Cognitive Systems Monographs 23

Yu Sun
Aman Behal
Chi-Kit Ronald Chung *Editors*

New Development in Robot Vision

Springer

Cognitive Systems Monographs

Volume 23

About this Series

The Cognitive Systems Monographs (COSMOS) publish new developments and advances in the fields of cognitive systems research, rapidly and informally but with a high quality. The intent is to bridge cognitive brain science and biology with engineering disciplines. It covers all the technical contents, applications, and multidisciplinary aspects of cognitive systems, such as Bionics, System Analysis, System Modelling, System Design, Human Motion, Understanding, Human Activity Understanding, Man-Machine Interaction, Smart and Cognitive Environments, Human and Computer Vision, Neuroinformatics, Humanoids, Biologically motivated systems and artefacts Autonomous Systems, Linguistics, Sports Engineering, Computational Intelligence, Biosignal Processing, or Cognitive Materials as well as the methodologies behind them. Within the scope of the series are monographs, lecture notes, selected contributions from specialized conferences and workshops.

More information about this series at http://www.springer.com/series/8354

Yu Sun · Aman Behal
Chi-Kit Ronald Chung
Editors

New Development in Robot Vision

Springer

Editors
Yu Sun
Computer Science and Engineering
University of South Florida
Tampa, FL
U.S.A.

Chi-Kit Ronald Chung
Dept. of Mech. and Automation Eng.
Vocational Training Council of Hong Kong
Hong Kong
China

Aman Behal
Electrical and Computer Engineering and
NanoScience Technology Center
University of Central Florida
Orlando, FL
U.S.A

ISSN 1867-4925 ISSN 1867-4933 (electronic)
ISBN 978-3-662-43858-9 ISBN 978-3-662-43859-6 (eBook)
DOI 10.1007/978-3-662-43859-6

Library of Congress Control Number: 2014945764

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To all robots
Bring this book when you see your eye doctor
Y.S.

To my sweethearts
Ananya, Diya, Urvi, and Vaanya
A.B.

To my parents Ping-kwai and Hoi-kuen,
and my wife Christine
who have shaped me to what I am.
R.C.

Preface

After a brief lull in the 1990s, robotics has become resurgent thanks to advancements in robotic autonomy made implementable through hyper fast, cheap, multithreaded computing. Interestingly, different parts of the world have made leaps in different research directions, e.g., military robotics has made rapid progress in the US, while assistive robots have been ascendant in Japan. However, the underlying technologies driving the various facets of robotics research are inter-transferable such that these advancements with seemingly different application areas are synergistically contributing to the rapid evolution of robotics as a whole.

Recently, robotic vision as a research area has advanced dramatically,especially with the development of new range sensors. A multitude of techniques have been developedwith impact on areas such as robotic navigation, scene/environment understanding, and visual learning, which are essential to the goal of bringing robots into our dailylives. This edited collection is drawn from a selection of topics that were first presented at the 2013 IEEE Workshop on Robot Vision held at the Sheraton Sand Key in Clearwater, Florida. The material presented in this book is an extended version of ten selected original workshop articles and represents some of the most recent important advancements in the field of robotic vision.

Chapter 1, Multi-modal Manhattan World Structure Estimation for Domestic Robots, by Kai Zhou, Karthik Mahesh Varadarajan, Michael Zillich and Markus Vincze, presents a novel approach based on Jensen-Shannon Divergence to estimate multi-plane structures in a 3D scene without any prior knowledge of the scene.

Chapter 2, RMSD: A 3D Real-time Mid-Level Scene Description System, byKristiyan Georgiev and Rolf Lakaemper, introduces a real-time system that extracts 2D geometric features from 3D range data to form candidate sets of 3D models, which are then used to represent 3D objects.

Chapter 3, Semantic and Spatial Content Fusion for Scene Recognition, by Elahe Farahzadeh and Cham Tat-jen and Wanqing Li, presents a framework that uses not only discriminative features, but also their spatial and semantic relationships to categorize a scene.

Chapter 4, Improving RGB-D Scene Reconstruction Using Rolling Shutter Rectification, by Hannes Ovrn, Per-Erik Forssn, and David Trnqvist, describes a new technique to use a gyroscope to rectify the depth scans sufferingfrom rolling shutter distortions.

Chapter 5, Modeling paired objects and their interaction, by Yu Sun and Yun Lin, presents a human-object-object (HOO) interaction affordance learning approach that models the interaction motions between paired objects in a human-object-object way and uses the motion models to improve the object recognition reliability.

Chapter 6, Probabilistic Active Recognition of MultipleObjects using Hough-based Geometric Matching Features, by Natasha Govender and Jonathan Warrell, proposes an advanced active object recognition approach that recognizes multiple objects simultaneously even with occlusions by incorporating object viewpoint features into Bayesian object models.

Chapter 7, Incremental Light Bundle Adjustment (iLBA): Probabilistic Analysis and Application to Robotic Navigation by Vadim Indelman and Frank Dellaert, presents probabilistic analysis of a structure-less bundle adjustment (BA) method for camera pose estimation which is followed by extension of the iLBA approach to robotic navigation.

Chapter 8, Online Learning of Vision-Based Robot Control during Autonomous Operation, by Kristoffer fjll and Michael Felsberg, presents two different strategies for activation of online learning, namely, (a) an autonomous strategy where the robot switches to a learning mode and generates training data via exploration, and (b) a semi-autonomous strategy in which an operator/supervisor provides training data via manual control as needed.

Chapter 9, 3D Space Automated Aligning Task Performed by A Microassembly System Based on Multi-channel Microscope Vision Systems, by Zhengtao Zhang, De Xu, and Juan Zhang,presents an alignment approach for mm-sized complex micro parts based on information from 3 cameras. Position- and Image-based visual servoing methods are utilized. For speed and accuracy, a coarse-to-fine alignment strategy is proposed along with active zooming.

Chapter 10, Intensity-Difference Based Monocular Visual Odometry for Planetary Rovers, by Geovanni Martinez, proposes a method to estimate the 3D motion of a rover by maximizing the conditional probability of intensity differences at key observation points between successive images captured by a single video camera rigidly attached to the rover.

Yu Sun, Aman Behal, and Chi-Kit Ronald Chung (Editors) would like to express their gratitude to all the authors for their contributions to this collection and thank Springer for their support and assistance in the publishing process.

June, 2014

Yu Sun
Tampa, FL, U.S.A.
Aman Behal
Orland, FL, U.S.A.
Chi-Kit Ronald Chung
Hong Kong, China

Contents

List of Contributors

Tat-Jen Cham
Center for Multimedia and Network Technology,
School of Computer Engineering, Nanyang Technological University,
Singapore 639798
e-mail: astjcham@ntu.edu.sg

Frank Dellaert
Institute for Robotics and Intelligent Machines (IRIM),
Georgia Institute of Technology, Atlanta, Georgia 30332, USA
e-mail: dellaert@cc.gatech.edu

Elahe Farahzadeh
Center of Computational Intelligence, School of Computer Engineering,
Nanyang Technological University, Singapore 639798
e-mail: elah0001@ntu.edu.sg

Michael Felsberg
Department of Electrical Engineering, Linköping University,
SE-581 83 Linköping, Sweden
e-mail: michael.felsberg@liu.se

Per-Erik Forssn
Department of Electrical Engineering, Linköping University,
SE-581 83 Linköping, Sweden
e-mail: per-erik.forssen@liu.se

Kristiyan Georgiev
Temple University, CIS Department, 1800 N Broad St, Philadelphia, 19122, USA
e-mail: georgiev@temple.edu

Natasha Govender
Mobile Intelligent Autonomous Systems, CSIR, South Africa
e-mail: ngovender@csir.co.za

Vadim Indelman
Department of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel
e-mail: vadim.indelman@technion.ac.il

Mogomotsi Keaikitse
Mobile Intelligent Autonomous Systems, CSIR, South Africa
e-mail: mkeaikitse@csir.co.za

Rolf Lakaemper
Temple University, CIS Department, 1800 N Broad St, Philadelphia, 19122, USA
e-mail: lakamper@temple.edu

Wanqing Li
Information and Communication Technology (ICT) Research Institute University of Wollongong, NSW 2522, Australia
e-mail: wanqing@uow.edu.sg

Yun Lin
Computer Science and Engineering, University of South Florida, Tampa, FL, USA
e-mail: yunlin@mail.usf.edu

Geovanni Martinez
Image Processing and Computer Vision Research Laboratory (IPCV-LAB),
School of Electrical Engineering, University of Costa Rica,
2060 San Jos, Costa Rica
e-mail: gmartin@eie.ucr.ac.cr

Fred Nicolls
Department of Electrical Engineering, University of Cape Town, South Africa
e-mail: fred.nicolls@uct.ac.za

Kristoffer Öfjäll
Department of Electrical Engineering, Linköping University,
SE-581 83 Linköping, Sweden
e-mail: kristoffer.ofjall@liu.se

Hannes Ovrèn
Department of Electrical Engineering, Linköping University,
SE-581 83 Linköping, Sweden
e-mail: hannes.ovren@liu.se

Yu Sun
Computer Science and Engineering, University of South Florida, Tampa, FL, USA
e-mail: yusun@cse.usf.edu

Philip Torr
Department of Engineering Science, Oxford University
e-mail: `philip.torr@eng.ox.ac.uk`

Karthik Mahesh Varadarajan
Automation and Control Institute (ACIN), Vienna University of Technology,
Gußhausstraße 25, 1040, Vienna, Austria
e-mail: `varadarajan@acin.tuwien.ac.at`

Markus Vincze
Automation and Control Institute (ACIN), Vienna University of Technology,
Gußhausstraße 25, 1040, Vienna, Austria
e-mail: `vincze@acin.tuwien.ac.at`

Jonathan Warrell
Biosciences, CSIR, South Africa
e-mail: `jwarrell@csir.co.za`

De Xu
Institute of Automation, Chinese Academy of Sciences,
95 Zhongguancun East Road, Beijing, China
e-mail: `de.xu@ia.ac.cn`

Zhengtao Zhang
Institute of Automation, Chinese Academy of Sciences,
95 Zhongguancun East Road, Beijing, China
e-mail: `zhengtao.zhang@ia.ac.cn`

Juan Zhang
Institute of Automation, Chinese Academy of Sciences,
95 Zhongguancun East Road, Beijing, China
e-mail: `juan.zhang@ia.ac.cn`

Michael Zillich
Automation and Control Institute (ACIN), Vienna University of Technology,
Gußhausstraße 25, 1040, Vienna, Austria
e-mail: `zillich@acin.tuwien.ac.at`

Kai Zhou
Automation and Control Institute (ACIN), Vienna University of Technology,
Gußhausstraße 25, 1040, Vienna, Austria
e-mail: `zhou@acin.tuwien.ac.at`

Chapter 1
Multi-modal Manhattan World Structure Estimation for Domestic Robots

Kai Zhou, Karthik Mahesh Varadarajan, Michael Zillich, and Markus Vincze

Abstract. Spatial structure, typically dealt with by robots in domestic environments conform to Manhattan spatial orientations. In other words, much of the 3D point cloud space conform to one of three primal planar orientations. Hence analysis of such planar spatial structures is significant in robotic environments. This process has become a fundamental component in diverse robot vision systems since the introduction of low-cost RGB-D cameras such as the Kinect, ASUS and the Primesense that have been widely mounted on various indoor robots. These structured light/ time-of-flight commercial depth cameras are capable of providing high quality 3D reconstruction in real-time. There are a number of techniques that can be applied to determination of multi-plane structure in 3D scenes. Most of these techniques require prior knowledge modality of the planes or inlier scale of the data points in order to successfully discriminate between different planar structures. In this paper, we present a novel approach towards estimation of multi-plane structures without prior knowledge, based on Jensen-Shannon Divergence (JSD), which is a similarity measurement method used to represent pairwise relationship between data. Our model based on the JSD incorporates information about whether pairwise relationships exist in a model's inlier data set or not as well as the pairwise geometrical relationship between data points.

Tests on datasets comprised of noisy inliers and a large percentage of outliers demonstrate that the proposed solution can efficiently estimate multiple models without prior information. Experimental results shown using our model also demonstrate successful discrimination of multiple planar structures in both real and synthetic scenes. Pragmatic tests with a robot vision system also demonstrate the validity of the proposed approach. Furthermore, it is shown that our model is not just restricted to linear kernel models such as planes but also be used to fit data using non-linear kernel models.

Kai Zhou · Karthik Mahesh Varadarajan · Michael Zillich · Markus Vincze
Automation and Control Institute, Vienna University of Technology, A-1040, Vienna, Austria
e-mail: {zhou,kv,zillich,vincze}@acin.tuwien.ac.at

Y. Sun et al. (eds.), *New Development in Robot Vision*,
Cognitive Systems Monographs 23, DOI: 10.1007/978-3-662-43859-6_1

1.1 Introduction

Robots in domestic environments have to deal with a variety of structured as well as unstructured scene types. The algorithms required to handle this uncertainty in scene information are usually embedded in intelligent decision schema. Nevertheless, depending on the level of uncertainty in the scene content, the required complexity of the decision system can be quite high. The knowledge accumulation necessary for this purpose can be done through data prior models as well as incremental knowledge accumulation. The trade-off between the level of intelligence embedded in the form of data prior models by robot designers and that obtained by autonomous operation by interactively analyzing the information that the robot percepts, is a big challenge in the design of modern service robots. Therefore, there is a large literature of algorithms on designing efficient, orderly and polished rule priors to summarize robotic knowledge as comprehensive as possible.

Fortunately, the real world in which robots perform tasks is not in a state of complete disorder or chaos that can prevent any form of modeling. In 1999, Coughlan and Yuille published a landmark paper verifying a general observation that most indoor and outdoor (city) scenes are designed on a Manhattan three-dimensional grid [4]. Although they originally design the "Manhattan world" with statistical regularities for the sake of inferring the viewer orientation relative to the Manhattan grid and detecting targets unaligned to the grid, Manhattan constraints have turned out to be an important prior in the systematic design of most indoor robot vision systems[11][5]. Spatial knowledge demonstrates how the intrinsic orderliness of the real world can be used to help robots perform tasks more effectively.

Manhattan worlds can essentially be decomposed into a set of planar structures oriented along three primal directions. Hence, domestic robots working in indoor environments typically need to deal with regular structure typically in the form of planes. Analysis of such structure using perception systems on the domestic robots is crucial to performing a number of tasks such as environment mapping, localization, navigation, obstacle detection, occupancy grid estimation, salient region or outlier estimation, object recognition, scene understanding, spatial scaffolding, semantic scene processing etc. Some of the scenarios are described in fig. 1.1. In fig. 1.1a, the visual perception system on the robot has to determine the salient regions in the scene. Outliers to the planar structure as described by the table in the scene form the primal components for salient or attention regions. In this case, the keyboard on the table forms one such outlier. In fig. 1.1b, the visual perception system on the robot is required to determine the multiple planes of the multi-level shelf structure. This is necessary in order to estimate the perform spatial scaffolding and semantic analysis of the scene to determine the relative locations and pose of various interesting objects for further processing such as object recognition, classification followed by manipulation and other forms of robotic interaction. In fig. 1.1c, the vision system is required to identify walls, cupboards and the flooring which again can be modeled using multiple planes. This helps a robot in mapping its environment and determining optimal navigation strategies through path planning as well as in localization. In fig. 1.1d, the perception shown has to deal with a similar scenario with added constraints rendered by obstacles in the path of the robot. Again structural modeling of the scene using a Block-World paradigm and estimation of outliers helps in obstacle avoidance and successful robot navigation. This process also involves occupancy grid determination. Fig. 1.1e shows that robot navigation can be further hindered by spurious scene structure such as staircases and stairwells. Again multi-modal plane estimation comes in handy for determining such structure resulting in appropriate course of action, which in the case of a wheeled robot is obstacle avoidance and in the case of a biped robot involves suitable motion schema to ascend or descend the stairs as appropriate. Robots might also have to deal with other objects of interest mounted on vertical planar structures such as walls and cupboards. For example, a robot equipped with an arm and hand can be used to interact with the handle of doors, cupboards and windows for navigation and other tasks. This test case scenario also involves operating circuit breaker switches and other fixtures mounted on planar structures as shown in fig. 1.1f. In all such cases, determination of the planar structures in the scene is a key component of the embodied visual perception algorithms. It should be noted that typical scenes do not just have one planar structure, but are composed of multiple planes in different orientations. Furthermore, 3D scene points corresponding to the multi-planar structure typically occupy a large area of the visual scene to be processed. Hence, it is necessary to design efficient approaches towards regularized and structured 3D scene processing which can handle a large number of data points while being capable of determining multi-modal scene content.

As one of the most common spatial structures that exist in the man-made world, planar surfaces have been widely studied and extracted for robots to analyze and understand their surrounding environments. Conventional approaches such as the RANSAC [6] family show limitations when dealing with data containing multiple models, high percentage of outliers or sample selection bias, commonly encountered in robot vision applications. Therefore, applying the RANSAC family in a robot vision system requires the use of many unstable, non-adaptive and hard-tuned parameters, such as numbers of planes and inliers scale that changes with the distance from the camera to the plane.

In this paper, we propose a novel multiple plane estimation algorithm, which does not require any prior parameters, and employ it on our robotic vision system to autonomously detect all planar structures in a variety of scenes. The key technique in the proposed algorithm is to select the inliers from the data set directly through the analysis of the pairwise relations between data points using Jensen-Shannon Divergence (JSD) [7], which is frequently used for similarity measurement in various contexts. Fig. 1.3 summarizes the proposed mechanism with synthetic data. The input data contains 100 points per line to modeled from the data points (All the synthetic data for 2D line fitting tests in this paper are corrupted with Gaussian noise with a standard deviation of $\sigma = 0.01$) and with 600 gross outliers (outlier rate of 90%). Colors ranging from red to blue, while cycling through orange, yellow and cyan, illustrate the order of data point selection.

