeles
e-mail: ylanda@math.ucla.edu

Nicolay M. Tanushev and Richard Tsai
University of Texas at Austin
e-mail: {nicktan,ytsai}@math.utexas.edu

G.S. Chirikjian et al. (Eds.): Algorithmic Foundations of Robotics VIII, STAR 57, pp. 663–678.
springerlink.com

that measures the signal strength that is being emitted from the yet-to-be-located source. We will refer to the information from the range sensor as the visibility of the robot and to the information from the signal strength sensor as the signal. While measurements can be taken anywhere, we are interested in having the robot take very few measurements with its sensors. Our goal is to design an algorithm that determines how the robot should navigate through the environment and where along its path it should take measurements. Motion planning is a fundamental problem in robotics. It has been an active area of research since 1980's. A comprehensive presentation of motion planning techniques can be found in the book by Latombe [17]. In [10, 11], the authors provide a broad review of problems specific to the research in robotics. In the most general form, motion planning consists of finding a robot's path from a start position to a goal position, while avoiding obstacles and satisfying some constraints [17]. A large class of optimal path planning problems can be formulated as problems involving Hamilton-Jacobi equations [6, 1], and solved efficiently [29, 24].

This work is primarily concerned with visibility-based navigation towards a target whose position is unknown. As the observer moves in space, its visibility region changes, thus modifying the information available to the observer about the space or progress towards the goal. The visibility region is the set of points that can be joined with a line segment to the observer's position, without intersecting the obstacle boundaries. Online motion planning strategies are considered in [3, 7, 12, 13, 18, 21, 23, 26] among many others. In these works, the robot is in an unknown or partially known planar environment and uses its on-board sensors to navigate toward a target whose position is either known a priori or is recognized upon arrival. In particular, the bug-family algorithms are considered, which have two reactive modes of motion: moving toward the target between the obstacles and following the obstacles' boundaries. These two modes interact incrementally until the target is reached or is found to be out of reach. We remark here that this paper considers problems in which the source cannot be identified until suitable inverse problems are solved and problems in which one does not want to reach the source but to locate it.

Today, computational geometry and combinatorics are the primary tools to solve the visibility-based problems [30, 9]. These techniques are mainly concerned with defining visibility on polygons and more general planar environments with special structure. The combinatorial approach leads to fast and elegant solutions in simplified planar polygonal environments. However, this approach becomes increasingly complex in more realistic settings.

A recurring theme in robotics research has been the notion of minimal sensing, that is, completing a given task with minimum information necessary. Minimal sensing techniques are implemented in SLAM (simultaneous localization and mapping) [2] to situate a robot on a partially constructed map. The generalized Voronoi graph is used to encode the topological map of the environment. A graph-matching process then leads to localization of the robot. Although providing a compact representation, the lack of metric information makes localization extremely difficult. Furthermore, the statistical errors accumulated during the process of localization

can grossly distort the map and therefore the robot's ability to know its precise position. A potential field method is presented in [31]. The proposed model simulates steady-state heat transfer with variable thermal conductivity. The optimal path problem is then the same as a heat flow with minimal thermal resistance. The thermal resistance in the configuration space for all different orientations of the robot can then be superimposed.

This source problem is classified as an inverse problem, however, it differs greatly from many typical inverse problems, which assume simple domains and dense arrays of sensors at fixed locations. In [20], Ling et al explore such a situation in which they recover the exact locations of multiple sources in a Poisson equation, given an initial guess for the locations and Dirichlet data collected on the boundary of the domain. To accomplish this, they use the special form of the free space Green's function for the Laplacian. For inverse problems related to the heat equations with sources see [4, 5] and for a more general discussion of partial differential equations see [6].

When the obstacles are unknown, the environment needs to be mapped out as the robot moves so that attempts to take measurements inside the obstacles are avoided and the robot's path does not intersect obstacles. The previous work of [15] and [16] on mapping of obstacles in unknown domains using visibility is useful in this regard. In it the authors propose an algorithm to construct a high-order accurate representation of the portions of the solid surfaces that are visible from a vantage point and to generate the corresponding occlusion volume. Also they propose an algorithm to construct a piecewise linear path so that any point on the solid surfaces is seen by at least one vertex of the path and an accurate representation of the solids is constructed from the point clouds that are collected at the vertices of the path. In [15] a novel visibility-based algorithm is introduced to navigate toward a known target in an unknown bounded planar environment with obstacles. Similarly to [26], the observer navigates toward one of the visible horizons, or edges on the visibility map. In particular, to reach the goal, the edge that is the nearest to the target is chosen. To proceed the observer must overshoot the horizon by the amount inversely proportional to the curvature of the obstacle near the horizon, thus no boundary-following motion is required.

The [15] algorithm was motivated by the work of LaValle, Tovar et al. [25, 19, 26]. In [26], a single robot (observer) must be able to navigate through an unknown simply or multiply connected piecewise-analytic planar environment. The robot is equipped with a sensor that detects discontinuities in depth information and their topological changes in time. As a result of exploration, the region is characterized by the number of gaps and their relative positions. No distance or angular information is accumulated. In contrast, the [15] algorithm maps the obstacles in Cartesian coordinates as the observer proceeds through the environment, and utilizes the recovered information for further path planning. At the termination of the path all the obstacle boundaries are reconstructed to give a complete map of the environment. A practical implementation of this algorithm on an economical cooperative control tank-based platform is described in [14].

From our experience, the discovery of signal sources may be very insensitive to the presence of (or parts of) obstacles in sub-regions of the domain, possibly due

to the decay of the signal strength. This suggests that the visibility path algorithm in [15] and [16] should be modified adaptively according to previous measurements and estimations of the signal source location.

To illustrate the main ideas in this paper, consider the following problem

$$\begin{aligned} \triangle u(x) &= \delta(x-y) \quad \text{in} \quad D_\Omega \equiv D \setminus \Omega \\ u &= 0 \qquad\quad\ \ \text{on} \quad \Gamma\,, \end{aligned} \tag{1}$$

where u denotes the signal strength, D denotes a bounded domain, $\Omega \subsetneq D$ denotes the solid obstacles in the domain, $\Gamma = \partial D_\Omega$, and y denotes the (unknown) source location. One can view this PDE as giving a description of the steady state of a diffusion problem. We have chosen Dirichlet boundary conditions in this example but our methods can be adapted to other common boundary conditions such as Neumann boundary conditions.

Let $\psi(\cdot;z) : D \mapsto \mathbb{R}$ describe the visibility of the domain from an observing location $z \in D_\Omega$. We require that $\psi(\cdot;z)$ be a signed distance function such that the set $W_z := \{x \in D : \psi(x;z) < 0\}$ corresponds to the region, including the interior of the solid obstacles, that is occluded from the observing location z. This means that the line segment connecting z and any point in W_z must intersect with the obstacles Ω. Such visibility functions can be computed efficiently using the algorithms described in [28, 27, 16, 15].

A first, rather simple approach, would be to use gradient descent to determine the sample locations via the ordinary differential equation,

$$\frac{dX}{ds} = -\nabla u, \text{ with } X(0) = z_0\,. \tag{2}$$

However, there are two drawbacks to this method. First, it only works in cases where the Green's function has a specific structure, such as in the case for the Laplace operator. For problems involving the Helmholtz operator, the solutions are typically oscillatory and this gradient approach does not apply at all. Second, even for the Laplace operator, one can come up with a pathological configuration for the obstacles where the gradient vanishes at points other then the source.

We continue with the method proposed in this paper. At an observing location z_1, we can measure the signal strength $I_1 = u(z_1)$. We look at the solution to the adjoint problem,

$$\begin{aligned} \triangle v_1 &= \delta(x-z_1) \quad \text{in} \quad D_\Omega \\ v_1 &= 0 \qquad\quad\ \ \text{on} \quad \Gamma\,. \end{aligned} \tag{3}$$

Now, for $y \neq z_1$ we have

$$\begin{aligned} v_1(y) &= \int_{D_\Omega} \delta(x-y) v_1 dx = \int_{D_\Omega} v_1 \triangle u dx \\ u(z_1) &= \int_{D_\Omega} \delta(x-z_1) u dx = \int_{D_\Omega} u \triangle v_1 dx\,, \end{aligned} \tag{4}$$

and thus, by Green's identity, $v_1(y) = u(z_1) = I_1$. Therefore the source must lie on the I_1 level set of v_1,

$$y \in \{x \in D_\Omega : v_1(x) = I_1\} . \tag{5}$$

Next, based on the visibility information, we select the next observing location z_2 from the region that is not occluded to z_1, that is $z_2 \in D \setminus W_{z_1}$. Denote the signal strength at z_2 by $I_2 = u(z_2)$. The function v_2 can be computed and we can narrow down the possible locations of y,

$$y \in \{v_1 = I_1\} \cap \{v_2 = I_2\} . \tag{6}$$

We can repeat this procedure for more measurements.

Our proposed algorithm can handle obstacles of a rather large class, including very complicated non-convex obstacles, see Figures 5 and 6 for an example. The only constraint comes from the size of the underlying mesh used to obtain the solution of the PDE (3), since this grid has to resolve the features of the obstacles. The discretization of the domain results in a simple system of linear equations that has to be solved. We refer the reader to [8, 22] for a discussion and efficient methods for solving PDEs.

In the case when the obstacles Ω are unknown, the visibility functions $\psi(x; z_k)$ provide a convenient over approximation of Ω, since $\Omega \subseteq W_{z_k}$. This can be used in conjunction with a maximum principle for Poisson's equation to estimate the location of y. We will discuss this in greater detain in later sections.

2 Mathematical Formulation

In the most general setting that we will consider, the inverse source problem can be formulated as follows. Let $u(x)$ satisfy,

$$\begin{array}{ll} Lu = \alpha\delta(x-y) & \text{in } D_\Omega \equiv D \setminus \Omega \\ Bu = 0 & \text{on } \Gamma , \end{array} \tag{7}$$

where D is a bounded domain, Ω are the (possibly unknown) obstacles, $\Gamma = \partial D_\Omega$, L is a linear partial differential operator, B is an operator specifying the boundary conditions, and $\alpha > 0$. We will assume that at a given point $z \in D_\Omega$ we can sample u and the domain. That is, at a given z we can measure $u(z)$ and its derivatives and the visibility function $\psi(x; z)$. The inverse source location problem is to recover the source location y and the source strength α from a sequence of sample locations z_k.

The main approach that we will use in this problem is to look at the adjoint operator, L^*, with the appropriate boundary conditions B^*:

$$\begin{array}{ll} L^*v = F_z & \text{in } D_\Omega \\ B^*v = 0 & \text{on } \Gamma , \end{array} \tag{8}$$

for some distribution F_z with support $\{z\}$. Now, using the properties of the adjoint and assuming that $z \neq y$, $(Lu, v) - (u, L^* v) = 0$, and hence, $\alpha v(y) = F_z[u]$. For example, if we use $F = \delta$, we get $\alpha v(y) = u(z)$. Similary, for $F = -\partial_{x_1}\delta$, we get $\alpha v(y) = \partial_{x_1} u(z)$.

For unknown domains, we use the methods developed in [15] to find the visibility function and use it determine the sequence of sample locations z_k.

3 Poisson's Equation

In this section we consider the case when $L = \triangle$ in 2 dimensions with Dirichlet boundary conditions:

$$\begin{array}{rll} \triangle u = \alpha\delta(x-y) & \text{in} & D_\Omega \\ u = 0 & \text{on} & \Gamma\,. \end{array} \tag{9}$$

We note that this operator is self-adjoint and that the following maximum principle from PDE theory holds:

Theorem 3.1. *Let D_Ω be bounded and w satisfy*

$$\begin{array}{rll} \triangle w = 0 & \textit{in} & D_\Omega \\ w \leq 0 & \textit{on} & \Gamma\,, \end{array} \tag{10}$$

then $w \leq 0$ in D_Ω.

Now, supposed that we have an over-estimate for the obstacles Ω^+, so that $\Omega \subseteq \Omega^+$ and let v and v^+ satisfy (9) with obstacle sets Ω and Ω^+ respectively. Then, since the fundamental solution for the Laplacian for any domain is non-positive, $v \leq 0$ on $\Gamma^+ = \partial D_{\Omega^+}$. Let $w = v - v^+$, so that w satisfies the conditions of Theorem 3.1 for D_{Ω^+}. Thus, $w = v - v^+ \leq 0$ in D_{Ω^+}. Furthermore, if we extend v^+ to D_Ω by 0, we have that $v^+ \geq v$ in D_Ω. This fact will be used in the case of unknown obstacles.