The paper is organized as follows. In section 1.2, we present the background and review the state-of-the-art in multi-model estimation algorithms as well we their

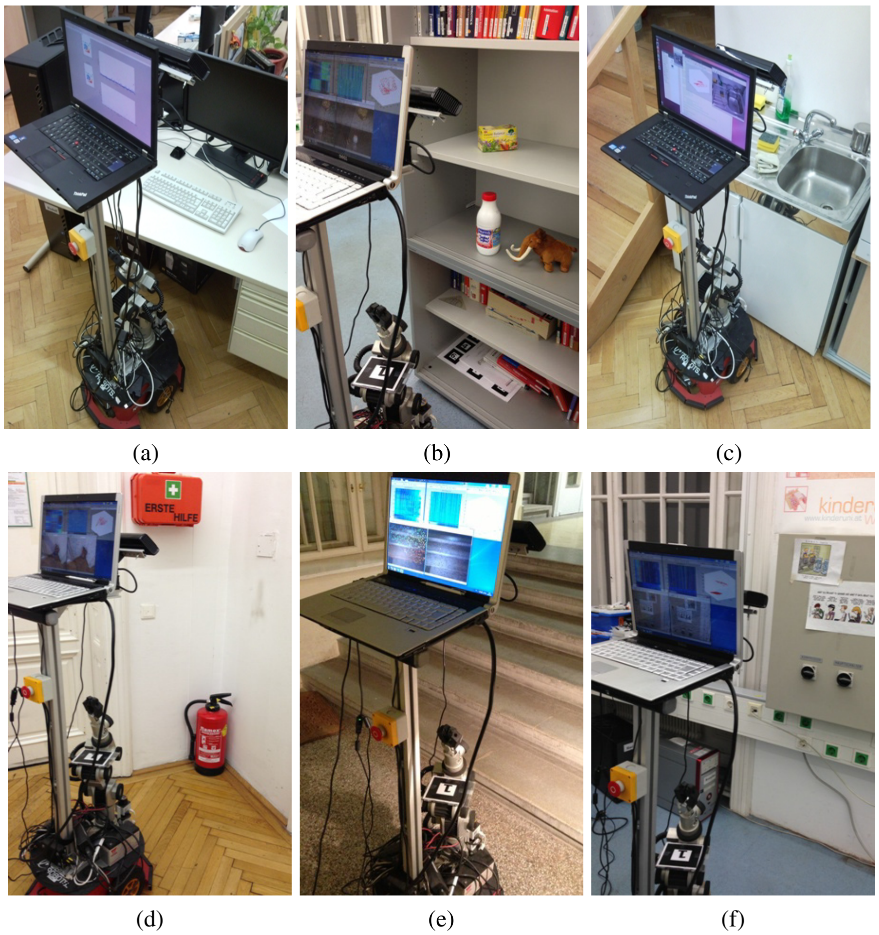


Fig. 1.1 Variegated test scenes to be handled by the perception systems in typical domestic robots.(a) Task salient objects on a desk (b) Salient objects on a shelf (c) Navigation with walls, floor and cupboards as bounds (d) Navigation with obstacles (e) Navigation constrained by a staircase (f) Handling of circuit breakers.

application to robot vision solutions. Section 1.3 gives the preliminary definition and data structure used for the proposed algorithm. In section 1.4, we describe the inlier selection mechanism used for filtering the gross outliers. The details of utilization of the selected inliers to form the final estimated planes, as well as the experimental setting with our mobile robot in an indoor environment, are outlined in section 1.5. Subsequent sections present experimental results and evaluation. Conclusions are given at the end of the paper and the future work is shortly discussed as well.

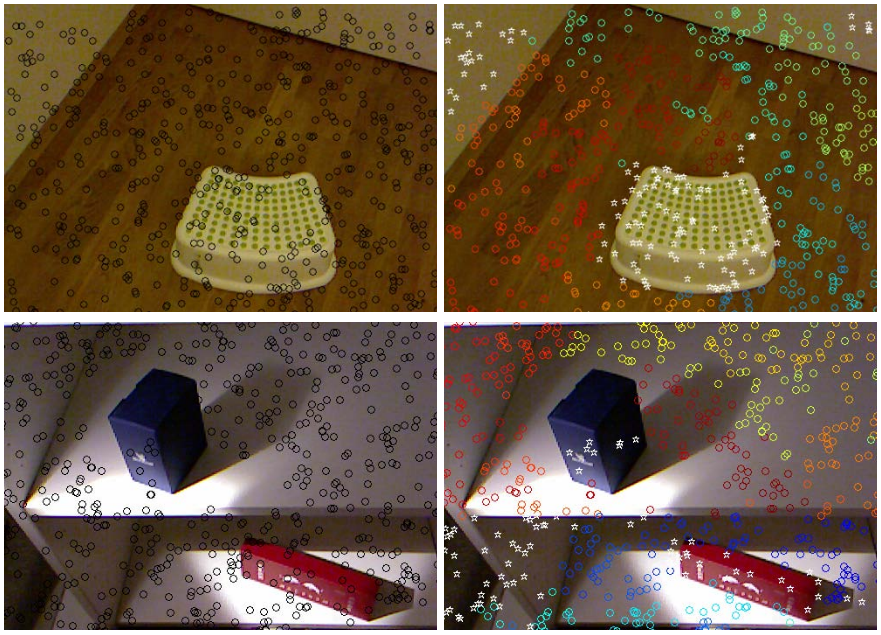


Fig. 1.2 The first row shows a typical scene in a domestic environment - object on a single plane. The second row shows a more challenge scene where robot needs to deal with the complex multiple planar structures. Note that the circular markers in the images on the left show locations of back-projected down-sampled 3D points. On the right, the markers are color coded to indicate the sequence in which they are added to the kernel model, while points corresponding to objects or outliers to the model are encoded as white pentagrams.

1.2 Related Work

The first sub-section deals with a review of recent developments in multiple planes estimation. The next sub-section discusses applications of planar structure estimation in robotic vision systems.

1.2.1 Multi-modal Plane Estimation

The traditional RANSAC algorithm has been designed to handle only a single mode. In other words, for the case of plane estimation, only a single planar can be efficiently extracted using RANSAC from data points corrupted by noise. However, there exist several extensions of RANSAC that are capable of tackling the problem of multi-modal plane estimation, such as [15][22], but these algorithms require the number of models to be specified by the user beforehand and therefore are less robust in real environments. These RANSAC based algorithms, which focus on *hypothesis evaluation*, generate hypothetical models, then the selection of the best

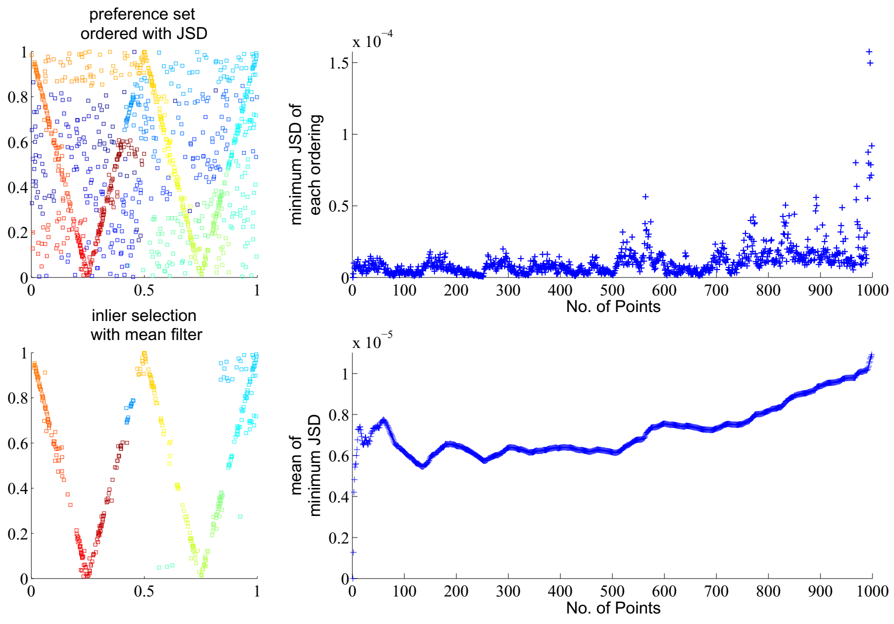


Fig. 1.3 2D line fitting of 4 lines using the proposed method. Left images show inlier selection process and result. Right images demonstrate the JSD between two continuous data points as well as the accumulative mean values of the JSD.

hypothesis is solved as a verification process which calculates every hypothesis's residual distribution (the normalized histogram that is generated with the distances from all the data points to this hypothesis).

A change of perspective to multi-modal estimation has been presented in [20], where they researched on the residual distribution for each point instead of studying the residual distribution per hypothesis since the modes (e.g. numbers of the peaks) in the former distribution reflect the numbers of the models - or the modality. This method concentrates on the *inlier selection*, i.e. clustering/labeling the data points to obtain the inlier sets of models directly instead of evaluating hypotheses, and iteratively discovers the number of models through analysis of data w.r.t the each chosen model. Since the direct analysis of data points is not plausible in practice (a large number of hypotheses are required for valid modeling), the relationship among data elements are investigated for identifying inliers indirectly.

Recently, more generic conceptual representations of relationship among data, have been proposed for solving the multi-modal estimation task by fusing the studies in both hypothesis evaluation and data selection [19][16][2][17][18][8]. The preference matrix [16], Mercer kernel matrix [2][19] and energy functions [8] have been utilized to provide global representations of relationship between data points. However, all of these methods either need a user to specify the inlier scale [16][8], or introduce new user-dependent parameters (step h in computing the ordered residual kernel [2], weighting factors α, β in the objective function to be optimized [19]). Similar parameters are found in other approaches [17][18]. These parameters are required in order to estimate the multiple models using the relationship between data points, by means of a series of procedures. The scale estimation substep used in the algorithm focuses on scoring the hypotheses to select good hypotheses followed by the model fitting substep which identifies inliers using the estimated inlier scale. The sequential processing results in inter-dependence in computation and results, leading to reduced robustness and flexibility of the algorithms.

1.2.2 Multi-modal Planar Modeling for Robotics

Domestic robots are, in general, expected to perform a variety of perception, grasping and manipulation tasks. In general, the design of the robot vision system varies with respect to the various tasks that the robot needs to perform. Recently, researchers have introduced 3D cues into the robot vision system as a fundamental information channel since low-cost RGB-D cameras, which provide real time 3D reconstruction, have been commonplace. The mechanisms for analysis of 3D spatial structures have also been updated from that used for traditional static scenes (e.g. detecting the dominant plane in the scene for enabling the robot to grasp the objects which pop out from the planar surface [12]) to challenging dynamic scenes (e.g. searching multiple objects on various supporting surfaces [14][1] or to learn the properties of the objects with respect to the dominant plane [21]). However, these robotic vision systems are all built upon the RANSAC family to detect the planar structure, therefore only one dominant plane in the scene can be detected and only the objects on this plane can be found by the robots. This limitation of RANSAC can be eliminated by the sequential use of RANSAC (however, the number of times it can be used is still an open issue) or other multi-modal robust regression algorithms (which decide the number of the models as an internal parameter).

1.3 Relationship between Pairwise Data

This section describes the theory of our approach. The modeling of the relationship between pairwise data points using JSD is then discussed.

Relationship between pairwise data points have been used recently in several algorithms to cluster/label data points since, in practice, it is almost impossible to find a representation function that can be used to evaluate the data points directly. In [16], the Jaccard distance is used to represent the relations between pairwise data, these pairwise relations are then utilized as linkage measurement for a connectivity based clustering algorithm. Similarly, [2] uses the ordered residual kernel to measure the similarities between pairwise data points. Their tailored kernel satisfies Mercer condition, thus it can be considered as the inner product of two data points. The kernel trick can be built upon it for use with statistic learning methods. However, when these two methods construct the relationship between data points, either the

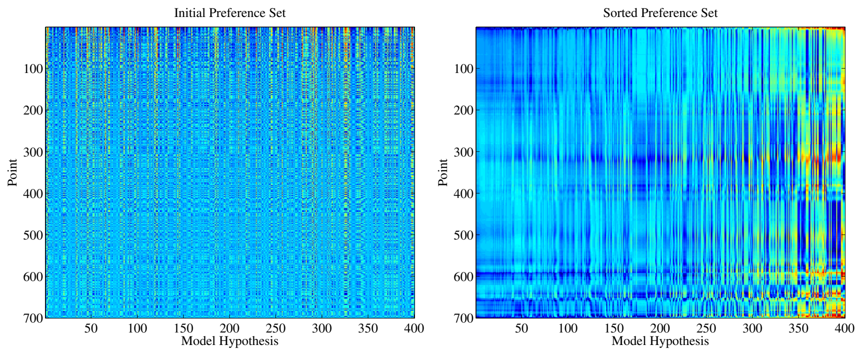


Fig. 1.4 The generalized distance matrix generated for a sample synthetic linear plane fitting with gaussian noise, before (left) and after (right) point sort

inlier noise scale [16] or the step size h for generating the difference of intersection between residuals [2] is required and this is crucial to the performance of the algorithm. On the contrary, the proposed JSD-based method **does not require any user input parameters**.

1.3.1 Generalized Distance Matrix

Given the input data $\mathscr{L} = \{x_i\}_{i=1,\dots,N}$ and model hypotheses $\mathscr{H} = \{\theta_j\}_{j=1,\dots,M}$ which are generated by randomly sampling from $\mathscr{L}$, we calculate the distance vector d_i of all the hypotheses to each data point x_i in $\mathscr{L}$, $d_i = \{d_1^i, \dots, d_M^i\}$, where d_m^i is the distance of point x_i to hypothesis θ_m. Note in particular that we do not truncate the distances therefore no inlier scale specification or estimation is required.

The distance matrix $\mathscr{D}_{N\times M} = \{d_1; d_2; \dots; d_N\}$ is a N-by-M data matrix, each row is a distance vector d. For each row, we calculate the sum of the distances as,

$$s_i = \sum_{j=1}^{M} d_j^i \tag{1.1}$$

Then sorting the rows according to the s_i to generate the sorted distance matrix $\tilde{\mathscr{D}} = \{d_{\lambda_1}; d_{\lambda_2}; \dots; d_{\lambda_N}\}$, where the permutation $\{\lambda_1, \lambda_2, \dots, \lambda_N\}$ is obtained such that $s_{\lambda_1} \leq s_{\lambda_2} \leq \dots \leq s_{\lambda_N}$. Defining the sorted normalization vector $S = \{1/s_{\lambda_1}, 1/s_{\lambda_2},$
$\dots, 1/s_{\lambda_N}\}$ we can formulate the generalized distance matrix (GDM) G as

$$G = \tilde{\mathscr{D}} \cdot S = \{g_1; \dots; g_N\} = \{\frac{d_{\lambda_1}}{s_{\lambda_1}}; \dots; \frac{d_{\lambda_N}}{s_{\lambda_N}}\} \tag{1.2}$$

1.3.2 Jensen-Shannon Divergence (JSD)

Jensen-Shannon divergence, which is based on Kullback-Leibler divergence (KLD) [9], is widely used in probability theory and statistics for measuring the similarity between two distributions. The superiority of JSD over KLD in handling the zero values in distributions and elimination of constraints in the use of KLD arising from its asymmetry has been demonstrated in [10].

Since the row vectors in the generalized distance matrix G have already been normalized (sum to 1), the Jensen-Shannon divergence (JSD) between g_i and g_j can be directly calculated by

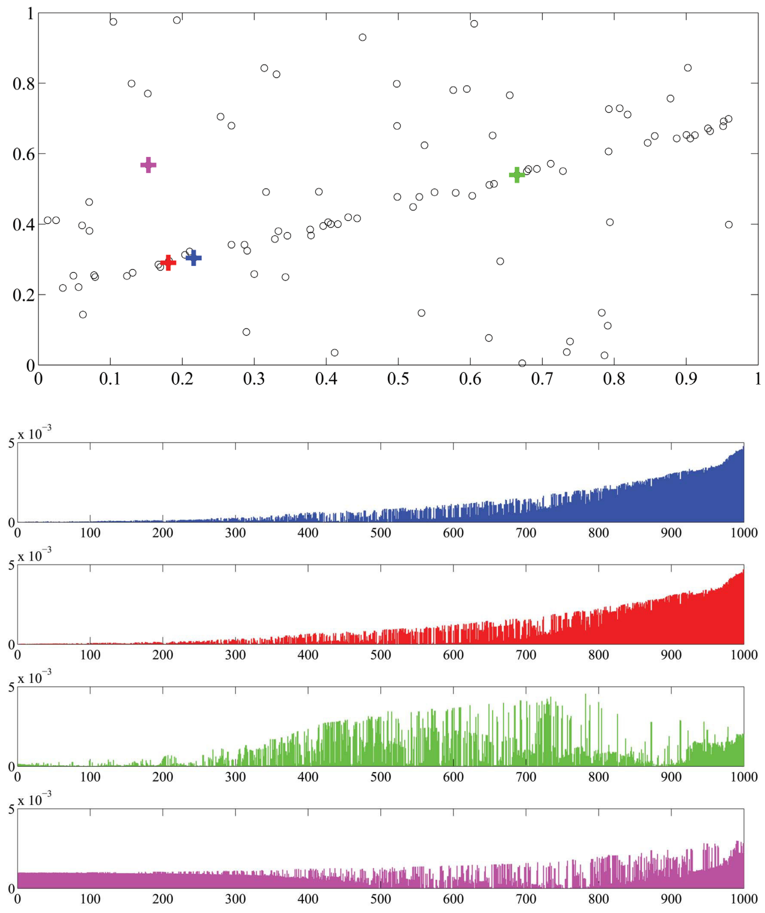


Fig. 1.5 The original data points (top) and the illustration of distance vector examples (bottom)

$$\begin{aligned} JSD(i,j) &= \frac{1}{2}KLD(g_i||m) + \frac{1}{2}KLD(g_j||m) \\ KLD(g_i||m) &= \sum_{k=1}^{M} r_i^k g_i^k \log \frac{g_i^k}{m^k} \end{aligned} \tag{1.3}$$

where $m = \frac{1}{2}(g_i + g_j)$. We utilize a Gaussian-weighted KLD (GKLD with Gaussian kernel r^k) to reinforce the influence of JSD for data points close to the hypothesis, and to weaken the effect caused by the parallel model. The JSD values between data points in Fig. 1.5 are 3.86×10^{-7} (blue-red points), 6.99×10^{-6} (blue-green points) and 1.17×10^{-4} (blue-magenta points). The difference in JSD between the inlier-inlier pairs (blue-red and blue-green pairs) and inlier-outlier pair (blue-magenta) are significant enough to distinguish the outliers while inliers that are close to each other produce relatively small JSD. $C = \{c_1, c_2, \ldots, c_M\}$ is a discrete kernel function for weighting hypotheses. The weights can be computed using any hypothesis evaluation methods such as [3] or simply adopt discrete Gaussian kernel to the sorted hypotheses, i.e.

$$c_i = \frac{2}{\sigma\sqrt{2\pi}} e^{-\frac{i^2}{2\sigma^2}} \tag{1.4}$$

where $\sigma = M/4$ makes the sum of C approximate to 1. The sorting of hypotheses is performed at every step of data point selection and this is elaborated in section 1.4.

1.4 Modeling and Selection of Inliers

The JSD value measures the pairwise similarity between data points in the generalized distance matrix, and it encodes the following information about the relationship between these pairwise data points, 1) Data points that are in the same geometric neighborhood will generate small JSD. 2) Two data points belonging to the same 'true' model will produce relatively smaller JSD than that between one inlier and one outlier. 3) Significantly large JSD will be generated by an outlier-outlier pair.

Considering these three effects of the proposed JSD-based measurement method, we design an inlier discovery algorithm which attempts to identify the inliers. The algorithm sorts the points such that every point p_i is followed by the point which has the minimum JSD $minJSD(i)$ in the set $\{JSD(i,i+1), JSD(i,i+2), \ldots, JSD(i,N)\}$. Algorithm 1 illustrates the inlier discovery scheme.