3.1 Known Environment

At a sample location z_k, equation (2) gives us

$$\alpha v_k(y) = u(z_k), \quad \alpha w_{k1}(y) = \partial_{x_1} u(z_k), \quad \alpha w_{k2}(y) = \partial_{x_2} u(z_k)\,, \tag{11}$$

where v_k, w_{k1} and w_{k2} satisfy (8) with F equal to $\delta(x - z_k)$, $-\partial_{x_1}\delta(x - z_k)$ and $-\partial_{x_2}\delta(x - z_k)$, respectively. Since v_k is non-zero except at the boundary, we can also form,

$$\frac{w_{k1}(y)}{v_k(y)} = \frac{\partial_{x_1} u(z_k)}{u(z_k)} \quad \text{and} \quad \frac{w_{k2}(y)}{v_k(y)} = \frac{\partial_{x_2} u(z_k)}{u(z_k)}\,. \tag{12}$$

Thus, y is in the intersection of the $u(z_k)$ level set of v, the $\partial_{x_1} u(z_k)$ level set of w_1 and so on. Note that in the last two equations, α does not appear. As we take more

and more measurements z_k, the intersection of all of these sets will be smaller and smaller. Furthermore, for a pair of measurements j and k, we can form,

$$\frac{v_k(y)}{v_l(y)} = \frac{u(z_k)}{u(z_l)} , \tag{13}$$

and so on. Note that these are also independent of α.

3.2 Unknown Environment

At a sample location z_k, we have the visibility function $\psi(x;z_k)$. From this function we can obtain an over-estimate of the obstacles, Ω_k^+, such that $\Omega \subseteq \Omega_k^+$. After K measurements, we let

$$\Omega^+ = \bigcap_{k=1}^{K} \Omega_k^+ , \text{ and}$$

$$\begin{array}{rll} \triangle v_k^+ = \delta(x-z_k) & \text{in} & D_{\Omega^+} \equiv D \setminus \Omega^+ \\ v_k^+ = 0 & \text{on} & \Gamma^+ , \end{array} \tag{14}$$

where $\Gamma^+ = \partial D_{\Omega^+}$.

The results from section 3.1 apply and for the v_k defined in that section, $\alpha v_k(y) = u(z_k)$, but since we don't know Ω, we cannot find v_k. However, by Theorem 3.1, $v_k \leq 0$ on Γ^+, since $\Gamma^+ \subset D_\Omega$. If we consider the difference $w = v_k - v_k^+$, we see that w satisfies (10), and Theorem 3.1 implies that $w \leq 0$. This maximum principle gives us that for $\alpha \geq 0$,

$$\alpha v_k^+(y) \geq \alpha v_k(y) = u(z_k) .$$

Thus,

$$y \in \bigcap_{k=1}^{K} \{x | \alpha v_k^+(x) \geq u(z_k)\} .$$

In the case when α is unknown, we would like to find an α independent set which includes y. From section 3.1, for a pair of samples k and l, we have

$$\frac{v_k(y)}{v_l(y)} = \frac{u(z_k)}{u(z_l)} \quad \text{and thus,} \quad \frac{v_k^+(y)}{v_l(y)} \geq \frac{u(z_k)}{u(z_l)} .$$

Now, let Ω^- be an under-estimate of the obstacles, so that $\Omega^- \subseteq \Omega$. A simple choice for Ω^- is the empty set (no obstacles). Also, let v^- satisfy (14) for Ω^-. By the maximum principle, $v_l \geq v^-$. Thus,

$$\frac{v_k^+(y)}{v^-(y)} \geq \frac{u(z_k)}{u(z_l)},$$

which is independent of α as desired.

3.3 Numerical Experiments and the Proposed Algorithm

We model the δ-source as a sharply rescaled Gaussian centered at the prescribed location. To determine the location of the source, we build a probability density as follows. For each measurement, we let the probability density, $p_k(x)$, be constant in the possible region (it may be a curve) and have Gaussian drop-off away from this region. After k different measurements, we let the probability density be

$$p(x) = c_0 \prod_{j=1}^{k} p_j(x)^{1/k}, \quad \text{with } c_0 \text{ s.t.} \int_\Omega p(x)dx = 1. \tag{15}$$

Known Strength, Known Environment

For this experiment, we assume that the source has strength $\alpha = 1$ and use Algorithm 1. We sample $u(x)$ at 3 locations, which results in 9 level sets if we use the level sets given by (11). The domain and results are given in Figure 1.

Algorithm 1. Source detection in known environment.

1: $u(z)$: solution of equation (9) that can be measured for any z.
2: k = 1
3: z_k: vantage point outside the occluding objects
4: compute v_k: solution of Equation (9) and any of the $w_{k1}, \ldots,$ that are availiable
5: compute p as in Equation (15)
6: **while** p is not localized **do**
7: $k = k+1$
8: chose z_k to be outside of the set $\{x : v_{k-1} > u(z_{k-1})\}$
9: compute v_k: solution of (9) and any of the $w_{k1}, \ldots,$ that are availiable
10: re-compute p as in Equation (15)
11: **end while**

Alternatively, we can use only v_k along with all pairs (13). The results for 3 measurements (total of 6 level sets after the 3rd) are shown in Figure 2.

Unknown Strength, Known Environment

For this case, we assume that the strength α is unknown and use Algorithm 1. We sample $u(x)$ at 3 locations. Since the strength is unknown, we use equaitons (13). After locating the source, its strength can be approximated using, $\alpha = u(z_k)/v_k(y)$. The results are shown in Figure 3. The actual source location is $(0.200, 0.400)$ and

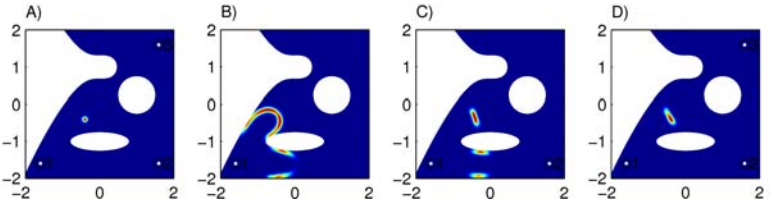


Fig. 1. Location of a source with known strength in a known environment for Poisson's equation. Location based on 3 measurements with v_k, w_{k1} and w_{k2}. Figures: A) The known environment, source and sample (1, 2, 3) locations; B) $p(x)$ after 1 measurement; C) $p(x)$ after 2 measurements; D) $p(x)$ after 3 measurements. Blue is zero probability; Red is high probability.