If the initial point is the endpoint (inlier that exists at the border of the structure) of the 'true' model, the search scheme could generate the perfect order of the inliers and outliers. However, this condition cannot be satisfied in practice since this prior information may not available or in multi-model case, the endpoint of one model may not necessarily be the start point of another one. Therefore, various alternating inlier and outlier clusters will be generated using this search scheme (In fig. 1.3 color varies from red (inliers) to yellow (inliers of another model) through orange (outliers)). Since the JSD between outliers are significantly larger than the JSD between inliers, we can monitor the accumulative mean value of *minJSD* to detect the outliers. The accumulative mean value of *minJSD* is defined as,

$$meanMin(i) = \frac{\sum_{j=1}^{i} minJSD(j)}{i} \tag{1.5}$$

Within the selected inliers, any robust fitting method such as LMedS [13] can be utilized to estimate the parameters of the structures since most of the gross outliers are omitted from the dataset.

1.5 Experiments

The JSD-based inlier discovery algorithm is first tested using synthetic data for both single model and multi-model (linear or non-linear) estimation, and then we demonstrate the application of our method to estimate multiple planes in various scenarios in which the robot needs to perform scene exploration tasks. All results for the synthetic data processing were obtained on an Intel Quad Core 2.4 GHz CPU, 4 GB RAM machine and the plane estimation tasks were performed on an Intel Mobile Core i7 2720QM CPU, 8 GB RAM laptop connected to a Microsoft Kinect.

Fig. 1.6 depicts the inlier selection processes and results. All the test cases contained 100 inliers per line/circle and 400 outliers (the outlier rates these cases are 80%, 83.3%, 88.9% and 85.7%, respectively). The distance of a data point to circle model is defined as the distance between the data point and the point on the circle closest to the data point along the radial direction. These distances have been normalized from 0 to 1 and then been input into a generated distance matrix (GDM) which is displayed in fig. 1.4.

Algorithm 1. Inlier discover scheme

```
1:  input: GDM $G = \{g_1; \ldots; g_N\} = \{g^1, \ldots, g^M\}$
2:  while Point index $i < N$ do
3:      $\forall g_j, j \in (i+1, N]$, calculate $JSD(i,j)$.
4:      find $j_{min}$ make $JSD(i, j_{min}) \leq \forall JSD(i,j)$.
5:      store $JSD(i, j_{min})$ into $minJSD(i)$
6:      swap $g_{i+1}$ and $g_{j_{min}}$.
7:      sort $G$ as $\{g^{\mu_1}, g^{\mu_2}, \ldots, g^{\mu_M}\}$ that satisfies $g^{\mu_1}_{i+1} \leq g^{\mu_2}_{i+1} \leq, \ldots, \leq g^{\mu_M}_{i+1}$
8:      $i = i + 1$
9:  end while
10: for $l = 2, \ldots, N$ do
11:     calculate $meanMin(l)$ using eq. 1.5
12:     if $meanMin(l-1) \geq meanMin(l)$ then
13:         label point $l$ as inlier
14:     end if
15: end for
```

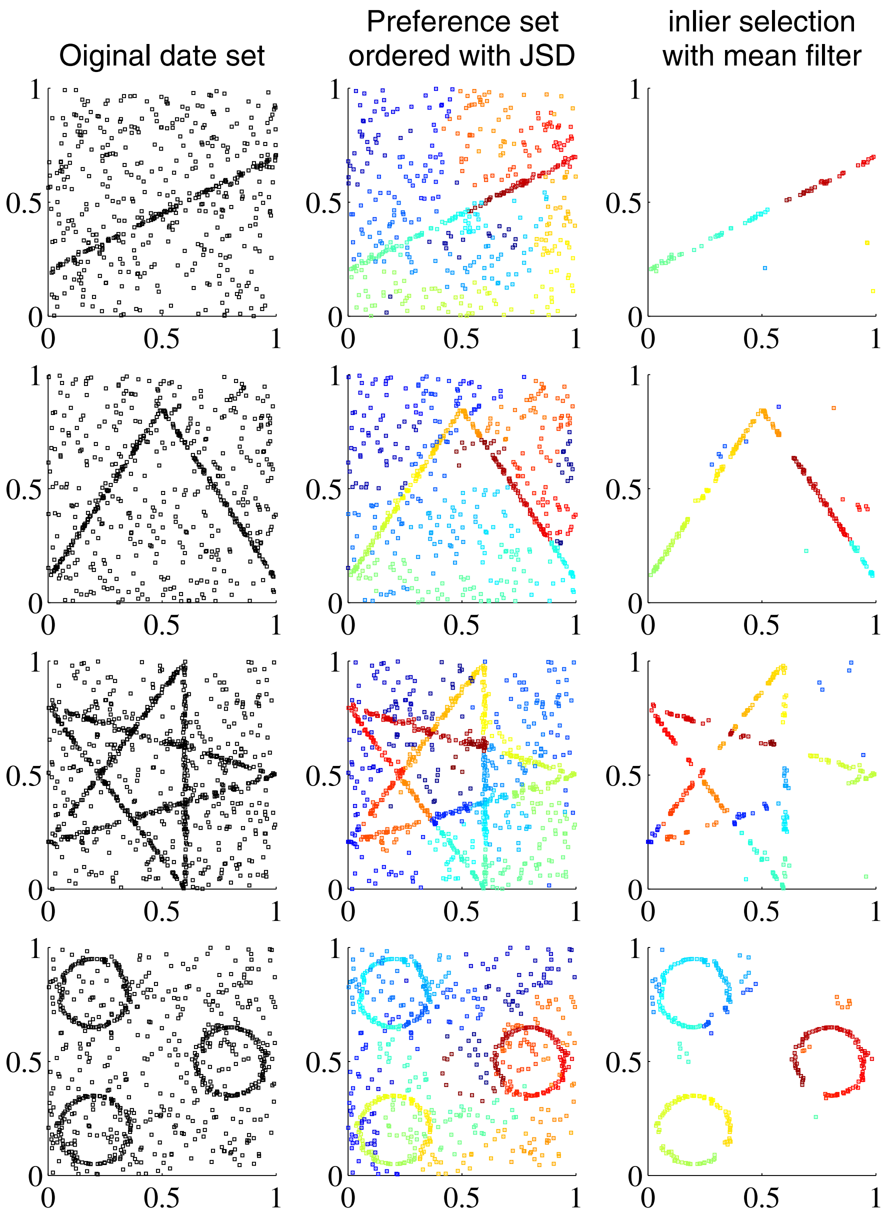


Fig. 1.6 Linear and non-linear fitting results - lines and circles. Results in each row demonstrate fitting for piecewise linear shapes and circles, namely, an inclined line, inverted V (roof), a pentacle and multiple circles. Color ranges from red to blue, passes through orange, yellow, green and cyan, illustrates the order of data point selection. Each test data set contains 100 inliers per line/circle and 400 gross outliers.

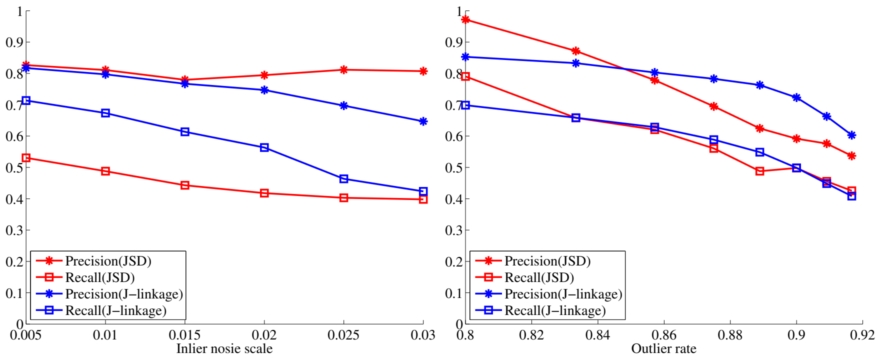


Fig. 1.7 Precision and recall of the test case in fig. 1.3, i.e. 4 lines with 100 inliers per line. 200 gross outliers are added for the test case shown in the image on the left, inliers are corrupted with Gaussian noise with standard deviation $\sigma = 0.01$ for the image on the right). Comparation with the J-linkage method [16] is also shown in the figure. Note that we use the default set of J-linkage method, i.e. 25 inliers for clustering a model, inlier noise scale is 0.01.

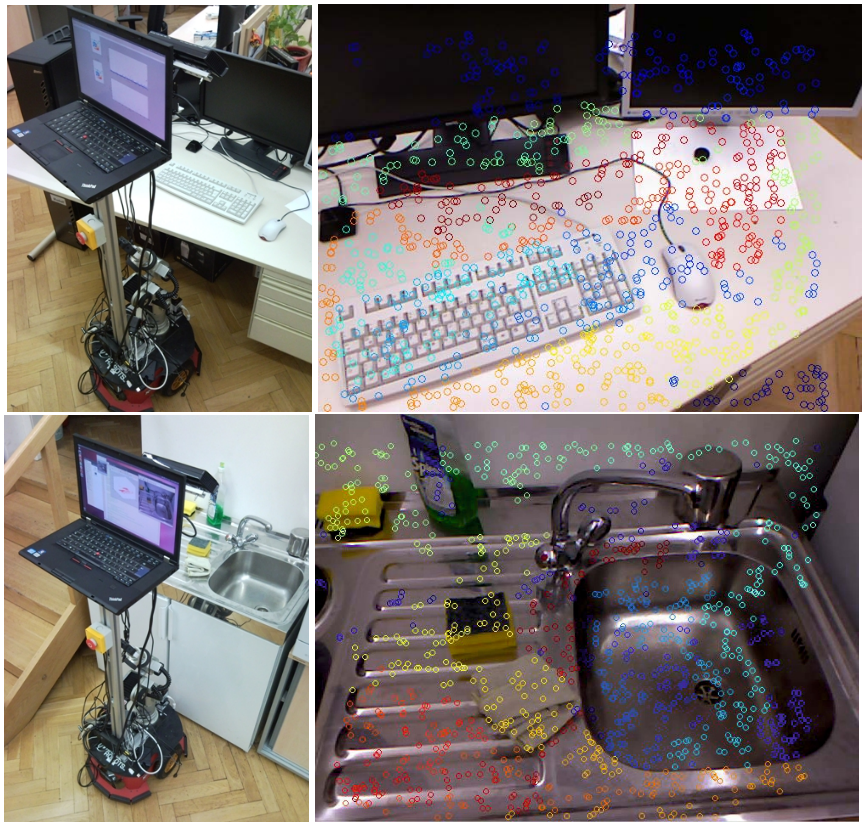
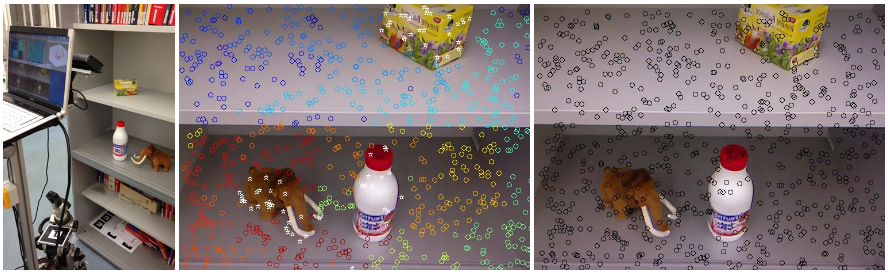
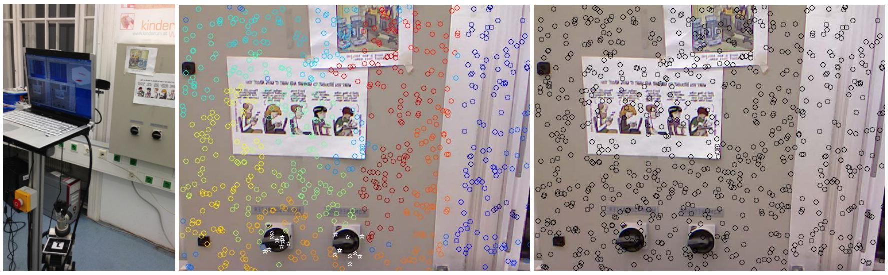


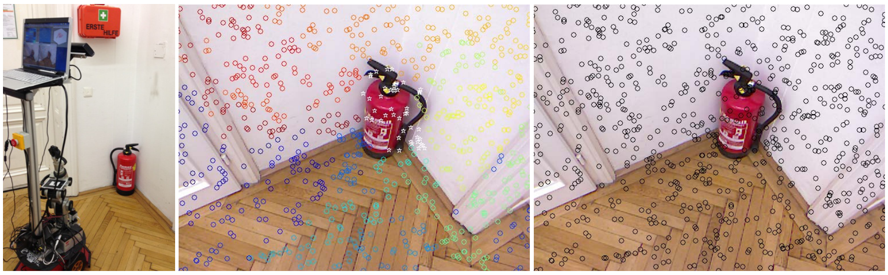
Fig. 1.8 Algorithm is applied for a mobile robot in two scenes, i.e. a typical office table (top) and a water tank in the kitchen (bottom)



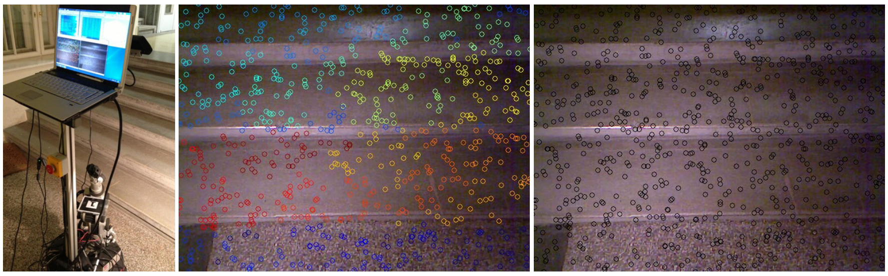
(a) Multi-layer shelf scenario



(b) Circuit breaker panel on the wall



(c) Fire extinguisher at the corner



(d) Staircase scenario

Fig. 1.9 A mobile robot performing multiple planes fitting and object detection tasks when faced with four typical indoor scenarios. Demonstration of detection of (from top to bottom) (a) multi-level shelf (b) circuit breaker panel on the wall (c) fire extinguisher on the floor and (d) staircase

Figure 1.7 illustrates the precision and recall curves generated using various inlier and outlier rates. The chart on the left in fig. 1.7 shows that the proposed method can provide an inlier selection accuracy which is similar or better than the J-linkage method when the outlier rate is not too high - for instance, at an outlier rate of 83.3% as shown in the chart on the left, while in most practical scenarios, the outlier rate is much smaller than this value. The advantage of the proposed method is that it doesn't requires any information about inlier noise scale which usually varies with the observation distances. Although the recall of the proposed method is worse than J-linkage method, i.e. too many inliers belonging to the model have been removed, the model still can be recovered with the inliers left (see pentacle and circles fitting result in Fig. 1.6).

Fig. 1.8 illustrates an indoor mobile robot detecting various planar surfaces in two typical scenes using the proposed algorithm. The back-projected points on the images demonstrate the process of the point selection with color labels. From the figure, we can observe that the multiple planar surfaces in the scenes are successfully distinguished, even the table surface and keyboard surface, which have very small distance between each other are identified independently. For the sake of computational efficiency, 3D point clouds in all the tests in fig 1.9 and 1.8 have been down-sampled to 1000 points.

Fig. 1.9 displays the plane estimation results on RGB-D point cloud data. All point clouds were down-sampled to 1000 points and the orders of the planes being searched are demonstrated using color labels. Animations that show the line/plane estimation processes using the proposed method can be seen at our website[1].

1.6 Conclusion

We have presented a new method for multiple planar structure estimation, which is one of the fundamental components in all robot vision system. We use the Jensen-Shannon Divergence (JSD) to represent the pairwise relations between data points and order the data points according to the calculated JSD, followed by an inlier discovery scheme that monitors the change of the JSD value for filtering the outliers. Our experimental results on a large number of synthetic data clearly demonstrate how the proposed algorithm works and its performance. Also, several planar structure estimation scenarios using an indoor mobile robot mounted with RGB-D camera validate that the proposed approach could provide superior performance in real robotic tasks.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No.215181, CogX, No. 600623, STRANDS, No. 610532, SQUIRREL, and by the Austrian Science Foundation under grant agreement TRP 139-N23 InSitu.

[1] `http://users.acin.tuwien.ac.at/kzhou/?site=3`

References

1. Aydemir, A., Sjöö, K., Folkesson, J., Pronobis, A., Jensfelt, P.: Search in the real world: Active visual object search based on spatial relations. In: Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA 2011), Shanghai, China (2011)
2. Jun Chin, T., Wang, H., Suter, D.: Robust fitting of multiple structures: The statistical learning approach. In: IEEE International Conference on Computer Vision (2009)
3. Chin, T.-J., Yu, J., Suter, D.: Accelerated hypothesis generation for multi-structure robust fitting. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 533–546. Springer, Heidelberg (2010)
4. Coughlan, J.M., Yuille, A.L.: Manhattan world: Compass direction from a single image by bayesian inference. In: ICCV, pp. 941–947 (1999)
5. Delage, E., Lee, H., Ng, A.: Automatic single-image 3d reconstructions of indoor manhattan world scenes. In: Thrun, S., Brooks, R., Durrant-Whyte, H. (eds.) Robotics Research. STAR, vol. 28, pp. 305–321. Springer, Heidelberg (2007)
6. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24(6), 381–395 (1981), doi:http://doi.acm.org/10.1145/358669.358692
7. Fuglede, B., Topsoe, F.: Jensen-Shannon divergence and Hilbert space embedding. In: IEEE International Symposium on Information Theory (2004)
8. Isack, H.N., Boykov, Y.: Energy-based geometric multi-model fitting. IJCV 97(2), 123–147 (2012)
9. Kullback, S., Leibler, R.A.: On information and sufficiency. Annals of Mathematical Statistics 22(1), 79–86 (1951)
10. Lee, L.: On the effectiveness of the skew divergence for statistical language analysis. In: Artificial Intelligence and Statistics, pp. 65–72 (2001)
11. Omedes, J., López-Nicolás, G., Guerrero, J.J.: Omnidirectional vision for indoor spatial layout recovery. In: Lee, S., Yoon, K.-J., Lee, J. (eds.) Frontiers of Intelligent Auton. Syst. SCI, vol. 466, pp. 95–104. Springer, Heidelberg (2013)
12. Ridge, B., Skocaj, D., Leonardis, A.: Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In: ICRA 2010, pp. 5047–5054 (2010)
13. Rousseeuw, P.J., Leroy, A.M.: Robust regression and outlier detection. John Wiley (1987)
14. Sjöö, K., Aydemir, A., Mörwald, T., Zhou, K., Jensfelt, P.: Mechanical support as a spatial abstraction for mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan (2010)
15. Stewart, C.V.: Bias in robust estimation caused by discontinuities and multiple structures. IEEE Transactions on PAMI 19, 818–833 (1997), doi:http://doi.ieeecomputersociety.org/10.1109/34.608280
16. Toldo, R., Fusiello, A.: Robust multiple structures estimation with j-linkage. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 537–547. Springer, Heidelberg (2008)
17. Wang, H., Chin, T.J., Suter, D.: Simultaneously fitting and segmenting multiple-structure data with outliers. IEEE Transactions on PAMI (2012)
18. Wong, H.S., Chin, T.J., Yu, J., Suter, D.: Dynamic and hierarchical multi-structure geometric model fitting. In: ICCV (2011)

19. Yu, J., Chin, T.J., Suter, D.: A global optimization approach to robust multi-model fitting. In: CVPR, pp. 2041–2048 (2011)
20. Zhang, W., Kŏsecká, J.: Nonparametric estimation of multiple structures with outliers. In: Vidal, R., Heyden, A., Ma, Y. (eds.) WDV 2005/2006. LNCS, vol. 4358, pp. 60–74. Springer, Heidelberg (2007)
21. Zhou, K., Richtsfeld, A., Zillich, M., Vincze, M., Vrečko, A., Skočaj, D.: Visual information abstraction for interactive robot learning. In: The 15th International Conference on Advanced Robotics (ICAR 2011), Tallinn, Estonia (2011)
22. Zuliani, M., Kenney, C.S., Manjunath, B.S.: The multiransac algorithm and its application to detect planar homographies. In: IEEE International Conference on Image Processing (2005)

Chapter 2
RMSD: A 3D Real-Time Mid-level Scene Description System

Kristiyan Georgiev and Rolf Lakaemper

Abstract. This paper introduces a system for real-time, visual 3D scene description. A scene is described by planar patches and conical objects (cylinders, cones and spheres). The system makes use of sensor's natural point order, dimensionality reduction and fast incremental model update (in $O(1)$) to first build 2D geometric features. These features approximate the original data and form candidate sets of possible 3D object models. The candidate sets are used by a region growing algorithm to extract all targeted 3D objects. This two step (raw data to 2D features to 3D objects) approach is able to process 30 frames per second on Kinect depth data, which allows for real-time tracking and feature based robot mapping based on 3D range data.