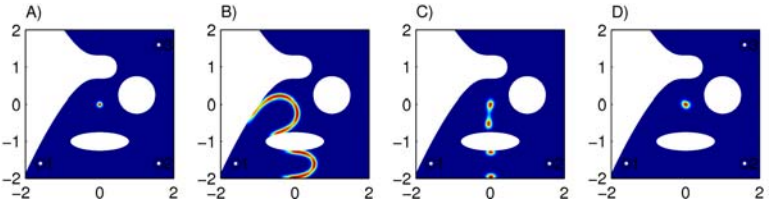


Fig. 2. Location of a source with known strength in a known environment for Poisson's equation. Location based on 3 measurements with v_k and pairwise combinations. Figures: A) The known environment, source and sample (1, 2, 3) locations; B) $p(x)$ after 1 measurement; C) $p(x)$ after 2 measurements; D) $p(x)$ after 3 measurements. Blue is zero probability; Red is high probability.

its strength is $\alpha = 1.817547$. The probability p after 3 measurements has a maximum at $(0.208, 0.400)$. The estimates for α are 1.861106, 1.837540, 1.767651, and the average is 1.822099.

Known Strength, Unknown Environment

In order to detect a source of given strength in an unknown environment the observer utilizes visibility information to proceed through the environment and to narrow down the region of possible source locations. In particular, let $\psi(\cdot, z_k)$ be the visibility level set function corresponding to the vantage point z_k. Then, $\{\psi(\cdot; z_k) \geq 0\}$ is the visible portion of D and $\{\psi(\cdot; z_k) < 0\}$ is invisible. Let Ψ_k denote joint visibility along the path. In the level set framework, $\Psi_k = \max_{j=1,\ldots,k}\{\psi(\cdot, z_j)\}$. The remaining occluded set $\{x \in D : \Psi_k(x) < 0\}$ may be used as an over-approximation of the obstacles Ω_k^+. Note, as the observer explores more of D, Ω_k^+ becomes a better approximation Ω. Thus, the $u(z_k)$ level set of v_k^+ would pass closer to the source location y.

Furthermore, let $\{q_j\}_{j=1}^M$ be the filtered out visible points on the boundaries of the obstacles, collected along the observer's path $\{z_1, \ldots, z_k\}$, see [15, 16] for details. To construct an under-approximation of the obstacles Ω_k^-, we take the union of all

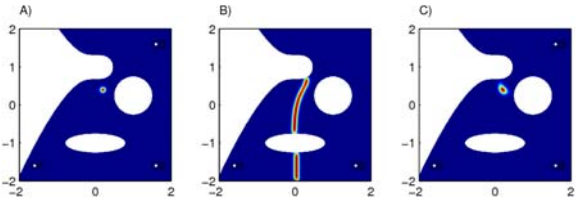


Fig. 3. Location of a source with unknown strength in a known environment for Poisson's equation. Location based on 3 measurements with v_k pairwise combinations. Figures: A) The known environment, source and sample (1, 2, 3) locations; B) $p(x)$ after 2 measurements; C) $p(x)$ after 3 measurements. Blue is zero probability; Red is high probability. Actual parameters: source location $(0.200, 0.400)$ with strength 1.817547. Location results: $(0.208, 0.400)$. Strength estimates, 1) 1.861106, 2) 1.837540, 3) 1.767651. Averaged $\alpha = 1.822099$.

ε-balls $B_\varepsilon(q_j)$, touching the visible points, such that $B_\varepsilon(q_j) \subseteq \{\Psi_k \leq 0\}$. Dirichlet boundary conditions are enforced at the union of the boundaries of $B_\varepsilon(q_j)$. Then, using Theorem 3.1, we may "sandwich" the location of the source $y \in \{x \in D : v_k^-(x) \leq u(z_k) \leq v_k^+(x)\}$.

As the observer proceeds through the environment, the next step along the path is chosen in the currently visible region, so that the resulting path avoids obstacles and is continuous and consists of a finite number of steps as in [15]. We adopt the algorithm in [15] to navigate through the unknown environment, in which the observer approaches one of the visible horizons, or edges on the piecewise-smooth visibility map, defined in [15]. The next step z_{k+1} is obtained by overshooting the horizon location by the amount inversely proportional to the curvature of the obstacle's boundary near the horizon.

To optimize the search, we choose a direction so that $z_{k+1} \in \{W_k \geq 0\}$, if the continuity of the path can be preserved. Otherwise, we simply proceed towards the nearest horizon, as was proposed in [15]. The algorithm terminates when the entire set $\{W_k \geq 0\}$ is visible from the vertices along the path. Note, that in most cases the proposed location algorithm would terminate prior to full mapping of the environment. However, if the environment has been fully explored before the source was located, the algorithm for the known environment may be applied.

Finally, we would like to remark that according to [15], the environment is considered to be completely explored when all the horizons detected along the path have been cleared. The observer may return to an earlier vantage point along the path to see other horizons. Therefore, the resulting path may branch out. The complete search strategy is described in Algorithm 2 below.

Note that v_k^+ and v_k^- are the level set functions. Then, for a given k, the set $\{x \in D : v_k^-(x) \leq u(z_k) \leq v_k(x)^+\}$, containing the source, is defined by another level set function W_k, positive in the interior of the set and negative outside. Numerically, W_k is defined in step 10 of the above algorithm. As the observer proceeds through the environment, we take the intersection of all such sets corresponding to each observing location. In the level set framework, this translates to $\min_{j=1,\ldots,k}\{W_j\}$,

Algorithm 2. Source detection in unknown environment with known strength.

1: $u(z)$: solution of equation (9) that can be measured for any z.
2: k = 1
3: z_k: vantage point outside the occluding objects
4: $\psi(\cdot, z_k)$: visibility with respect to z_k
5: Ψ_k: joint visibility along the path
6: construct Ω_k^+: over-estimate of Ω with respect to z_k
7: construct Ω_k^-: under-estimate of Ω with respect to z_k
8: compute v_k^+: solution of Equation (9) with obstacles Ω^+
9: compute v_k^-: solution of Equation (9) with obstacles Ω^-
10: set $W_k(x) := -(v_k^+(x) - u(z_k))(v_k^-(x) - u(z_k)), x \in D$
11: **while** $\{W_k \geq 0\} \not\subseteq \{\Psi_k \geq 0\}$ **do**
12: k = k+1
13: set $\Psi_k = \max\{\psi(\cdot, z_k), \Psi_{k-1}\}$
14: construct Ω_k^+, Ω_k^-
15: compute v_k^+, v_k^-
16: set $W_k(x) := \min\{-(v_k^+(x) - u(z_k))(v_k^-(x) - u(z_k)), W_{k-1}(x)\}, x \in D$
17: **if** $\{W_k \geq 0\} \cap \{\psi(\cdot, z_k) > 0\} \neq \emptyset$ **then**
18: choose $z_k \in \{W_k \geq 0\} \cap \{\psi(\cdot, z_k) > 0\}$
19: **else**
20: choose $z_k \in \{\psi(\cdot, z_k) > 0\}$ according to the exploration algorithm in [15]
21: **end if**
22: **end while**