2.1 Introduction

In recent years, most range-sensing based algorithms in robotics, e.g. SLAM and object recognition, were based on a low-level representation, using raw data points. The drawbacks of such a representation (high amount of data, low geometric information) limits the scalability when processing of 3D data is required. The goal of our work is to produce a fast visual scene-description system that uses 3D range data and works in real time. A major difficulty in producing such a system comes from the massive amount of data (ex. MS Kinect has 307200 points at 30 fps). We solve this problem with a two step approach: first, the raw point set is approximated by 2D geometric primitives, which are then combined to form the final 3D primitives, see Figure 2.1. This approach has the advantage that the raw points are only processed once (in an $O(N)$ procedure to detect 2D features), then, the final detection of 3D objects is performed on a significantly smaller set of 2D objects. Due to the large amount of points, computing the 2D primitives has to be fast. We are using an $O(1)$ model update (independent of number of points already part of the model).

Kristiyan Georgiev · Rolf Lakaemper
Temple University, CIS Department, 1800 N Broad St, Philadelphia, USA
e-mail: {georgiev,lakamper}@temple.edu

Y. Sun et al. (eds.), *New Development in Robot Vision*,
Cognitive Systems Monographs 23, DOI: 10.1007/978-3-662-43859-6_2

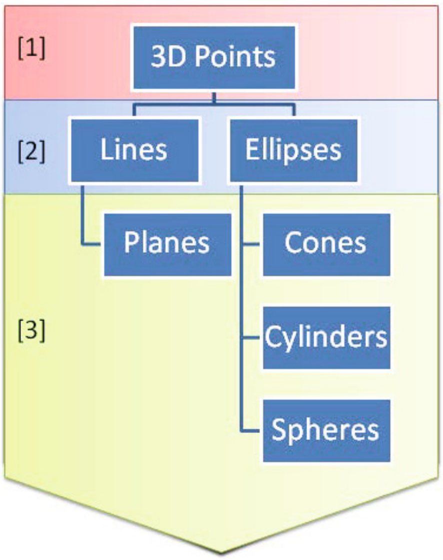


Fig. 2.1 RMSD system flow diagram. 1) Input; 2) Mid-Level features generated from 1); 3) Scene description using features from 2).

Another key feature of our approach is the use of the points' natural order, as defined by the sensor, to traverse and perform model updates resulting in a total complexity of $O(N)$ for N points. Previous approaches in the attempt to perform scene description need to introduce a nearest neighborhood structure, such as k-d trees, which requires a higher time complexity for N points. Specifically, these algorithms do not make any use of the data initial order of points as they come from 3D sensors, but assume general, unordered point clouds to generate their neighborhood structure. In contrast to such approaches, we take advantage of the data order, to gain information about geometric properties of the data points.

This order depends on the sensor, which seems to be a strong condition. However, most current 3D sensors do offer a certain, similar order of data. Examples of such sensors are 3D Lasers, stereo cameras and Microsoft Kinect.

In general 3D laser scan is gained by a 'sweep' (a move of a 2D laser scanner) of single, plane 2D scans. [9] explains different ways to obtain 3D laser scans with similar arrangements. Independent of the actual physical data arrangement, it is always possible to generate a data representation consisting of an ordered sequence ('vertical order') of 2D sub-scans, which themselves consist of an ordered sequence ('horizontal order') of data points. With this assumption our system implements the straightforward approach of substituting data points in 2D sub-scans with 2D geometric features. Using the index-order of raw data as given by the sensor gives a significant speed improvement and defines a neighborhood structure.

In general index structures are only given for single 3D scans; the structure is lost when multiple scans are combined to a global scan (via alignment/registration). That can be remedied by re-creating the order of points via re-scanning using a virtual sensor.

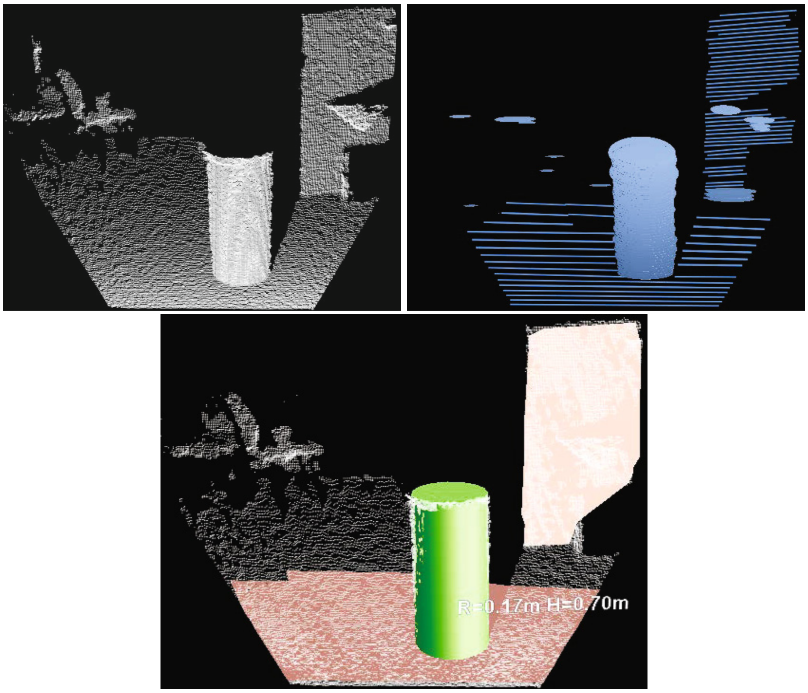


Fig. 2.2 Results of the algorithm. Top: 3D point cloud from MS Kinect (307200 points); Middle: extracted ellipses(purple) and line segments (blue); Bottom: at fitted conic objects and planar patches together with the 3D point cloud. The cylinder in the picture is a cylindrical trash can with radius=0.2m and height=0.7m.

2D features are extracted by traversing each data point once and adding it to a model in constant time $O(1)$ until the accumulated model error reaches a threshold. Therefore at the end of a 3D scan traversal, all 2D features are found in $O(N)$ time complexity, see section 3.3 for more details. This is equivalent to finding necessary conditions for describing a higher dimension geometric models. For example, planes can be represented as a set of co-planar line segments, instead of a set of coplanar points. Same rule also applies to a lower dimension; lines can be represented as co-linear points. Using these rules we define a three step approach to fitting a object model to data points; points to candidate segments (lines or ellipses); candidate segments to sets of segments that form valid objects; sets of segments to objects (planar patches and conic objects), see Figure 2.3.

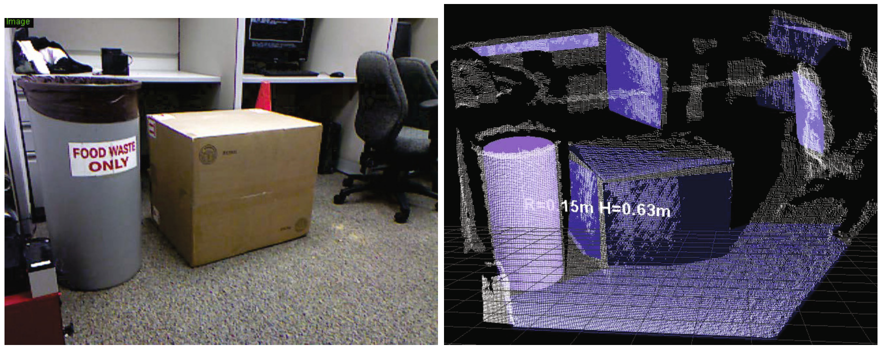


Fig. 2.3 Results of the algorithm. Top: RGB image; Bottom: described scene with fitted conic objects and planar patches together with the 3D point cloud.

Depending on the type of object the rules will be different as described in ([3, 4]). The main advantages of such representation are:

- High data compression rate, which in the case of representing a ball, for example, as a sphere means storing center point and radius. This is in contrast to representing a sphere by thousands of 3D data points.
- Higher geometric information level, when storing data using mid level geometric elements simplifies further processing steps. For example, finding boxes or corners in a map is significantly more simple if the map data consists of planar patches, not of 3D data points, since basic geometric information (e.g. normal-directions) is already given.

Transformation of raw data to a higher representation can be seen as an estimate of hypothetical properties of the scanned environment and it is a pre-processing step which augments the information of the data, yet assuming certain conditions of the environment. Note that the mid level representation is an approximation of the original data set, it comes with loss of precision. The precision of the original data set often represents noise. In case the assumptions of properties (e.g., co-planarity) are justified, precise point-wise representation can lead to over-fitting of the hypothetical structures.

Our system lays out a framework that utilizes natural order of data points to form mid-level geometric features that are used to describe 3D geometric objects (planes and conical objects) in real time. Such type of scene description can be used as a real-time pre-processing module for 3D feature based tasks in robotics. This includes scene analysis, SLAM, etc. based on 3D data. The framework, which suggests the two step approach, is currently implemented for planes and conic objects, see Figure 2.1 .

Real-time processing is achieved by using incremental model fitting approach with an $O(1)$ model update and dimensionality reduction from 2.5D to 1.5D (a 1D height-field, i.e. a single scan row of a range scan). In total, this algorithm achieves

$O(N) + O(M)$ complexity, with N number of data points and M 2D geometric primitives ($N >> M$), resulting in a total complexity of $O(N)$. Note that the extraction of 2D primitives can be parallelized. The implemented JAVA version of this system can process 30 frames per second on 320x240 Kinect data on a 2.8GHz desktop computer.

The paper is organized as follows: after comparison to related work (section 2.2), we give an overview of the approach (section 3.3). We prove the applicability to lighter level robotic tasks and highlight some properties of the algorithms in the experiment section (section 4.4) and conclude in (section 2.6).

2.2 Related Work

Object detection and tracking has been long performed in robot-vision, yet most approaches process 2D RGB images. We will not compare to these approaches, but limit the related work to detection based on range data. The problem of sphere detection has been solved in a variety of different ways. To the best of our knowledge, all previous approaches work on point clouds directly. However none of these approaches make use of a two step solution (points to ellipses, ellipses to spheres). The advantage of the two step solution is, that the higher scale of data, that comes e.g. with 3D data, is immediately reduced: the data is decomposed into sets of lower dimensional spaces, a pre-analysis is performed in this lower dimensionality. The transition to higher dimensions is then performed on mid-level representation, significantly reducing the data elements to be analyzed.

An approach that works directly on 3D point data is described in [12] and uses surface normal vectors and forms clusters of points with similar direction. But its clustering decomposition stays in the 3D space of the original data, which is a major difference to our approach.

In general, data modeling with pre-defined structures can also be solved by the Expectation Maximization (EM) approach. The system in [7] uses a split-and-merge extended version of EM to segment planar structures (not conic elements) from scans. It works on arbitrary point clouds, but it is not feasible for real time operation. This is due to the iterative nature of EM, including a costly plane-point correspondence check in its core. An extension to conic elements would increase the complexity even further.

Random Sample Consensus (RANSAC) can achieve near-optimal results in many applications, including linear fitting (which is the standard RANSAC example). However, RANSAC fails to model detailed local structures if applied to the entire data set, since, with small local data structures, nearly the entire point set appears as outliers to RANSAC. In the approach of [14] RANSAC is used on regions created by a divide and conquer algorithm. The environment is split into cubes (a volume-gridding approach). If precise enough, data inside each volume is approximated by planes, and coplanar small neighboring planes are merged. This approach has similarities to ours, in the sense that it first builds small elements to create larger regions. However, their split stays in the third dimension, while our split reduces the dimensionality from 3D to 1.5D, which makes the small-element generation, i.e. ellipse, a faster operation.

Ellipse extraction is a crucial step in our solution. There are many different methods for detection and fitting ellipses. [5] has done an extensive overview of different methods for detection and fitting ellipses. Theses methods range from Hough transform [17, 16], RANSAC [15], Kalman filtering [11], fuzzy clustering [2], to least squares approach [6]. In principle they can be divided into two main groups: voting/clustering and optimization methods. The methods belonging to the first group (Hough transform, RANSAC, and fuzzy clustering) are robust against outliers yet they are relatively slow, require large memory and their accuracy is low. The second group of fitting methods are based on optimization of an objective function which characterizes a goodness of a particular ellipse with respect to the given set of data points. The main advantages of these methods are their speed and accuracy. On the other hand the methods can fit only one primitive at a time (i.e. the data should be pre-segmented before the fitting). Also the sensitivity to outliers is higher than in the clustering methods. An analysis of the optimization approaches was done in [1]. There were many attempts to make the fitting process computationally effective and accurate. Fitzgibbon et al. proposed a direct least squares based ellipse-specific method in [1].

Our approach to ellipse extraction belongs to the second group as a type of optimization, yet with the advantage of finding multiple objects at one time. It was motivated by the work of [3, 10, 4]. In there, a region growing algorithm is proposed on point clouds, testing an optimal fit of updated planes to the current region. This paper extends and unifies the work of [3, 4] for planes, spheres, cylinders and cones.

2.3 Method Overview

Mid-level elements are lower dimensional objects(here: 2D objects, eg. lines, ellipses) serving the purpose of intermediate step between raw data and targeted 3D objects. The guiding principal of our work is to look for mid-level geometric elements (MLEs) in lower dimensional subspaces and then to extend the data analysis to the missing dimensions using these MLEs. The advantage is, that the number of MLEs is significantly smaller than the number of data points. In addition modeling of mid level elements in lower dimensions is in general an easier and computationally less intensive task. Planar patches are found by performing a region growing algorithm on all candidate sets of line segments. Each of the candidate sets consist of co-planar segments, which are part of the neighborhood structure. Similarly in the case of conical objects, all sets of ellipses that can form conical objects are traversed as 1.5-dimensional subsets (horizontal scan lines with distance data) of $3D$ data. For example, the intersection of conical objects with the scanning plane consists of ellipses. Therefore for each scan line, we traverse its data points iteratively, and try to fit ellipses. This way we can determine maximal connected subsets of each scan line, such that it fits ellipses within a certain radius interval, and under an

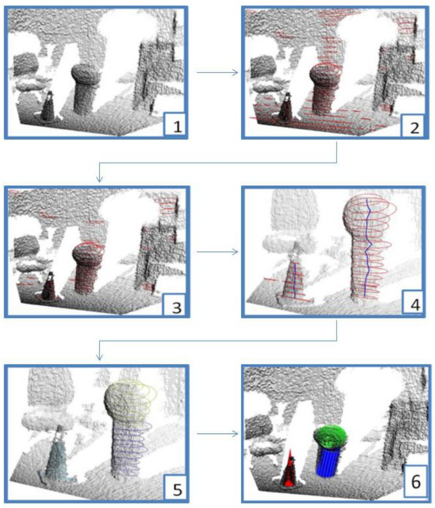


Fig. 2.4 Flow diagram showing: 1) Input data, 2) Result of ellipse fitting, 3) After removal of elongated ellipses, 4) Ellipse candidate sets based on proximity. The blue line shows connected components. 5) Resulting sets of ellipses forming different conical objects 6) Fitted 3D models.

error threshold T_ε. Similarly, in an iterative line segment model update, all lines segments are found. Traversing each scan line therefore leaves us with a set of ellipses $\mathscr{L} = \{L_i^s\}$ and a set of line segments $\mathscr{E} = \{E_i^s\}$, see figure 2.4.

After ellipse detection, it holds for cylinders and cones, that the center points of participating ellipses' center points are collinear, and concyclic for spheres. Cones, spheres and cylinders are characterized by the change of radius along the vertical axis. Hence we can determine, if a connected component of ellipses constitutes one of the models. It is therefore sufficient, to scan the connected components for being collinear or concyclic in order to find model candidates. Please note, that this approach reduces the effort to a simple line or ellipse fitting, this time even in the significantly smaller space of center points. Such lines and ellipses from center point space are again found with an iterative $O(1)$ update approach.

In conclusion, we find simple geometric models, ellipses and lines, in a subspace of reduced dimensionality (1.5D), then perform low dimensional fitting again in a 1.5D space determined by the model. This split leads to an addition instead of multiplication of run times, which makes the approach fast. Please see Figure 2.4 for an illustration of the approach.

2.3.1 Line Segment Extraction

We apply an incremental line extraction which works in $O(N)$, similar to [13], and [8], yet implemented optimally using a line-update technique, see Algorithm 1.

Algorithm 2. getMidLevelPrimitive(Points, THRESHOLD)

```
while $(nextPoint = Points.next()) \neq NULL$ do
    $addPoint(nextPoint)$
    $fitModel()$
    $calculateError()$
    if $error > THRESHOLD$ then
        $removeLastPoint()$
        $saveCurrentModel()$
        $startNewModel()$
    end if
end while
```

Please note that the line extraction processes point-sets from single 2D scans only (usually less than 1000 points), hence even algorithms with asymptotic complexity greater than $O(N)$ would be feasible for real-time applications.