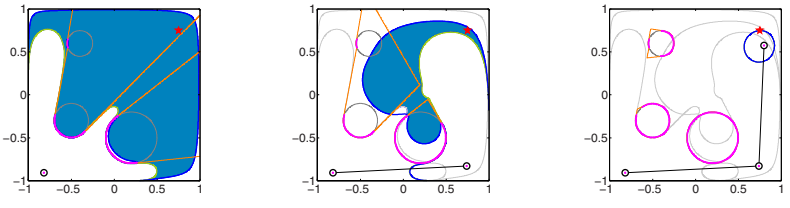


Fig. 4. Unknown environment, known source strength. The source is located at $(0.75, 0.75)$. Orange contour is boundary of Ω_k^+ and the magenta contour is the boundary of Ω_k^-. The blue region is $W_k \geq 0$, the green contour is the $u(z_k)$ level set of v_k^+ and the blue contour is the $u(z_k)$ level set of v_k^-.

computed in step 16. Similarly, joint visibility along the path Ψ_k is computed as $\max_{j=1,\ldots,k}\{\psi(\cdot, z_j)\}$ in step 13.

Figures 4, 5, and 6 demonstrate the performance of Algorithm 2. In all these figures, the over-approximation of the obstacles Ω^+, based on joint visibility, is depicted by the orange contour, and the under-approximation Ω^-, based on ε-balls around the visible boundary points, is depicted by the magenta contour. The $u(z_k)$ level set of v_k^+ is shown in green and the $u(z_k)$ level set of v_k^- is shown in blue. The blue region is the set $\{W_k \geq 0\}$. The location of the source is marked by the red star and the path is shown in black, with circles indicating the discrete steps.

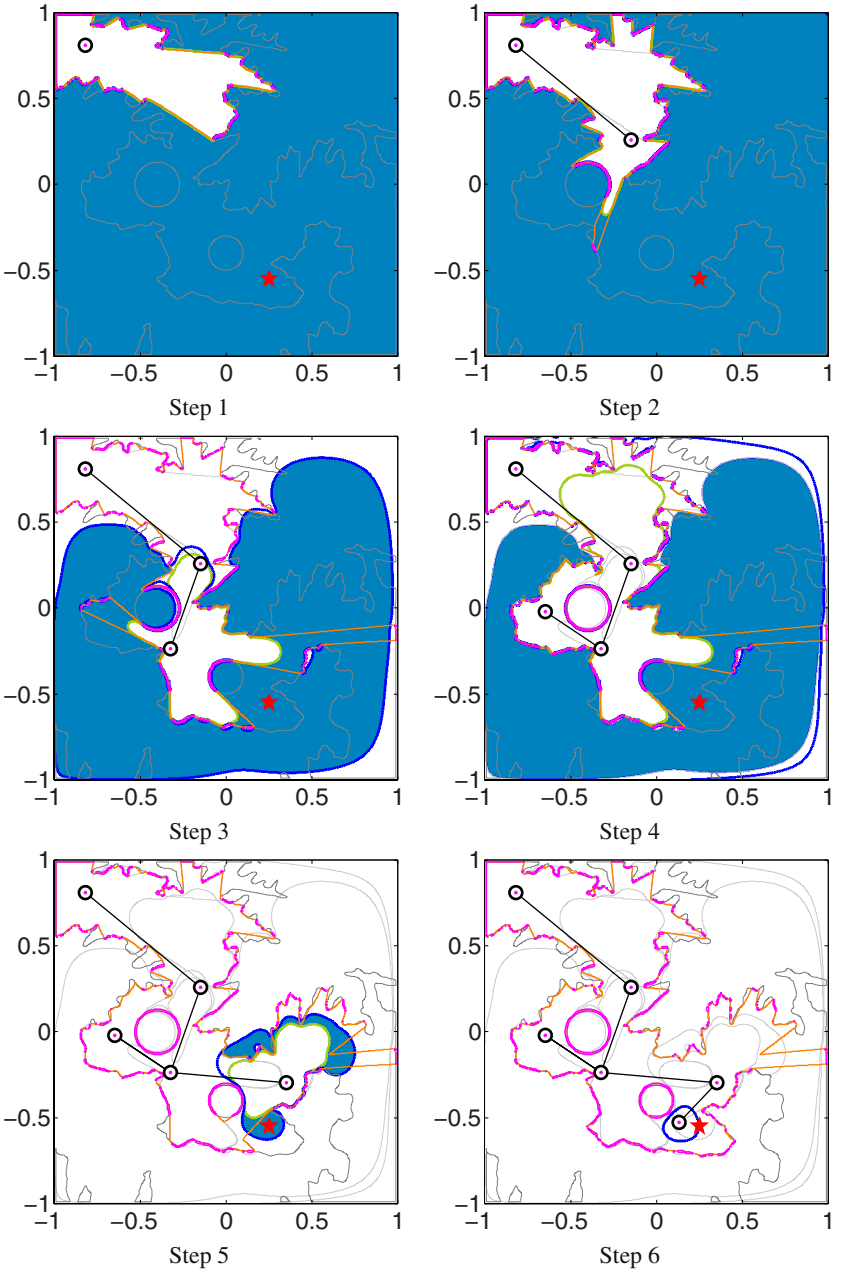


Fig. 5. Unknown environment, known source strength. The source is located at $(0.25, -0.55)$. Orange contour is boundary of Ω_k^+ and the magenta contour is the boundary of Ω_k^-. The blue region is $W_k \geq 0$, the green contour is the $u(z_k)$ level set of v_k^+ and the blue contour is the $u(z_k)$ level set of v_k^-.

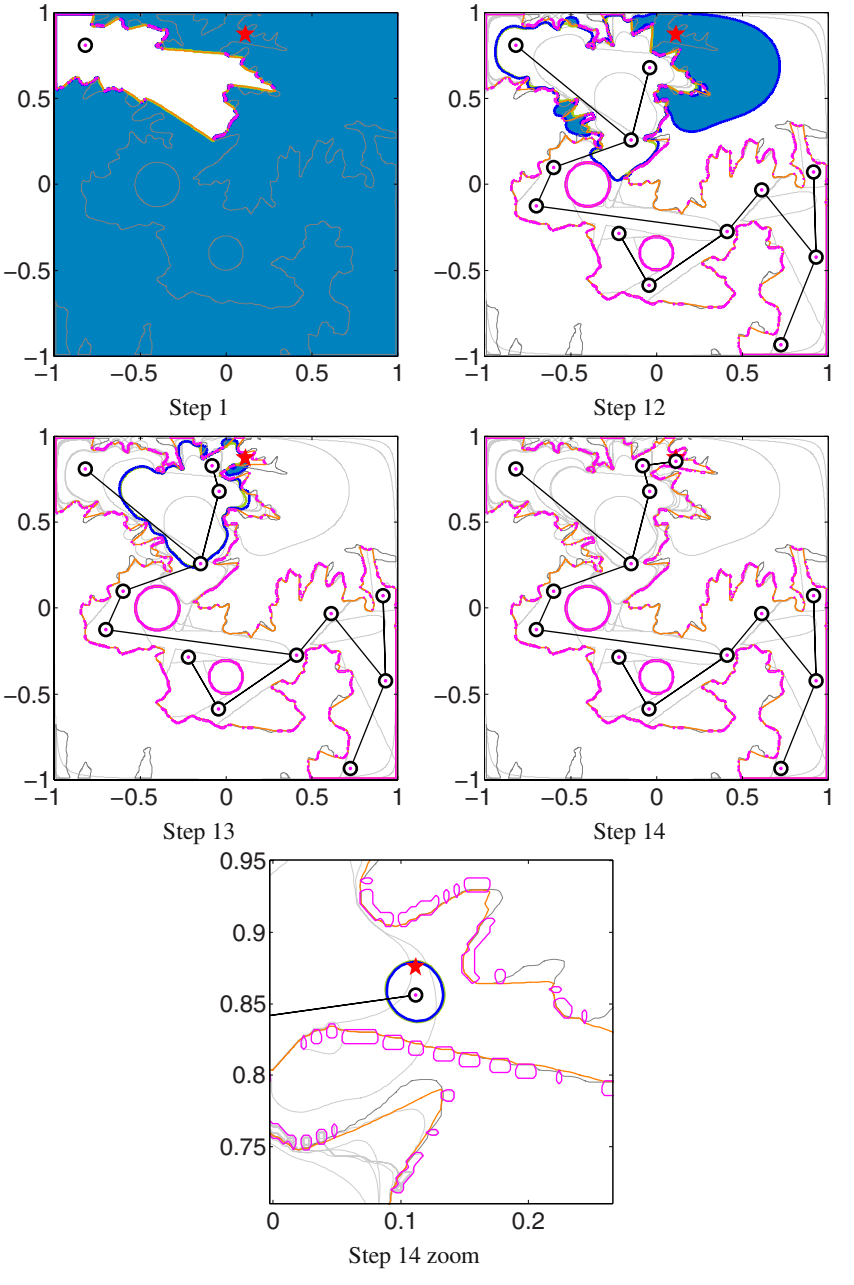


Fig. 6. Unknown environment, known source strength. The source is located at $(0.112, 0.876)$. Orange contour is boundary of Ω_k^+ and the magenta contour is the boundary of Ω_k^-. The blue region is $W_k \geq 0$, the green contour is the $u(z_k)$ level set of v_k^+ and the blue contour is the $u(z_k)$ level set of v_k^-. Steps 2 through 11 are skipped since no information regarding the source location is available.

Figure 4 shows a simple environment with three disk-shaped obstacles. The source is located at $(0.75, 0.75)$. The observer may not see the source from its initial position at $(-0.82, -0.91)$. The blue region $\{W_1 \geq 0\}$ almost overlaps with the invisible set $\{\psi_1 < 0\}$. The next vantage point is chosen to be inside the blue region. One can see that after two steps the region $\{W_2 \geq 0\}$, containing the source, has shrunk significantly. Finally, after three steps, $u(z_k)$ level sets of v_k^+ and v_k^- coincide, and the source is located somewhere on the curve $\{W_3 = 0\}$. Since this set is entirely visible from the observer's position, the search is complete. Note that the environment has not been entirely explored up to this point.

Figure 5 depicts a much more complex example. Here, the environment is constructed from a single slice of the Grand Canyon height-map[1] at a fixed elevation. The complexity of this particular environment is in the fractal-like structure of its boundaries, which complicates the visibility-based navigation. We further increased the complexity of the Grand Canyon terrain by adding two disk-shaped holes to the interior of the region. The source is concealed in a small bay with coordinates $(0.25, -0.55)$. At step 1 the blue region $\{W_1 \geq 0\}$, containing the source overlaps with the invisible set $\{\psi_1 < 0\}$. Therefore the observer simply approaches the nearest edge to arrive at z_2. From now on there is a preferable direction to approach. The next observing position z_3 is chosen according to step 18 of the algorithm. Now there are two possible directions to investigate. The observer chooses the nearest one to arrive at z_4. Since there are no new horizons at z_4, the observer backtracks to z_3 and explores the second choice horizon. As the observer approaches the source, the blue region shrinks. At z_5 the observer chooses the nearest of three possible horizons. Finally, the entire set of possible source locations is visible from z_6 and, therefore, the algorithm terminates. We remark that the source has been found long before the entire environment has been explored.

Finally, Figure 6 depicts the most complicated example. The source is concealed in a small cave at $(0.112, 0.876)$. Steps 1 through 12 are chosen according to the original [15] exploration algorithm, since the sets $\{W_2 \geq 0\}$ and $\{\Psi_k \geq 0\}$ coincide for $k = 1, \ldots, 12$. Finally, the observer backtracks to z_2 to clear previously unexplored horizons. At z_{14} the set containing the source becomes visible. In this example, the observer must explore almost the entire region to finally locate the source.

4 Conclusion

In this paper, we have developed an algorithm that can locate a source of unknown strength for a generic partial differential operator in a bounded domain with obstacles. The algorithm relies on the solution of the adjoint problem and the reciprocity that exists between the operator and its adjoint. We have shown examples for the case of Poisson's equation.

In the case of unknown obstacles, we have proposed a method for locating the source which is based on previous unknown environment exploration methods and relies on the maximum principle to determine a set of possible source locations. This

[1] Data from: `ftp.research.microsoft.com/users/hhoppe/data/gcanyon`

algorithm also works in the case of unknown source strength. Several examples for Poisson's equation were shown.

Acknowledgments

Burger's research is supported by the German Science Foundation, DFG, via the project "Regularisierung mit singulären Energien". Landa's research is supported by an ARO MURI subcontract from U. South Carolina and an ONR MURI subcontract from Stanford University. Tanushev's research is partially supported by NSF DMS-0636586 and ARO MURI U. South Carolina Subaward 07-1409. Tsai's research is supported in parts by NSF DMS-0513394, a Sloan Fellowship, and ARO MURI U. South Carolina Subaward 07-1409.