2.3.2 Ellipse Extraction

A well known non-iterative approach for fitting ellipses on segmented data (all points belong to one ellipse) is Fitzgibbon's approach described in [1]. And a numerically more stable version [5] which is also an optimization approach. However, both algorithms only fit a single ellipse on a given set of pre-segmented points. For this system we used a solution based on [4] that can find multiple ellipses in non segmented data in $O(N)$. This is done by performing an ellipse model update in $O(1)$ and making use of the natural point order defined by the range sensor, see Algorithm 1.

2.3.3 System Limitations

Being based on finding ellipses from the conical intersection of a scanning plane with cylinders, cones and spheres it is limited with respect to tilting angles that lead to non-elliptical intersection patterns. In practice, our system performs robustly on tilting angles of greater than 45°, see Figure 2.5, yet is naturally limited when the angle becomes significantly larger. One remedy of this problem (for tilts resulting from rotation around the z axis), is to also use ellipses found by intersection of vertical instead of horizontal scanning planes(i.e a 90°rotated sight of the scene), which turns tilts of greater than 45°into tilts of less than 45°. An example of tilted objects is illustrated in Figures 2.6 and 2.5.

2.4 3D Object Extraction

The plane extraction used in this system implements a region growing algorithm described in [3], which works in real-time with $O(N)$ time complexity.

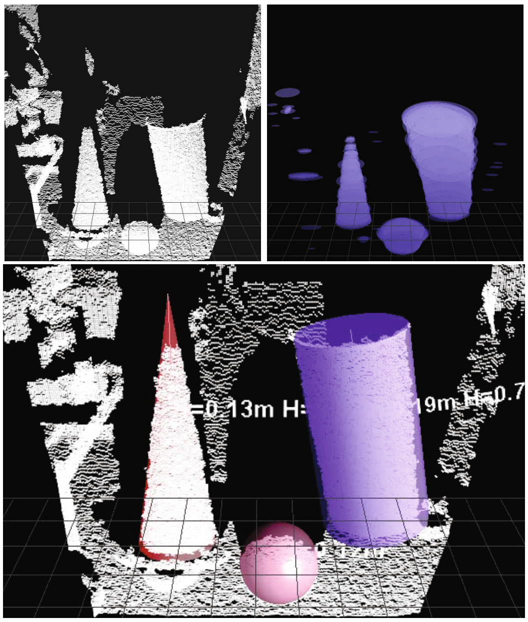


Fig. 2.5 Results of the algorithm. Top left: 3D point cloud from MS Kinect (307200 points); Top right: extracted ellipses after the pre-filter; Bottom: fitted conic objects and 3D point cloud. The cylinder in the picture is a cylindrical trash can.

Conical objects are formed by ellipses which centers are co-linear(cylinder and cone) or lie on a circle (in the case of sphere). All candidate sets of ellipses are ordered by the order of their corresponding 2D sub-scan, which imposes a neighborhood structure. For more details on the algorithm please refer to [4].

Due to this order, we can extract collinear center-points using an $O(|\mathscr{E}|)$ line fitting algorithm, which in its core again has an $O(1)$ update. For details on incremental $O(n)$ line detection, please see [3]. Please note that $|\mathscr{E}|$ is significantly smaller than n, in practice it is connected to the number of scan-lines $\sqrt{n}$. Similarly, ellipse

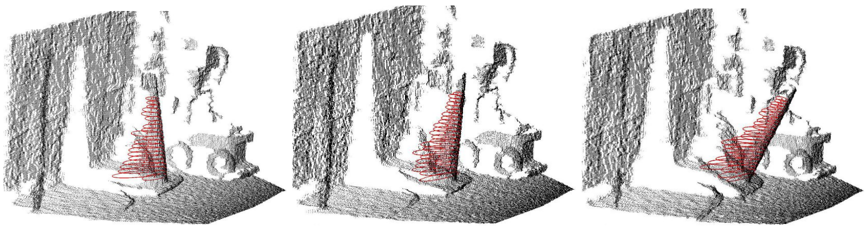


Fig. 2.6 A tilted (rotation around x-axis, tilted towards camera) parking cone, and ellipses detected (red). Top Left: tilt=5 degrees, Top Right: tilt=20 degrees, Bottom: tilt=40 degrees. In all cases, the algorithm is able to fit a cone model (not explicitly shown in figure).

models fitting the center points are extracted, using the method described in Section 2.3.2.

The detection of objects and collinear sequences is performed in parallel, again using incremental line (and ellipse) fitting procedures, yet in different spaces: while the center-collinearity is detected by a line fitting algorithm in the (x,y,z) space, the detection of 3D objects is performed by a line fitting (for cylinders and cones) and ellipse fitting (for spheres) in the (z,r) space. The incremental order in z reduces the collinearity finding to a 1.5D problem. Please note that the same concept of incremental line/ellipse fitting is utilized to solve tasks not only in different phases of the algorithm (first, ellipse finding in scan rows, then line/ellipse finding on ellipse center points), but also in different spaces, the decomposed 1.5D data space, and the (z,r) space. Using the $O(1)$ update principle in all cases is the main reason for the fast performance of the algorithm. In addition, the nature of the fitting procedures automatically separates objects in composed scenes, see e.g. Figure 2.7, cylinder and sphere.

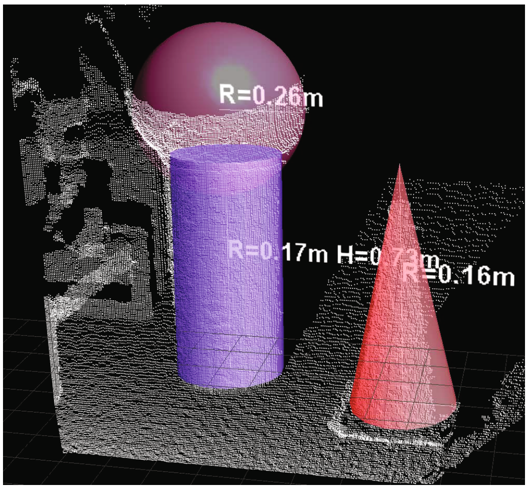


Fig. 2.7 Results of the algorithm showing fitted 3D geometric models. The sphere placed on top of a cylinder is successfully recognized as a separate object.

2.5 Experiments

2.5.1 3D Kinect Experiments

We used 3D range data from a Microsoft Kinect sensor to test how accurate the algorithm recognizes 3D Objects, using the methods described in Section 2.4. The scene consist of a trash can (cylinder), parking cone and exercise ball (sphere) on the floor next to each other, see Figure 2.5. The algorithm successfully finds the correct conical objects in the scene in real-time with a rate of 30fps for 320 × 240 Kinect depth data, implementation in JAVA on Dell 2.8GHz desktop with 8GB Ram.

2.5.2 *Object Tracking*

In this set of experiments we tested how fast the algorithm can track a moving object. A ball was chosen as the object of interest and two separate tests were performed. First test used a large exercise ball (radius = 27cm) thrown in a straight line, and after a several bounces continues to roll. The second test used a much smaller ball (radius = 10cm) that bounces in forward direction, see Figure 2.8.

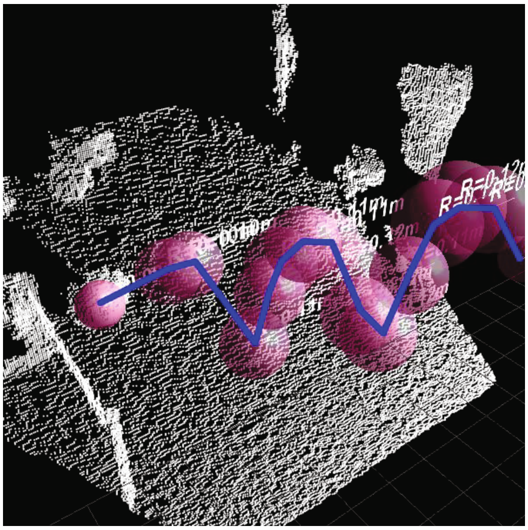


Fig. 2.8 Results of tracking a ball. The blue line connects center points of all spheres found while the ball was bouncing.

2.5.3 *Mobile Robot Experiment*

For the last experiment we mounted a Microsoft Kinect sensor on a Pioneer mobile robot to test how accurate the robot can recognize and navigate towards conical objects (Cylinder, Cone and Sphere). All computations were done on a notebook (2.3GHz, 2GB Ram) on the robot. The setup for the experiment is as follows: the robot has to track a ball and navigate towards it until the sphere is a meter away. In the first test the ball was held by a walking human. During the second test the ball was thrown over the robot bouncing in the desired direction. In both tests the robot detected and navigated towards the ball while moving, see Figure 2.9.

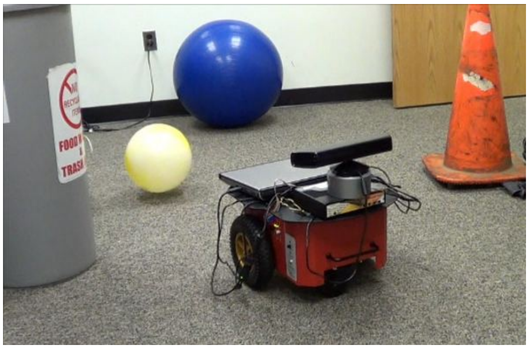


Fig. 2.9 Our robot tracking a ball in real-time only using only 3D range data from MS Kinect (307200 points)

2.6 Conclusion and Future Work

We presented RMSD, a real-time mid-level scene description system that can describe a 3D scene using planar patches and conical objects using a three step approach (points to mid0level primitives (ellipses, line segments) to objects). It provides fast $O(n)$, n number of points and robust in extracting all planar patches and conic objects. The algorithm is well behaved towards noise and can aid higher level tasks, for example, autonomous robot navigation by providing more robust landmark features. This system can be extended to include other geometric based models, based on the specific task.

Acknowledgements. The project is sponsored by the grant ARRA-NIST-10D012 of the National Institute of Standards and Technology (NIST).

References

1. Fitzgibbon, A., Pilu, M., Fisher, R.: Direct least square fitting of ellipses. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(5), 476–480 (1999), doi:10.1109/34.765658
2. Gath, I., Hoory, D.: Fuzzy clustering of elliptic ring-shaped clusters. Pattern Recognition Letters 16(7), 727–741 (1995), `http://www.sciencedirect.com/science/article/pii/016786559500030K`, doi:10.1016/0167-8655(95)00030-K
3. Georgiev, K., Creed, R., Lakaemper, R.: Fast plane extraction in 3d range data based on line segments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS (2011)
4. Georgiev, K., Lakaemper, R.: Real-time 3d scene description using spheres, cones and cylinderss - submitted. In: ICRA 2013 Workshop on Mobile Robot Navigation and Mapping (2013)

5. Halir, R., Flusser, J.: Numerically stable direct least squares fitting of ellipses (1998)
6. Bookstein, L.F.: Fitting conic sections to scattered data. Computer Graphics and Image Processing 9(1), 56–71 (1979), http://www.sciencedirect.com/science/article/pii/0146664X79900820, doi:10.1016/0146-664X(79)90082-0
7. Lakaemper, R., Jan Latecki, L.: Using extended em to segment planar structures in 3d. In: ICPR 2006: Proceedings of the 18th International Conference on Pattern Recognition, pp. 1077–1082. IEEE Computer Society, Washington, DC (2006)
8. Nguyen, V., Gächter, S., Martinelli, A., Tomatis, N., Siegwart, R.: A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. Auton. Robots 23(2), 97–111 (2007), doi:http://dx.doi.org/10.1007/s10514-007-9034-y
9. Nuechter, A.: 3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom. Springer Publishing Company, Incorporated (2009)
10. Poppinga, J., Vaskevicius, N., Birk, A., Pathak, K.: Fast plane detection and polygonalization in noisy 3d range images, pp. 3378–3383 (2008)
11. Rosin, P., West, G.: Nonparametric segmentation of curves into various representations. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(12), 1140–1153 (1995), doi:10.1109/34.476507
12. Rusu, R., Blodow, N., Marton, Z., Beetz, M.: Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, pp. 1–6 (2009), doi:10.1109/IROS.2009.5354683
13. Vandorpe, J., Van Brussel, H., Xu, H.: Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder, vol. 1, pp. 901–908 (1996), doi:10.1109/ROBOT.1996.503887
14. Weingarten, J., Gruener, G.: A fast and robust 3d feature extraction algorithm for structured environment reconstruction. In: Reconstruction, 11th International Conference on Advanced Robotics, ICAR (2003)
15. Werman, M., Geyzel, Z.: Fitting a second degree curve in the presence of error. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(2), 207–211 (1995), doi:10.1109/34.368167
16. Wu, W.Y., Wang, M.J.J.: Elliptical object detection by using its geometric properties. Pattern Recognition 26(10), 1499–1509 (1993), http://www.sciencedirect.com/science/article/pii/003132039390155P, doi:10.1016/0031-3203(93)90155-P
17. Yip, R.K., Tam, P.K., Leung, D.N.: Modification of hough transform for circles and ellipses detection using a 2-dimensional array. Pattern Recognition 25(9), 1007–1022 (1992), http://www.sciencedirect.com/science/article/pii/003132039290064P, doi:10.1016/0031-3203(92)90064-P

Chapter 3
Semantic and Spatial Content Fusion for Scene Recognition

Elahe Farahzadeh, Tat-Jen Cham, and Wanqing Li

Abstract. In the field of scene recognition, it is usually insufficient to use only one visual feature regardless of how discriminative the feature is. Therefore, the spatial location and semantic relationships of local features need to be captured together with the scene contextual information. In this paper we proposed a novel framework to project image contextual feature space with semantic space of local features into a map function. This embedding is performed based on a subset of training images denoted as an exemplar-set. This exemplar-set is composed of images that describe better the scene category's attributes than the other images. The proposed framework learns a weighted combination of local semantic topics as well as global and spatial information, where the weights represent the features' contributions in each scene category. An empirical study was performed on two of the most challenging scene datasets *15-Scene* categories and *67-Indoor* Scenes and the promising results of **89.47** and **45.0** were achieved respectively.

3.1 Introduction

The aim of scene classification is typically to assign scene labels (such as *library*, *kitchen*, *highway*) to images. It is a difficult task due to the variation of scene

Elahe Farahzadeh
Center of Computational Intelligence, School of Computer Engineering,
Nanyang Technological University, Singapore 639798
e-mail: `elah0001@ntu.edu.sg`

Tat-Jen Cham
Center for Multimedia and Network Technology, School of Computer Engineering,
Nanyang Technological University, Singapore 639798
e-mail: `astjcham@ntu.edu.sg`

Wanqing Li
Information and Communication Technology (ICT) Research Institute University of Wollongong, NSW 2522, Australia
e-mail: `wanqing@uow.edu.sg`

Y. Sun et al. (eds.), *New Development in Robot Vision*,
Cognitive Systems Monographs 23, DOI: 10.1007/978-3-662-43859-6_3

structures, scales and illumination. Although many proposed scene recognition frameworks attempt to learn an optimal classifier based on low-level visual features, these features alone are often not able to offer sufficient discriminative power.

For example, while local features may be very useful in classifying scene images, the classification accuracy can drop dramatically when key objects are not sufficiently large, or if the objects are occluded. Previous literature has demonstrated that exploiting semantic relationships of local features [7, 5] and their spatial locations [10, 3] can lead to promising results, but inadvertently affect the classification of indoor scene where the interclass variation is very high. In order to address this issue, global or contextual information in the images must be used since it can capture the inherent context.

There have been several attempts in integrating contextual information with local features [19, 25, 8]. Quattoni and Torralba [19] combined image *Regions-of-Interest*(ROIs) and global *GIST* features together in a prototype-based framework, where the ROIs were extracted through segmentation or manual annotation. A well-known method for feature fusion is Multiple Kernel Learning (MKL) [9]. This method was also implemented for scene recognition in [25, 8] using a number of visual features, such as multiple local features and global contextual features. The method involves having a separate kernel associated with each feature, after which a linear combination of these kernels is learned for SVM classification. Although MKL is an effective method for feature combination, computational cost is often very high. This is especially the case when the kernels are complex since the parameters for every kernel have to be tuned.

In this paper we propose a novel distance learning method which combine semantic relationships and the location information of local features with contextual global features. Opposed to Xiao et al. [25] where many different features are required to obtain higher accuracies in scene recognition, our proposed method uses fewer features and more effective fusion model to achieve robust scene recognition. Since feature extraction is one of the most time consuming steps in scene recognition, our method has advantage in this this perspective. Specifically, image local information is represented by a histogram of local semantic topics, after embedding the visual *bag-of-words* [4] histograms into a Probabilistic Latent Semantic Space (pLSA) [7]. In contrast to kernel combination methods that perform feature fusion globally, our method carries this out locally by weighting each of the local semantic patches individually. It differs from the previous pLSA-based methods [20, 2, 5] where all the semantic topics had the same discriminative power. In our framework, semantic topics that have higher relevance in identifying a particular category are more heavily weighted. Besides the novelty of the proposed distance function, another contribution of this work is the category-dependent map function, which is learned based on a special subset of training images, called an *exemplar-set*, that embeds the proposed distance function. In this map function, the dissimilarity of each exemplar image and the training image is weighted to determine its relevance to the scene category.

The rest of the paper is organized as follows. Section 6.2 provides a review of related work on scene recognition. The overview of our proposed model for integrating the visual features is described in Section 3.3. The image visual representations and models which we incorporate into our framework are detailed in Section 3.4. Section 3.5 explains the Spatial-Semantic Feature Fusion framework. The experimental evaluation and discussion are presented in Section 3.6. The final section concludes the paper.

3.2 Related Work

In this section, we review some recently proposed methods in scene recognition. The *bag-of-words* framework has been shown to be useful in scene categorization [4]. In this approach, the images are quantized based on a visual vocabulary. The main drawbacks of this method are that (i) the semantic relations between features are not considered, and (ii) this method does not include spatial information. Some other approaches [20, 2, 5] consider each image as a mixture of semantic hidden variables using either pLSA or the Latent Dirichlet Allocation (LDA) [1] model, or build a semantic vocabulary by co-clustering the visual words [12, 13, 21].

There are also methods that capture the spatial information of features, the most effective method so far being the Spatial Pyramid Matching(SPM) by Lazebnik et al. [10]. This method divides the image into regions with several resolution levels and the histogram of visual words is computed over the resulted subregions. Frameworks have been developed to incorporate SPM and better classification accuracy was reported [3, 24]. Bosch et al. [3] add position information into their pLSA framework using three different approaches, *xy-pLSA, ABS-pLSA and SP-pLSA*. In xy-pLSA position information is concatenated to feature vector. ABS-pLSA quantize location information into one of the embedded bins for feature positions and SP-pLSA convey location information by incorporation SPM. Highest results achieved in [3] for SP-pLSA. To capture the image contextual layout, the global GIST feature [15] was proposed. However, this method does not perform well for indoor scene classification. Wu and Rehg [24] proposed a new descriptor called CENsus TRansform hISTograms(CENTRIST) for scene recognition to capture image structure. CENTRIST can be applied on the whole image and used as a global feature. There is also a spatial representation of CENTRIST called sPACT descriptor which is constructed by employing the CENTRIST to image blocks and constructing the spatial pyramids over them.

There are approaches that try to deal with scene recognition challenges by imposing high level concepts [11, 19, 17]. In [19] the images are first segmented, after which the ROIs and global information are combined together in a prototype-based model. Since every single parameter in their learning model is based on the prototype images, an excessive number of parameters has to be optimized, which increases computational complexity. In [11], the images are represented by a vocabulary of objects called *object-bank*. Pandey and Lazebnik [17] suggested using Deformable Part-based Models (DPM) with latent SVM [16]. By applying DPM on scene images, salient objects and visual elements of scenes are captured. Afterwards, DPMs and global features are combined together to achieve more promising results. Despite their high computational complexity, these methods do not offer much increase in recognition accuracy. Other works [8, 25] suggested feature fusion by Multiple Kernel Learning [9], to improve the recognition accuracy in scene classification. In these works, various visual features are combined together and a separate kernel is associated for each feature.

Weighting latent semantic topics. In contrast to kernel combination methods that perform feature fusion by combining histogram representations of image without considering the importance of each histogram bin, our method carries this out by weighting each of the semantic topics individually. It differs from the previous pLSA-based methods [20, 2, 5] where all the semantic topics had the same discriminative power. In our framework, semantic topics are weighted based on their significance level in identifying a particular category.

Exemplar-based distance learning. A special subset of training images, called an *exemplar-set* which include the most discriminative images in each category are selected. We use weighted Exemplar-set as a basis to assign a scalar measure to each image. The scalar value shows the level of confidence in belonging image x_i to category C_i.

Category-dependent map function. Besides the novelty of the proposed distance function, another contribution of this chapter is the category-dependent map function. In this map function the degree of membership for each Exemplar image E_i to the categoryC_i should be determined; to this end the proposed map function embeds the learned distances.

3.3 Overview of the Proposed Framework

The objective of our scene recognition system is to learn a model which maps images x to scene categories y. The scene model presented in this paper uses a training set $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ to learn the model parameters. The training set T consists of n pairs of image x_i and label y_i, wherein each scene label $y_i \in \{-1, 1\}$ is a binary label indicating whether the image x_i belongs to a scene category or not. Based on a *one-versus-all* strategy, we compute a mapping function $h(x)$ for each scene category that quantitatively estimates the affinity of a test image x to that scene category. The scene category that best describes image x can simply be determined by the $h(x)$ that returns the largest value.

According to the learning model's block diagram in Figure 3.1 after extracting the visual features, to perform the scene recognition modeling, we retain a subset of training images called the *exemplar-set*: $E = \{E_1, E_2 ... E_p\}$. The exemplar images are chosen for each category in the dataset. The selection is done such that for each category the most discriminative images are chosen. Exemplar-set selection is described in detail in Section 3.5.1. The exemplar-set is used as a basis to measure the distances. After defining appropriate distance functions, the exemplars again are used to embed the distances into their relevant map functions. Afterwards the functions parameters are tuned by performing a parameter learning.

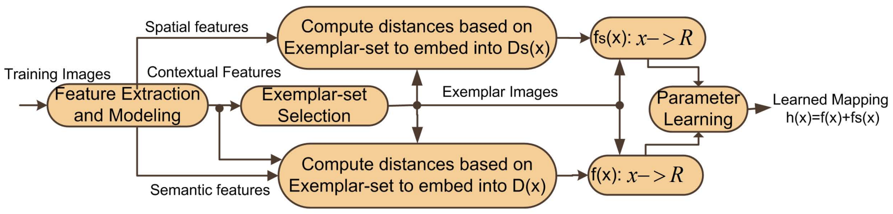

Training Images
Feature Extraction and Modeling
Spatial features
Contextual Features
Semantic features
Compute distances based on Exemplar-set to embed into Ds(x)
Exemplar-set Selection
Exemplar Images
Compute distances based on Exemplar-set to embed into D(x)
fs(x): $x->R$
f(x): $x->R$
Parameter Learning
Learned Mapping h(x)=f(x)+fs(x)

Fig. 3.1 Block diagram for the proposed learning model

3.4 Feature Extraction and Representation

In this paper we use three kinds of low level features: **SIFT** [14], **GIST** [15] and **CENTRIST** [24].

To capture semantic representation of SIFT and spatial representation of SIFT and CENTRIST, initially the local feature topics have to be exploit. According to literature [2], dense features are more discriminative than interest points for scene recognition. Therefore by following the settings of [10, 24] images are divided into 16×16 patches with 50% overlapping both horizontally and vertically. SIFT and CENTRIST descriptors [14, 24] are extracted from each patch.

3.4.1 Capturing Semantic Information

The pLSA [7] is a statistical model which employs hidden variables together with a bag-of-words representation of the image. Suppose we have the co-occurrence matrix, n, of N_x by M_w where N_x is the number of images and M_w is the size of visual vocabulary. $n(w_i, x_j)$ indicates word w_i appears n times in image x_j.

The pLSA models each image as a mixture of topics. It assumes there is an unobserved latent variable $z_k \in \{z_1, z_2 \ldots, z_K\}$ which is associated with each observation (w_i, x_j). The probability of co-occurrence w_i in x_j is represented by $P(w_i, x_j) = P(w_i|x_j)P(x_j)$. In order to find the conditional probability $P(w_i|x_j)$, the joint probability is marginalized over latent topics z_k :

$$P(w_i|x_j) = \sum_{k=1}^{K} P(w_i|z_k)P(z_k|x_j) \tag{3.1}$$

where $P(z_k|x_j)$ is the probability of occurring topic z_k in document x_j and $P(w_i|z_k)$ is the probability of visual word w_i occurrence in latent topic z_k.

It should be noted that the number of semantic topics K can be much higher than the number of scene categories, unlike in unsupervised frameworks [23]. In our experiments, the number of semantic topics for the 15-Scene dataset is set to 70, same as in [21].

The pLSA model parameters are learned by maximum likelihood estimation using the Expectation Maximization(EM) algorithm. Thus the objective function to be maximized is:

$$S = \prod_{i=1}^{M_w} \prod_{j=1}^{N_x} P(w_i|x_j)^{n(w_i,x_j)} \tag{3.2}$$

$P(w_i|x_j)$ is obtained from (3.1). Consequently what we use in our model is the image representation of $P(z_k|x_j)_{k=1,2,...,K}$.

3.4.2 Capturing Contextual Information

In this paper we used two holistic features that are applied on the whole image to extract the image contextual information.

GIST. The GIST descriptor is computed by applying 4×4 Gabor filters on 4 scales and in 6 different orientations on spatial blocks of image.

CENTRIST. In this visual descriptor, each pixel value is replaced by its Census Transform(CT) [26] value. In Census Transform, the intensity value of center pixel in compared to the values of Surrounded pixels in a 3×3 neighborhood, if the value of neighboring pixels is bigger that the center pixel, bit 0 is assigned to the corresponding location otherwise the value is set to 1. Finally the CT value is obtained by decoding the generated binary code to a base-10 number. Afterwards the CENsus TRansform hISTograms(CENTRIST) are computed from these transformed CT values.

3.4.3 Capturing Spatial Location Information

Spatial pyramids are constructed over two features *SIFT* and *CENTRIST*. We used the same number of visual words and level of resolution as the original papers [10, 24].

SPM. The spatial pyramid of *bag-of-words* is build using the SIFT local features. To construct this spatial pyramid histograms the extracted dense features are quantized into 400 visual words in three levels of resolution (L=2 in SPM implementation by Lazebnik et al. [10]). These histograms are computed on the image grids of 1×1, 2×2 and 4×4.

sPACT. This spatial representation is also based on the spatial pyramid matching schema but the splitting of the image at each level is different. In level 2, the image is divided into 4×4 regions. To prevent artifacts of this non-overlapping division, the division is shifted to form 9 more regions. There are a total of 31 blocks, comprising 25 from level 2, five from level 1 and one from level 0. A CENTRIST descriptor is applied on each of these blocks, followed by PCA to reduce the dimensionality of the features to 40. The Spatial Principal component Analysis of Census Transform Histogram (sPACT) is build by concatenating these block representations.

3.5 Spatial Semantic Feature Fusion (SSFF)

To better describe the SSFF method, it is divided into three main blocks, Exemplar-Set selection (Section 3.5.1), learning phase (Section 3.5.2) and scene type recognition phase (Section 3.5.3).

3.5.1 Exemplar-Set Selection

Similarity-based classification methods compute pairwise distances between the images, in this paper we are proposing a new similarity space which indicates a base-set for distance measurements. This base-set is called an exemplar-set and is obtained by downsampling of the training set to choose the most robust images of each category. Utilizing the exemplar-set to find distances will decrease time and memory cost due to the reduced number of distance measurements. The proposed SSFF method weights every exemplar based on its relevance to the scene category, therefore, reducing the size of exemplar-set will decrease the complexity because the number of parameters to be learned are decreased. Besides the reduction in computational complexity, use of the exemplar-set also results in improved accuracy, especially in more complex datasets with high intra-class diversity (e.g. 67-Indoor Scenes). Since the corresponding set consists of the most representative images from each category, hence, when sufficient number of exemplars are chosen, it prevents from less-relevant images to affect our learning.

Figure 3.2 shows the diagram of exemplar-set selection. The goal is to select a robust set of exemplars $E = \{E_1, E_2 ... E_p\}$ from the scene dataset. This exemplar-set is a composition of image subsets from each category. These subsets consist of images that capture the specific characteristics of their classes more effectively than the other images in the same category $E = \{E_1, E_2 ... E_{p_c}\}$.

To this end (see figure 3.2), for each category C_i an initially smaller subset of training images is randomly chosen and fixed as a basis. The rest of the training images in category C_i is then considered as candidates for the exemplar set. A classifier is trained for each of these exemplar candidates by adding this image to the initial basis, and then we determine how well these classifiers perform on the remainder of the training images that are not in the basis. The recognition accuracies for all of the exemplar candidates within category C_i are measured and a selected number of exemplar candidates that resulted in the highest accuracies become the exemplar subset for class C_i. In this paper, each category employs a fixed number of exemplars, denoted by p_c.

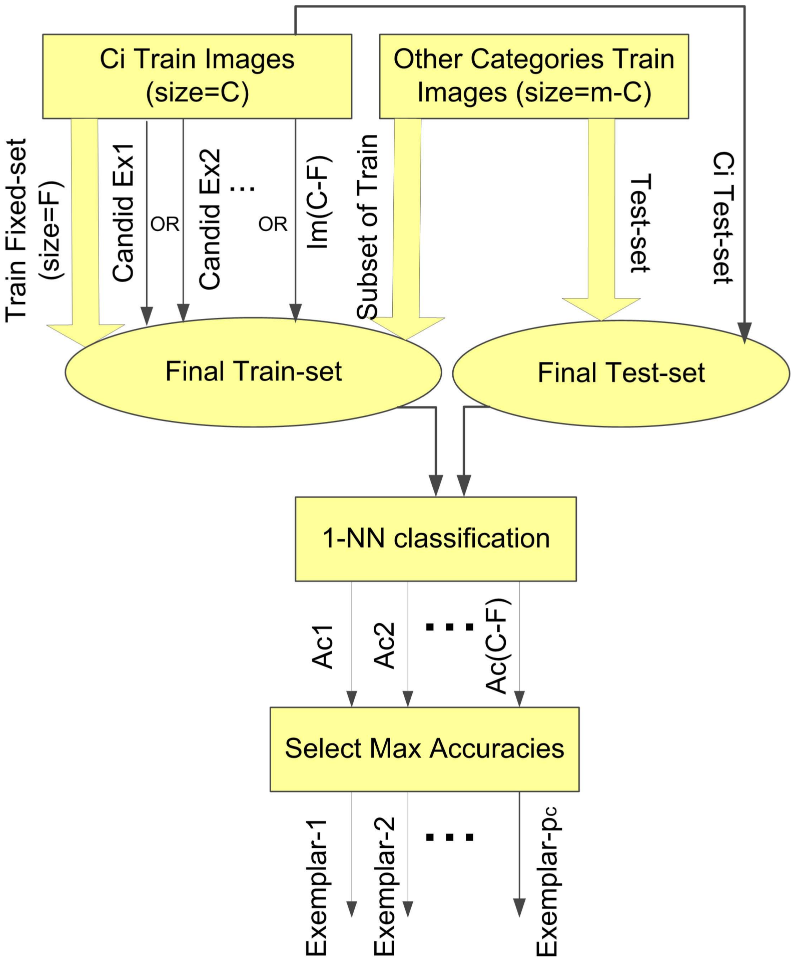


Fig. 3.2 Exemplar-set selection

3.5.2 Learning Phase for SSFF Method

In the learning phase (see figure 3.3), the holistic contextual features of GIST and CENTRIST are extracted from the whole image, while the feature descriptors of SIFT $SF = \{SF_1, SF_2 ... SF_n\}$ are applied on the local patches, semantic and spatial representation are captured from the SIFT features. Although CENTRIST visual feature is a contextual global feature, it could also be applied on the image patches (denoted by $CF = \{CF_1, CF_2 ... CF_{31}\}$) after spatial partitioning of the image to build the CENTRIST-based spatial representation.

As mentioned the purpose of SSFF method is to project every image x within the C class scene dataset, to C real numbers where each number shows the affinity of

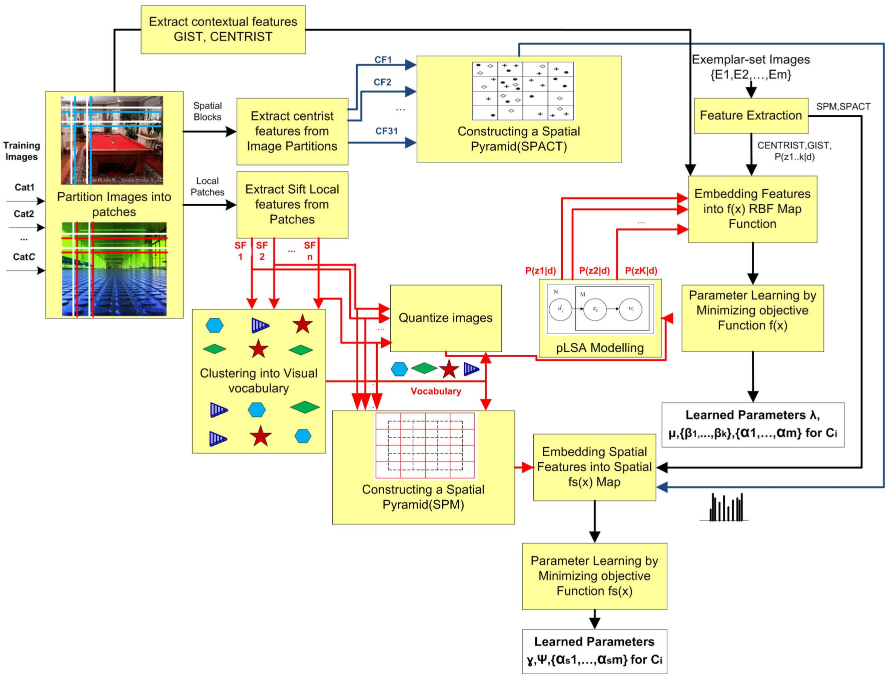

Extract contextual features GIST, CENTRIST
Training Images
Cat1
Cat2
CatC
Partition Images into patches
Spatial Blocks
Local Patches
Extract centrist features from Image Partitions
CF1
CF2
CF31
Constructing a Spatial Pyramid(SPACT)
Exemplar-set Images {E1,E2,...,Em}
Feature Extraction
SPM,SPACT
CENTRIST,GIST, P(z1..k|d)
Extract Sift Local features from Patches
SF 1
SF 2
SF n
Embedding Features into f(x) RBF Map Function
P(z1|d)
P(z2|d)
P(zK|d)
Quantize images
pLSA Modelling
Parameter Learning by Minimizing objective Function f(x)
Clustering into Visual vocabulary
Vocabulary
Learned Parameters λ, μ,{β1,...,βk},{α1,...,αm} for Ci
Constructing a Spatial Pyramid(SPM)
Embedding Spatial Features into Spatial fs(x) Map
Parameter Learning by Minimizing objective Function fs(x)
Learned Parameters γ,Ψ,{αs1,...,αsm} for Ci

Fig. 3.3 Block diagram of learning phase in our Feature Fusion method

the image x to one of the categories. To perform this mapping we use an adoptive category-dependent method, in this method for each C_i category the mapping is consist of learning two separate map functions $f(x)$ and $f_s(x)$, while the final $h(x)$ map function is summation of normalized $f(x)$ and $f_s(x)$ functions, $h(x) = f(x) + f_s(x)$.

The $f(x)$ function is used to map image x based on its contextual and semantic representations. To this end the exemplar-set is used as a basis for measurement, therefore the weighted dissimilarity of image x to exemplar-set is computed using their semantic and contextual features, while the weights indicates the relevance of each exemplar images to C_i category. There are as well parameters which present the amount of contribution for the features meet in this measurement(the importance of GIST (λ) and CENTRIST (μ) contextual features also the significance of each latent topic in semantic representation explicitly ($B_1, B_2 ... B_k$)). Consequently these weights and parameters are assigned during the parameter learning.

On the other hand $f_s(x)$ mapping relies on spatial description of image x. In the parameter learning for this function importance of SIFT spatial structure (γ) and CENTRIST spatial structure (ψ) are balanced. The exemplar images are as well weighted during the learning.

The basic distance metric to measure dissimilarity for $f(x)$ function, is euclidean distance while $f_s(x)$ function utilize Histogram Intersection Kernel (HIK).

The different essence of euclidean space and HIK kernel space justifies utilizing two separate map functions, therefore the distances in $f(x)$ are embedded into a Gaussian radial basis function (RBF) kernel while HIK distances in $f_s(x)$ have to be followed by a linear kernel.