References

1. Bardi, M., Capuzzo-Dolcetta, I.: Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations. Birkhäuser Boston Inc., Boston (1997); With appendices by Maurizio Falcone and Pierpaolo Soravia
2. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. The MIT Press, Cambridge (2005)
3. Deng, X., Kameda, T., Papadimitriou, C.: How to learn an unknown environment. i: the rectilinear case. J. ACM 45(2), 215–245 (1998)
4. El Badia, A., Ha-Duong, T.: On an inverse source problem for the heat equation. Application to a pollution detection problem. J. Inverse Ill-Posed Probl. 10(6), 585–599 (2002)
5. El Badia, A., Ha-Duong, T., Hamdi, A.: Identification of a point source in a linear advectiondispersion- reaction equation: application to a pollution source problem. Inverse Problems 21(3), 1121–1136 (2005)
6. Evans, L.C.: Partial Differential Equations. Amer. Math. Soc. (1998)
7. Gabriely, Y., Rimon, E.: Competitive complexity of mobile robot on line motion planning problems. In: WAFR, pp. 249–264 (2004)
8. Gibou, F., Fedkiw, R.P., Cheng, L.-T., Kang, M.: A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. J. Comput. Phys. 176(1), 205–227 (2002)
9. Goodman, J.E., O'Rourke, J. (eds.): Handbook of discrete and computational geometry, 2nd edn. CRC Press LLC, Boca Raton (2004)
10. Halperin, D., Kavraki, L.E., Latombe, J.C.: Robot algorithms. In: Atallah, M. (ed.) Handbook of Algorithms and Theory of Computation. CRC Press, Boca Raton (1999)
11. Halperin, D., Kavraki, L.E., Latombe, J.C.: Robotics. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry. CRC Press, Boca Raton (2004)
12. Kamon, I., Rivlin, E.: Sensory-based motion planning with global proofs. TRA 13(6), 814–822 (1997)
13. Kao, M.-Y., Reif, J.H., Tate, S.R.: Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In: SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms) (1993)

14. Landa, Y., Galkowski, D., Huang, Y., Joshi, A., Lee, C., Leung, K., Malla, G., Treanor, J., Voroninski, V., Bertozzi, A., Tsai, Y.-H.: Robotic path planning and visibility with limited sensor data. In: American Control Conference, ACC 2007, July 2007, pp. 5425–5430 (2007)
15. Landa, Y., Tsai, R.: Visibility of point clouds and exploratory path planning in unknown environments. CAM Report 08-28 (Submitted to Communications in Mathematical Sciences) (2008)
16. Landa, Y., Tsai, R., Cheng, L.-T.: Visibility of point clouds and mapping of unknown environments. Lecture Notes in Computational Science and Engineering, pp. 1014–1025. Springer, Heidelberg (2006)
17. Latombe, J.-C.: Robot motion planning. Kluwer Academic Publishers, Dordrecht (1991)
18. Laubach, S.L., Burdick, J.W.: An autonomous sensor-based path-planning for planetary microrovers. In: ICRA (1999)
19. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
20. Ling, L., Hon, Y.C., Yamamoto, M.: Inverse source identification for Poisson equation. Inverse Problems in Science and Engineering 13(4), 433–447 (2005)
21. Lumelsky, V.J., Stepanov, A.A.: Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica 2, 403–430 (1987)
22. Osher, S., Fedkiw, R.P.: Level set methods: an overview and some recent results. J. Comput. Phys. 169(2), 463–502 (2001)
23. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theoretical Comput. Sci. 84, 127–150 (1991)
24. Sethian, J.A.: Level set methods and fast marching methods, 2nd edn. Cambridge University Press, Cambridge (1999); Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science
25. Tovar, B., Guilamo, L., LaValle, S.M.: Gap navigation trees: A minimal representation for visibility-basedtasks. In: Erdmann, M., Hsu, D., Overmars, M., van der Stappen, A.F. (eds.) Algorithmic Foundations of Robotics VI. Springer, Heidelberg (2005)
26. Tovar, B., Murrieta-Cid, R., LaValle, S.M.: Distance-optimal navigation in an unknown environment withoutsensing distances. IEEE Trans. Robotics 23(3), 506–518 (2007)
27. Tsai, R., Kao, C.-Y.: A level set formulation for visibility and its discretization. Journal of Scientific Computing (to appear)
28. Tsai, Y.-H.R., Cheng, L.-T., Osher, S., Burchard, P., Sapiro, G.: Visibility and its dynamics in a PDE based implicit framework. J. Comput. Phys. 199(1), 260–290 (2004)
29. Tsitsiklis, J.: Efficient algorithms for globally optimal trajectories. IEEE Transactions on Automatic Control 40(9), 1528–1538 (1995)
30. Urrutia, J.: Art gallery and illumination problems. In: Sack, J.R., Urrutia, J. (eds.) Handbook of Computational Geometry. Elsevier Science, Amsterdam (2000)
31. Wang, Y., Chirikjian, G.: A new potential field method for robot path planning. In: Proceedings of the IEEE International Robotics and Automation Conference, San Francisco, CA, pp. 977–982 (2000)

Author Index

Springer Tracts in Advanced Robotics

Edited by B. Siciliano, O. Khatib and F. Groen

Further volumes of this series can be found on our homepage: springer.com

Vol. 56: Buehler, M.; Iagnemma, K.; Singh, S. (Eds.)
The DARPA Urban Challenge – Autonomous Vehicles in City Traffic
625 p. 2009 [978-3-642-03990-4]

Vol. 55: Stachniss, C.
Robotic Mapping and Exploration
196 p. 2009 [978-3-642-01096-5]

Vol. 54: Khatib, O.; Kumar, V.; Pappas, G.J. (Eds.)
Experimental Robotics: The Eleventh International Symposium
579 p. 2009 [978-3-642-00195-6]

Vol. 53: Duindam, V.; Stramigioli, S.
Modeling and Control for Efficient Bipedal Walking Robots
211 p. 2009 [978-3-540-89917-4]

Vol. 52: Nüchter, A.
3D Robotic Mapping
201 p. 2009 [978-3-540-89883-2]

Vol. 51: Song, D.
Sharing a Vision
186 p. 2009 [978-3-540-88064-6]

Vol. 50: Alterovitz, R.; Goldberg, K.
Motion Planning in Medicine: Optimization and Simulation Algorithms for Image-Guided Procedures
153 p. 2008 [978-3-540-69257-7]

Vol. 49: Ott, C.
Cartesian Impedance Control of Redundant and Flexible-Joint Robots
190 p. 2008 [978-3-540-69253-9]