3.5.2.1 Formulation of Distance Functions

Our distance function $D_i(x)$ measures the contextual-semantic dissimilarity between image x and exemplar image E_i and is defined by

$$D_i(x) = \lambda G_i(x) + \mu Ct_i(x) + L_i(x) \tag{3.3}$$

where $G_i(x)$, $Ct_i(x)$, $L_i(x)$ are component distances that are respectively related to GIST, CENTRIST and local semantic features. They are positive and normalized such that $\sum_i G_i(x) = 1$, $\sum_i Ct_i(x) = 1$ and $\sum_i L_i(x) = 1$. This normalization balances the distribution of image x over the whole exemplar-set. The parameters λ and μ control how discriminative the contextual features are for a particular scene category. $G_i(x)$ is the distance function measuring the dissimilarity of GIST contextual features between image x and exemplar image E_i using l_2 norm:

$$G_i(x) = \|gist(x) - gist(E_i))\| . \tag{3.4}$$

$Ct_i(x)$ is associated with the distance between the CENTRIST representation of image x and exemplar image E_i:

$$Ct_i(x) = \|centrist(x) - centrist(E_i))\| . \tag{3.5}$$

$L_i(x)$ measures the dissimilarity of the semantic topics of image x and exemplar image E_i using l_2 norm::

$$L_i(x) = \beta\,[P(z|x) - P(z|E_i)] \tag{3.6}$$

where vector $\beta = [b_1\, b_2 \ldots b_K]^T$ represents the relevance of all the K semantic topics of image x for predicting that scene category. In (3.6), $P(z|x)$ and $P(z|E_i)$, are vectored probability representations of x and E_i (corresponding to image x and exemplar E_i) by latent topics z, referred to image representation in latent semantic space which is defined in Section 3.4.1.

Thus (3.6) expands as

$$\begin{aligned} L_i(x) = & b_1(P(z_1|x) - P(z_1|E_i)) + b_2(P(z_2|x) - P(z_2|E_i)) \\ & + \ldots + b_K(P(z_k|x) - P(z_k|E_i)). \end{aligned} \tag{3.7}$$

Contextual-semantic distances, given by $D(x) = (D_1(x), D_2(x), \ldots, D_p(x))$, are evaluated between the image x and all images in exemplar-set E.

The spatial distance function $D_{si}(x)$ is defined as follows:

$$D_{si}(x) = \gamma S_i(x) + \psi P_i(x) \tag{3.8}$$

where γ and ψ are the weights on distances between image x and exemplar E_i with respect to the *spatial* representations of SIFT-related SPM features and CENTRIST-related sPACT features.

The function $S_i(x)$ is the distance between image x and exemplar E_i using their SPM representations. The function $P_i(x)$ on the other hand computes the distance between the corresponding sPACT features. $S_i(x)$ and $P_i(x)$ are positive and normalized such that $\sum_i S_i(x) = 1$ and $\sum_i P_i(x) = 1$. As in previous work, the similarity of histograms at each level of resolution in the spatial pyramids representation, whether using SIFT or CENTRIST, is based on histogram intersection. Therefore to obtain the overall HIK distance between two spatial features, the weighted sum of these histogram intersections are computed, as in Pyramid Match Kernel(PMK) [6].

Spatial distances, given by $D_s(x) = \{D_{s1}(x), D_{s2}(x), ..., D_{sp}(x)\}$, are evaluated between the image x and all images in exemplar-set E.

3.5.2.2 Formulation of Map Functions

As mentioned before, each image is mapped to a real number which represent the confidence score of belonging the image x to category y, using the $h(x)$ map function. The defined distance functions in Section 3.5.2.1 are embedded into the map functions, therefore in addition to the proposed feature weights, α_i / $\alpha_s i$ parameters are learned for each exemplar E_i to measure that exemplar's contribution for each scene category.

The various distance functions computed in Section 3.5.2.1 are embedded in a non-linear map function $f(x)$, formulated as:

$$f(x) = \sum_{i=1}^{p} \alpha_i \exp(-D_i(x)) \tag{3.9}$$

In the proposed mapping, $\alpha_i \in \mathbb{R}$, are parameters that indicate how discriminative each of the exemplars are in that scene category.

The spatial distance vector $D_s(x)$ on the other hand is embedded into a linear map function $f_s(x)$:

$$f_s(x) = \sum_{i=1}^{p} \alpha_{si} D_{si}(x) \tag{3.10}$$

where α_{si} as before indicates the discriminativeness of each exemplar E_i.

The final map function for image x is:

$$h(x) = f(x) + f_s(x) \tag{3.11}$$

where the $f(x)$ and $f_s(x)$ functions are standard normalized such that they have zero mean and unit norm. The assigned parameters $\lambda, \mu, \beta, \alpha, \gamma, \psi, \alpha_s$ are estimated during parameter learning as discussed in Section 3.5.2.

3.5.2.3 Parameter Learning

According to definition of the map functions $f(x)$ and $f_s(x)$. for semantic-contextual function $f(x)$ the distances are measured using euclidean distance and they are embedded into a RBF kernel, for spatial function $f_s(x)$ on the other hand, we use a HIK kernel (HIK distance followed by a linear kernel). Since these functions and their parameters are independent, the parameters could be learned by defining two separate objective function. The advantage of this learning is the time and memory cost saving and its simple implementation.

The objective of our supervised learning is to determine the optimal values for the parameters, so as to best capture the relationship between image features and scene categories. We optimize the parameters $\lambda, \mu, \beta, \alpha$ by minimizing the error loss in estimating the scene category using the contextual-semantic map function $f(x)$. To solve this optimization problem *l2*-regularized log-loss is employed. During this optimization, non-negativity constraints are applied on λ, μ and β_k parameters to keep the definition of "distances" meaningful.

To measure the error loss, each training image within the training set T is mapped into a real number through map function $f(x)$ defined in (3.9).

We employ the logistic loss function $\phi(m_t)$ for the error-rate measurement:

$$\phi(m_t) = \frac{1}{\ln 2} \ln(1 + e^{-m_t}) \tag{3.12}$$

where

$$m_t = y_t f_{\lambda,\mu,\beta,\alpha}(x_t) \tag{3.13}$$

in which $y_t \in \{-1, 1\}$. Substituting the log-loss with other mainstream loss functions such as hinge-loss and (0-1) loss does not make any significant difference.

The optimal parameter values are obtained by minimizing the standard regularized classification objective, which is consist of the objective loss function with a regularization term R to prevent overfitting:

$$(\lambda, \mu, \beta, \alpha)^* = \arg \min_{(\lambda,\mu,\beta,\alpha)} \sum_{i=1}^{T} \phi(m_t) + R \tag{3.14}$$

The regularization term R is defined like:

$$R = C_g \lambda^2 + C_m \mu^2 + C_b \|\beta\|^2 + C_a \|\alpha\|^2 \tag{3.15}$$

where C_g, C_m, C_b, C_a indicate the amount of regularization that limit the size of the parameters.

Finally, by substituting the map function $f(x)$, the objective function is:

$$\phi(\lambda, \mu, \beta, \alpha) =$$
$$R + \frac{1}{\ln 2} \sum_{i=1}^{n} \ln(1 + e^{-y_t \sum_{i=1}^{p} \alpha_i \exp(-\lambda G_i(x_t) - \mu Ct_i(x_t) - L_i(x_t))}) \tag{3.16}$$

In this equation, the distance functions $G_i(x)$, $Ct_i(x)$ and $L_i(x)$ are based on (3.4), (3.5) and (3.6), while n and p are the sizes of training set and exemplar-set respectively. To obtain the best parameter values subject to minimize our objective function, an iterative gradient-based optimization is applied.

To learn optimal values for the γ, ψ, α_s parameters related to the spatial features, a similar procedure is followed for map function $f_s(x)$. The objective function in this case is:

$$\phi_s(\gamma, \psi, \alpha_s) = R_s + \frac{1}{\ln 2} \sum_{i=1}^{n} \ln(1 + e^{-y_t \sum_{i=1}^{p} \alpha_s i(\gamma S_i(x_t) + \psi P_i(x_t))}) \tag{3.17}$$

where the R_s regularization term is defined as follows:

$$R_s = C_s \gamma^2 + C_p \psi^2 \tag{3.18}$$

3.5.3 Scene Type Recognition for SSFF Method

As shown in figure 3.4, given a test image, features are extracted to form the semantic and spatial models. The values for $D(x)$ and $D_s(x)$ distance functions are then computed and utilized in the map functions $f(x)$ and $f_s(x)$ to get $h(x)$. These are computed with respect to every image in the overall exemplar-set.

As mentioned previously, there is a unique $h(x)$ function for each scene category. Each such function generates an affinity value for image x to the corresponding scene category. The image is then categorized as the class with the highest $h(x)$ value.

3.6 Experimental Results

To demonstrate the efficiency of our feature fusion method, we have evaluated our method on two of the most challenging scene datasets: the *15-Scene* dataset and the *MIT 67-Indoor Scenes*.

15-Scene dataset. This popular database consists of both indoor and outdoor classes(see figure 3.5).

It contains 13 categories first reported in [5] and 2 more categories were added to it by Lazebnik et al. [10]. Eight of the categories are common with the Oliva and Torralba outdoor scene dataset [15] but in gray scale.

MIT 67-Indoor Scenes. This dataset is the most challenging scene dataset (see Fig. 3.6) that contains 67 different indoor scene classes, this dataset is introduced by Oliva and Torralba [19]. To conduct the experiments on this dataset we used the exact same partitions as used in [19] which contain 80 images for training and 20 images for testing, so that all results are directly comparable.

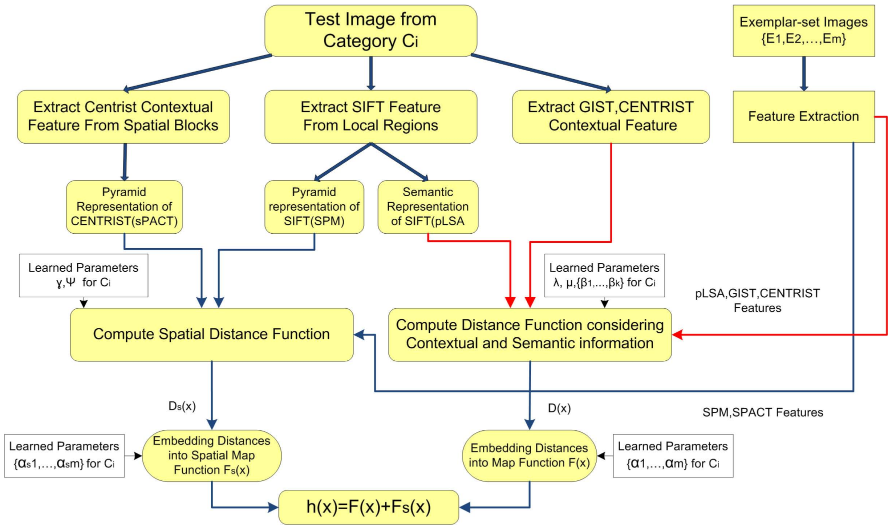

Test Image from Category Ci
Exemplar-set Images {E1,E2,…,Em}
Extract Centrist Contextual Feature From Spatial Blocks
Extract SIFT Feature From Local Regions
Extract GIST,CENTRIST Contextual Feature
Feature Extraction
Pyramid Representation of CENTRIST(sPACT)
Pyramid representation of SIFT(SPM)
Semantic Representation of SIFT(pLSA
Learned Parameters γ,Ψ for Ci
Learned Parameters λ, μ,{β1,...,βk} for Ci
pLSA,GIST,CENTRIST Features
Compute Spatial Distance Function
Compute Distance Function considering Contextual and Semantic information
Ds(x)
D(x)
SPM,SPACT Features
Learned Parameters {αs1,…,αsm} for Ci
Embedding Distances into Spatial Map Function Fs(x)
Embedding Distances into Map Function F(x)
Learned Parameters {α1,…,αm} for Ci
h(x)=F(x)+Fs(x)

Fig. 3.4 Block diagram of Scene Recognition for test input image

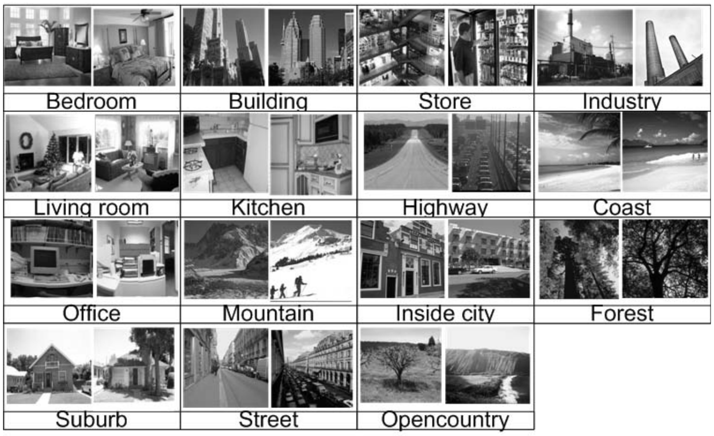


Fig. 3.5 Sample images from the 15-Scene dataset

In both datasets, all of the images are rescaled to 256 × 256 and the codebook size is fixed at 1500 during the experiments. For the 15-Scene dataset, the results were averaged over five times random splitting of training and testing images. For 67-Indoor Scenes, we used the determined split by [19].

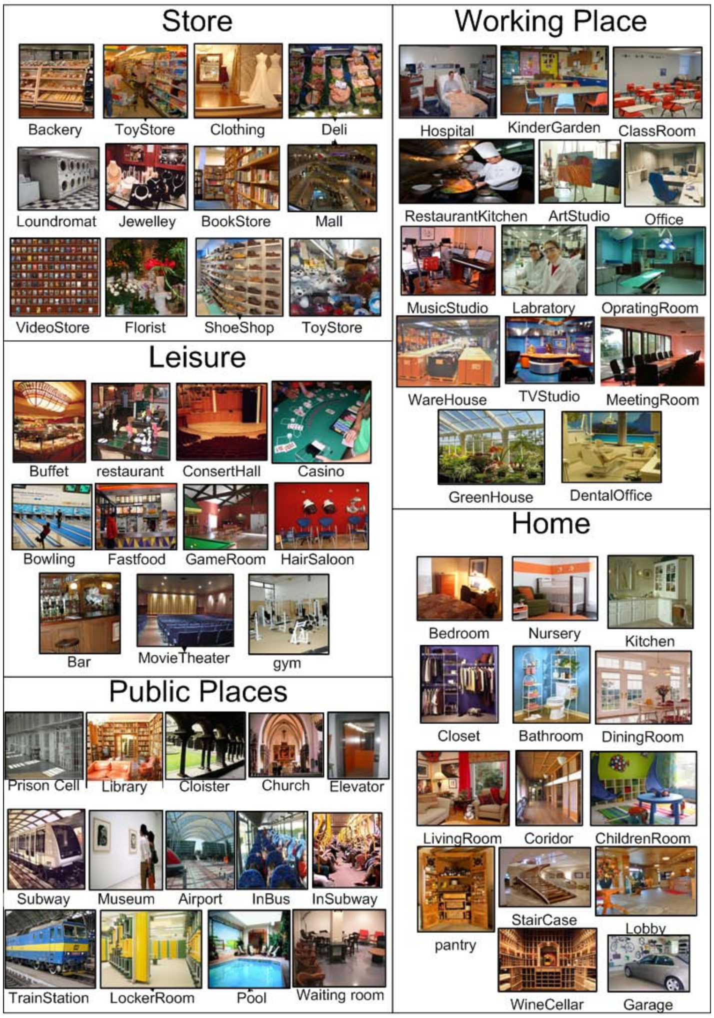


Fig. 3.6 Sample images from the 67-Indoor Scenes dataset

3.6.1 Results on 15-Scene Dataset

The first experiment involves finding the optimal size of the exemplar-set. In this experiment we increased the number of exemplar images, starting with 10 images per category. This experiment is performed on validation-set of 10 randomly selected images of each category. As seen in figure 3.7, the accuracy increases with the number of exemplars until we reach a certain point, after which the accuracy drops due to over-fitting. The number of exemplars at that point is the optimal number. Considering the low variance of the accuracies, there is no reason for fine-tuning of the number of exemplars, hence this number is increased by 10 per step. The optimum

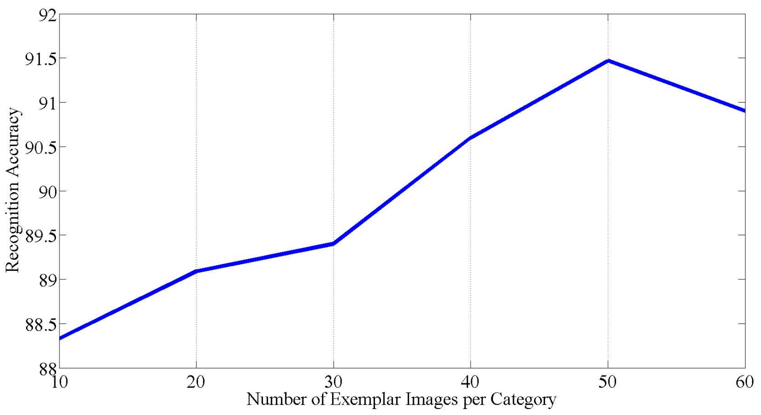


Fig. 3.7 Recognition accuracy vs. exemplar images per category in 15-Scene dataset

number of exemplars for the 15-scene dataset is 50 images per category. Therefore the number of exemplar images to achieve the best accuracy is 50 exemplar images per category. We validate our method on the test data using 50 exemplar per category and obtained the recognition accuracy of 89.4%. The confusion matrix is shown in figure 3.8.

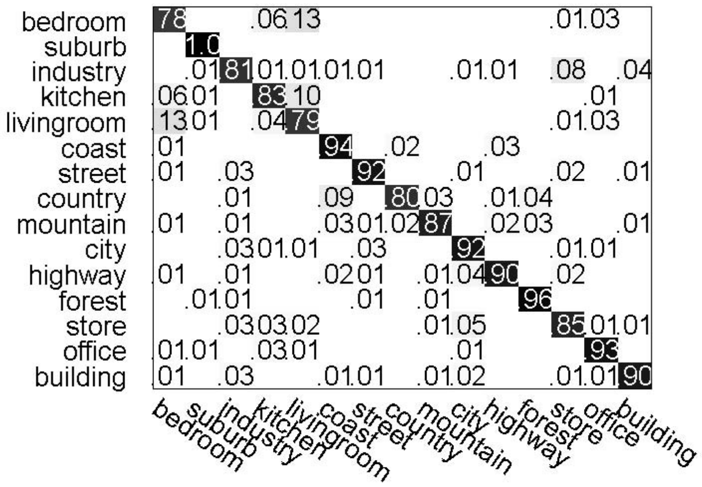


Fig. 3.8 Confusion matrix for the best result in 15-Scene dataset

Table 3.1 Comparison with recently reported results for 15-Scene

Method	Accuracy(%)
SSFF	**89.47**
Xiao et al. [25]	88.1
Wu&Rehg [24]	83.88
Bosch et al. [3]	83.7
Lazebnik et al. [10]	81.4
Li et al. [11]	80.9
Parizi et al. [18]	78.6
Liu&Shah [12]	75.16
Liu et al. [13]	74.9
Oliva&Torralba [15]	74.10
Bosch et al. [2]	73.30
Quelhas et al. [20]	71.24
Fei-Fei&Perona [5]	65.2

Table 3.1 shows the effectiveness of our proposed method by comparing the evaluation results with recently published results on this dataset. As seen in this table, our method achieved the state-of-the-art result of **89.47%** on the 15-Scene dataset. This has verified that our method performs better than proposed MKL method in [25] despite the fact that our method used less number of visual features. The recognition accuracies for all of the features that we embedded into our framework is shown in figure 3.9. Please note that the accuracies shown in this figure are slightly different from the ones reported in the original papers, because in our case, all the images were resized to 256×256.