Vol. 48: Wolter, D.
Spatial Representation and Reasoning for Robot Mapping
185 p. 2008 [978-3-540-69011-5]

Vol. 47: Akella, S.; Amato, N.; Huang, W.; Mishra, B.; (Eds.)
Algorithmic Foundation of Robotics VII
524 p. 2008 [978-3-540-68404-6]

Vol. 46: Bessière, P.; Laugier, C.; Siegwart R. (Eds.)
Probabilistic Reasoning and Decision Making in Sensory-Motor Systems
375 p. 2008 [978-3-540-79006-8]

Vol. 45: Bicchi, A.; Buss, M.; Ernst, M.O.; Peer A. (Eds.)
The Sense of Touch and Its Rendering
281 p. 2008 [978-3-540-79034-1]

Vol. 44: Bruyninckx, H.; Přeučil, L.; Kulich, M. (Eds.)
European Robotics Symposium 2008
356 p. 2008 [978-3-540-78315-2]

Vol. 43: Lamon, P.
3D-Position Tracking and Control for All-Terrain Robots
105 p. 2008 [978-3-540-78286-5]

Vol. 42: Laugier, C.; Siegwart, R. (Eds.)
Field and Service Robotics
597 p. 2008 [978-3-540-75403-9]

Vol. 41: Milford, M.J.
Robot Navigation from Nature
194 p. 2008 [978-3-540-77519-5]

Vol. 40: Birglen, L.; Laliberté, T.; Gosselin, C.
Underactuated Robotic Hands
241 p. 2008 [978-3-540-77458-7]

Vol. 39: Khatib, O.; Kumar, V.; Rus, D. (Eds.)
Experimental Robotics
563 p. 2008 [978-3-540-77456-3]

Vol. 38: Jefferies, M.E.; Yeap, W.-K. (Eds.)
Robotics and Cognitive Approaches to Spatial Mapping
328 p. 2008 [978-3-540-75386-5]

Vol. 37: Ollero, A.; Maza, I. (Eds.)
Multiple Heterogeneous Unmanned Aerial Vehicles
233 p. 2007 [978-3-540-73957-9]

Vol. 36: Buehler, M.; Iagnemma, K.; Singh, S. (Eds.)
The 2005 DARPA Grand Challenge – The Great Robot Race
520 p. 2007 [978-3-540-73428-4]

Vol. 35: Laugier, C.; Chatila, R. (Eds.)
Autonomous Navigation in Dynamic Environments
169 p. 2007 [978-3-540-73421-5]

Vol. 34: Wisse, M.; van der Linde, R.Q.
Delft Pneumatic Bipeds
136 p. 2007 [978-3-540-72807-8]

Vol. 33: Kong, X.; Gosselin, C.
Type Synthesis of Parallel Mechanisms
272 p. 2007 [978-3-540-71989-2]

Vol. 30: Brugali, D. (Ed.)
Software Engineering for Experimental Robotics
490 p. 2007 [978-3-540-68949-2]

Vol. 29: Secchi, C.; Stramigioli, S.; Fantuzzi, C.
Control of Interactive Robotic Interfaces – A Port-Hamiltonian Approach
225 p. 2007 [978-3-540-49712-7]

Vol. 28: Thrun, S.; Brooks, R.; Durrant-Whyte, H. (Eds.)
Robotics Research – Results of the 12th International Symposium ISRR
602 p. 2007 [978-3-540-48110-2]

Vol. 27: Montemerlo, M.; Thrun, S.
FastSLAM – A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics
120 p. 2007 [978-3-540-46399-3]

Vol. 26: Taylor, G.; Kleeman, L.
Visual Perception and Robotic Manipulation – 3D Object Recognition, Tracking and Hand-Eye Coordination
218 p. 2007 [978-3-540-33454-5]

Vol. 25: Corke, P.; Sukkarieh, S. (Eds.)
Field and Service Robotics – Results of the 5th International Conference
580 p. 2006 [978-3-540-33452-1]

Vol. 24: Yuta, S.; Asama, H.; Thrun, S.; Prassler, E.; Tsubouchi, T. (Eds.)
Field and Service Robotics – Recent Advances in Research and Applications
550 p. 2006 [978-3-540-32801-8]

Vol. 23: Andrade-Cetto, J,; Sanfeliu, A.
Environment Learning for Indoor Mobile Robots – A Stochastic State Estimation Approach to Simultaneous Localization and Map Building
130 p. 2006 [978-3-540-32795-0]

Vol. 22: Christensen, H.I. (Ed.)
European Robotics Symposium 2006
209 p. 2006 [978-3-540-32688-5]

Vol. 21: Ang Jr., H.; Khatib, O. (Eds.)
Experimental Robotics IX – The 9th International Symposium on Experimental Robotics
618 p. 2006 [978-3-540-28816-9]

Vol. 20: Xu, Y.; Ou, Y.
Control of Single Wheel Robots
188 p. 2005 [978-3-540-28184-9]

Vol. 19: Lefebvre, T.; Bruyninckx, H.; De Schutter, J. Nonlinear Kalman Filtering for Force-Controlled Robot Tasks
280 p. 2005 [978-3-540-28023-1]

Vol. 18: Barbagli, F.; Prattichizzo, D.; Salisbury, K. (Eds.)
Multi-point Interaction with Real and Virtual Objects
281 p. 2005 [978-3-540-26036-3]

Vol. 17: Erdmann, M.; Hsu, D.; Overmars, M.; van der Stappen, F.A (Eds.)
Algorithmic Foundations of Robotics VI
472 p. 2005 [978-3-540-25728-8]

Vol. 16: Cuesta, F.; Ollero, A.
Intelligent Mobile Robot Navigation
224 p. 2005 [978-3-540-23956-7]

Vol. 15: Dario, P.; Chatila R. (Eds.)
Robotics Research – The Eleventh International Symposium
595 p. 2005 [978-3-540-23214-8]

Vol. 14: Prassler, E.; Lawitzky, G.; Stopp, A.; Grunwald, G.; Hägele, M.; Dillmann, R.; Iossifidis. I. (Eds.)
Advances in Human-Robot Interaction
414 p. 2005 [978-3-540-23211-7]

Vol. 13: Chung, W.
Nonholonomic Manipulators
115 p. 2004 [978-3-540-22108-1]

Vol. 12: Iagnemma K.; Dubowsky, S.
Mobile Robots in Rough Terrain – Estimation, Motion Planning, and Control with Application to Planetary Rovers
123 p. 2004 [978-3-540-21968-2]