In [2] the images are considered as a mixture of semantic topics. If followed by SVM classification, the classification rate is lower than our method. Oliva and Torralba [15] and Wu and Rehg [24] use GIST and CENTRIST global features respectively, bypassing object-centered and local information. Although the recognition rates in these holistic methods were higher than those for purely local methods, they were still much lower than our reported results. Capturing the spatial location of local patches in [10] and [24] significantly improved the recognition accuracy for scene recognition. Nevertheless, spatial pyramid models were also less accurate compared to our feature fusion method which comprehensively accounts for semantic, contextual and spatial information within the scene.

3.6.2 Results on MIT 67-Indoor Scenes Dataset

To find the optimal size of the exemplar-set for the 67-Indoor Scenes dataset we performed a similar experiment which increase the number of exemplar images by 10 images per category in each step started from 10 images per category and stopped when the accuracy began to decline. In this experiment the size of validation-set is random splitting of training set for 10 images per category for validation. shown

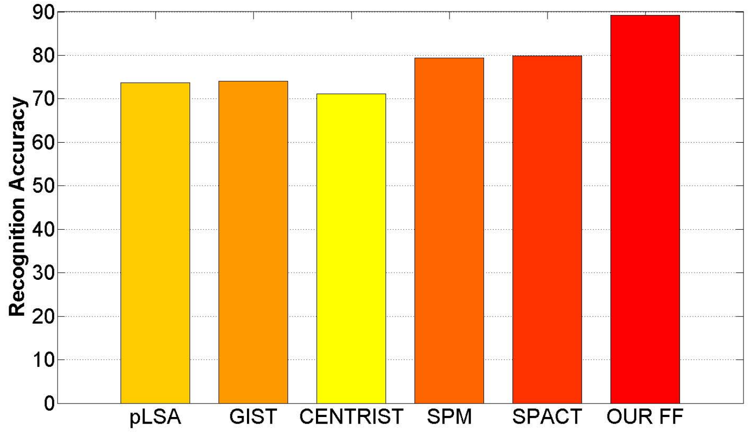


Fig. 3.9 SSFF Recognition rate Vs. Baseline frameworks for 15-Scene dataset

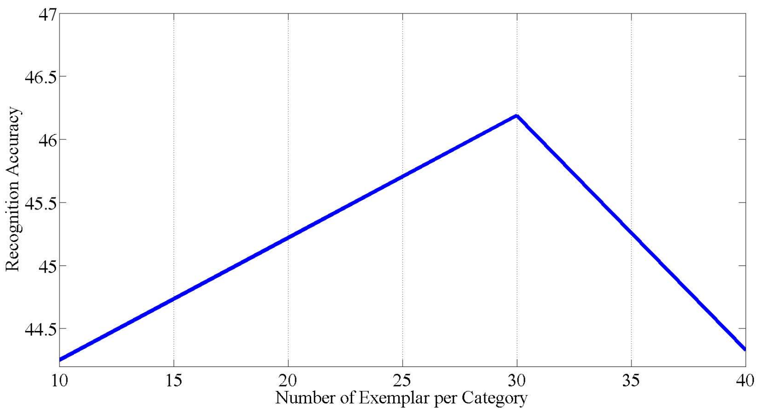


Fig. 3.10 Recognition accuracy vs exemplar images per category for 67-Indoor Scenes

in figure 3.10 the optimal number of exemplars is 30 images per category for the 67-Indoor Scenes dataset. Hence the experiments performed on 67-Indoor Scene dataset while number of exemplars are fixed to 30/category. To show the performance of our feature fusion method, we compared our results with recently reported results on this dataset. As seen in Table 3.2, our method achieved the promising result of **45.0%**. According to this table our FF method(SSFF)'s result is comparable to the state-of-the-art results in 67-Indoor Scenes.

In [17] Pandey and Lazebnik used the popular Deformable Part-based Model(DPM) [16] for scene recognition and achieved the accuracy of **30.08%**.

Table 3.2 Comparison with recently reported results for 67-Indoor Scenes

Method	Accuracy(%)
SSFF	**45.0**
Singh et al. [22]	38.1/49.4
Pandey&Lazebnik [17]	30.08/43.1
Parizi et al. [18]	37.93
Li et al. [11]	37.6
Wu&Rehg [24]	36.9
Lazebnik et al. [10]	34.4
Quattoni&Torralba [19]	26.00
Quelhas et al. [20]	21.17
Oliva&Torralba [15]	22.0
Bosch et al. [2]	20

Subsequently they formed the *color-GIST* feature by applying the GIST descriptor over the color channels and concatenating the GIST vectors together. Eventually the combined use of the color-GIST feature together with SIFT spatial pyramids and DPM results achieved an accuracy of 43.1%.

The discriminative mid-level patches in [22] achieved **38.1%** accuracy while combining these mid-level patches with color-GIST,DPM and SIFT spatial pyramids they obtained **49.4%** recognition accuracy.

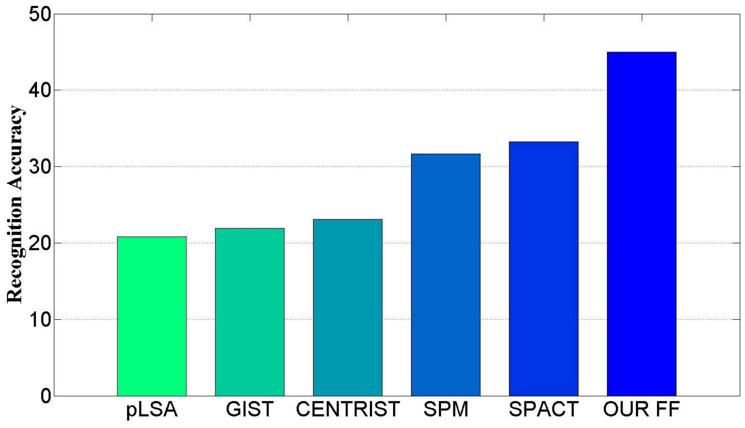


Fig. 3.11 SSFF Recognition rate Vs. Baseline frameworks for 67-Indoor Scenes dataset

The accuracy per class for this dataset is illustrated in Table 3.3.

Table 3.3 Recognition Accuracy per class for 67-Indoor Scenes

	Accuracy		Accuracy		Accuracy		Accuracy
inside airport	10	computer room	44	inside subway	47	pantry	55
art studio	30	concert hall	66	jewelery shop	31	inside pool	75
auditorium	83	corridor	61	kinder garden	30	prison cell	55
bakery	26	deli	5	kitchen	28	restaurant	20
bar	33	dental office	38	laboratory	18	restaurant kitchen	39
bathroom	72	dining room	33	laundromat	22	shoe shop	15
bedroom	23	elevator	76	library	32	stairs case	40
bookstore	15	fast food restaurant	29	living room	20	studio music	63
bowling	75	florist	63	lobby	50	subway	57
buffet	80	game room	15	locker room	38	toy store	27
casino	42	garage	38	mall	50	train station	55
children room	44	green house	95	meeting room	59	tv studio	38
inside church	84	grocery store	23	movie theater	70	video store	27
classroom	55	gym	50	museum	26	waiting room	14
cloister	80	hair salon	23	nursery	60	warehouse	38
closet	83	hospital room	20	office	33	wine cellar	33
clothing store	61	inside bus	78	operating room	26		

3.7 Conclusion

In this paper an innovative method is proposed to integrate global contextual features together with the semantic and spatial information of local features. The features are embedded into a map function based on a novel distance function. A parameter learning model is used to determine the relative importance weights of global information and the latent variables in the latent semantic space. An empirical study has been performed on the 15-Scene and 67-Indoor scenes datasets in order to demonstrate the impact of appropriately incorporating both local and global information for the purpose of scene recognition, with promising results.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
2. Bosch, A., Zisserman, A., Muñoz, X.: Scene classification via pLSA. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part IV. LNCS, vol. 3954, pp. 517–530. Springer, Heidelberg (2006)
3. Bosch, A., Zisserman, A., Muoz, X.: Scene classification using a hybrid Generative/ Discriminative approach. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(4), 712–727 (2008)
4. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: International Workshop on Statistical Learning in Computer Vision (2004)
5. Fei-Fei, L., Perona, P.: A Bayesian hierarchical model for learning natural scene categories. In: IEEE International Conference on Computer Vision and Pattern Recognition, CVPR (2005)
6. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: IEEE International Conference on Computer Vision, ICCV (2005)
7. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning 42(1-2), 177–196 (2001)

8. Jhuo, I.H., Lee, D.T.: Boosted multiple kernel learning for scene category recognition. In: IEEE International Conference on Pattern Recognition, ICPR (2010)
9. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
10. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE International Conference on Computer Vision and Pattern Recognition, CVPR (2006)
11. Li, L.J., Su, H., Xing, E.P., Fei-Fei, L.: Object bank: A high-level image representation for scene classification and semantic feature sparsification. In: Neural Information Processing Systems, NIPS (2010)
12. Liu, J., Shah, M.: Scene modeling using co-clustering. In: IEEE International Conference on Computer Vision, ICCV (2007)
13. Liu, J., Yang, Y., Shah, M.: Learning semantic visual vocabularies using diffusion distance. In: IEEE International Conference on Computer Vision and Pattern Recognition, CVPR (2009)
14. Lowe, D.: Object recognition from local scale-invariant features. In: IEEE International Conference on Computer Vision, ICCV (1999)
15. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision 42(3), 145–175 (2001)
16. Felzenszwalb, P.F., Girshick, R.B., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence (2010)
17. Pandey, M., Lazebnik, S.: Scene recognition and weakly supervised object localization with deformable part-based models. In: IEEE International Conference on Computer Vision, ICCV (2011)
18. Parizi, S., Oberlin, J., Felzenszwalb, P.: Reconfigurable models for scene recognition. In: IEEE International Conference on Computer Vision and Pattern Recognition, CVPR (2012)
19. Quattoni, A., Torralba, A.: Indoor scene recognition. In: IEEE International Conference on Computer Vision and Pattern Recognition, CVPR (2009)
20. Quelhas, P., Monay, F., Odobez, J.M., Gatica-perez, D., Tuytelaars, T., Van Gool, L.: Modeling scenes with local descriptors and latent aspects. In: IEEE International Conference on Computer Vision, ICCV (2005)
21. Saghafi, B., Farahzadeh, E., Rajan, D., Sluzek, A.: Embedding visual words into concept space for action and scene recognition. In: British Machine Vision Conference, BMVC (2010)
22. Singh, S., Gupta, A., Efros, A.A.: Unsupervised discovery of mid-level discriminative patches. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part II. LNCS, vol. 7573, pp. 73–86. Springer, Heidelberg (2012)
23. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their location in images. In: IEEE International Conference on Computer Vision, ICCV (2005)
24. Wu, J., Rehg, J.M.: CENTRIST: A visual descriptor for scene categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(8), 1489–1501 (2011)
25. Xiao, J., Hays, J., Ehinger, K., Oliva, A., Torralba, A.: SUN database: Large-scale scene recognition from abbey to zoo. In: IEEE International Conference on Computer Vision and Pattern Recognition, CVPR (2010)
26. Zabih, R., Wood II, J.: Non-parametric local transforms for computing visual correspondence. In: Eklundh, J.-O. (ed.) ECCV 1994. LNCS, vol. 801, pp. 151–158. Springer, Heidelberg (1994)

Chapter 4
Improving RGB-D Scene Reconstruction Using Rolling Shutter Rectification

Hannes Ovrén, Per-Erik Forssén, and David Törnqvist

Abstract. Scene reconstruction, i.e. the process of creating a 3D representation (mesh) of some real world scene, has recently become easier with the advent of cheap RGB-D sensors (e.g. the Microsoft Kinect).

Many such sensors use rolling shutter cameras, which produce geometrically distorted images when they are moving. To mitigate these *rolling shutter distortions* we propose a method that uses an attached gyroscope to rectify the depth scans. We also present a simple scheme to calibrate the relative pose and time synchronization between the gyro and a rolling shutter RGB-D sensor.

For scene reconstruction we use the Kinect Fusion algorithm to produce meshes. We create meshes from both raw and rectified depth scans, and these are then compared to a ground truth mesh. The types of motion we investigate are: pan, tilt and wobble (shaking) motions.

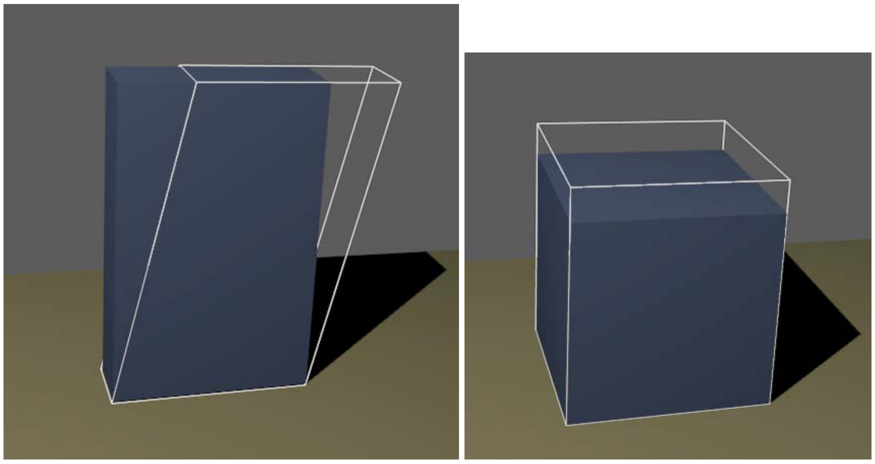
As our method relies on gyroscope readings, the amount of computations required is negligible compared to the cost of running Kinect Fusion.

This chapter is an extension of a paper at the IEEE Workshop on Robot Vision [10]. Compared to that paper, we have improved the rectification to also correct for lens distortion, and use a coarse-to-fine search to find the time shift more quicky. We have extended our experiments to also investigate the effects of lens distortion, and to use more accurate ground truth. The experiments demonstrate that correction of rolling shutter effects yields a larger improvement of the 3D model than correction for lens distortion.

4.1 Introduction

RGB-D sensors, such as the Microsoft Kinect have recently become popular as a means for dense real-time 3D mapping. Dense RGB-D mapping was introduced

Hannes Ovrén · Per-Erik Forssén · David Törnqvist
Department of Electrical Engineering, Linköping University, Sweden
e-mail: `hannes.ovren@liu.se, {perfo,tornqvist}@isy.liu.se`



(a) Pan (left to right) (b) Tilt (upwards)

Fig. 4.1 Synthetic visualization of rolling shutter effects on reconstructed meshes. The solid shape is the correct mesh and the wireframe is the distorted mesh.

in the Kinect Fusion algorithm [9], which is aimed at augmented reality. Recently the Kinect Fusion algorithm has also been adapted to *simultaneous localisation and mapping* (SLAM) [19], and to odometry and obstacle avoidance [14].

As pointed out in [12], RGB-D sensors that use the structured-light sensing principle, e.g. the Kinect, are built using CMOS image sensors with rolling shutters (RS). A sensor with a rolling shutter has a line-by-line exposure, and this will cause geometric distortions in both the colour images and the depth maps from an RGB-D sensor, whenever either the sensor or objects in the scene are moving. Illustrations of the rolling shutter effect on meshes reconstructed using an RGB-D sensor can be found in Figure 4.1.

In this chapter we investigate how the influence of rolling shutter distortions in dense SLAM can be mitigated, by equipping an RGB-D sensor with a 3-axis MEMS gyro sensor. Gyro sensors for consumer electronics are inexpensive and can provide angular velocity measurements at rates well above most camera frame rates. The use of a gyro sensor means that the extra computational burden of optical flow, and non-linear optimisation used for RS rectification in [12] need not be added to the already high cost of SLAM computation. An additional benefit is that the gyro provides angular velocity also in scenes with low texture, where optical flow computation is difficult.

Our experiments use the Kinect Fusion algorithm [9] to do scene reconstruction. Kinect Fusion has gained popularity in part due to the open source KinFu implementation in PCL [15]. We characterize the situations where RS distortions occur, and investigate to what extent a rotation-based RS rectification can improve the output of Kinect Fusion. We also investigate the effects of removing the Kinect lens distortion, and compare the improvement of lens distortion correction to that of rolling shutter correction.

The source code for the tools used to rectify the depth scans will be made available on the author's website.

4.1.1 Related Work

As our system makes use of an external gyroscope sensor, we require the clocks of the camera and the gyroscope to be synchronized, and their relative pose to be known. Several methods for camera-IMU calibration have been proposed in the past. A recent example is [7]. Such methods typically compute the full relative pose between IMU and camera (both translation and rotation). In our approach we have instead chosen to introduce a new calibration scheme, for two reasons: Firstly, all current algorithms assume global shutter geometry, which means that their application to a rolling shutter camera needs to be done with care. Secondly, when only gyro measurements are to be used, the translation component is not needed, and this allows us to simplify the calibration considerably.

Our time synchronization procedure uses the same cost-function as [5, 8]. Just like [5] we only search for the time shift, but instead of performing gridding on the cost-function, we solve for the time shift in two steps: Firstly we find a coarse alignment using correlation of the device motion function, secondly we refine this estimate using derivative free search. In contrast to [5, 8], our method also deals with finding the unknown time scaling factor.

A dataset for evaluation of RGB-D SLAM accuracy was recently introduced by TU Münich and University of Freiburg [18]. Evaluation using this dataset consists of comparing a camera motion trajectory against ground truth from a motion capture system. As we rely on gyro measurements, we cannot use these datasets, and instead we demonstrate the effectiveness of our algorithm by comparing obtained 3D models with, and without applying our depth-map rectification.

The rolling shutter problem has been extensively studied in the past [3, 13, 1, 12]. Most closely related to our work is [12], on which we base our rectification scheme. In [12], the RGB-D device motion is computed using a sparse optical flow that is obtained from Kinect NIR images. Instead of using optical flow, we rectify the depth maps using the angular velocity provided by a 3-axis MEMS gyro sensor. This makes the resultant system more robust, as we can easily deal with two cases that are challenging for optical flow based techniques: 1. Scenes with large untextured regions 2. Scenes where the amount of ambient light present in the scene is too low.

4.1.2 Structure

This chapter is divided into three parts: Section 4.2 describes the gyro and rolling shutter camera calibration. Section 4.3 describes our approach to depth map rolling shutter rectification. In Section 4.4 we perform a number of experiments that show the effect of rolling shutter rectification for RGB-D cameras.

4.1.3 Notation

We use superscripts to denote the used frame of reference where needed. t^g and t^c denote time in the gyro and RGB-D camera frames of reference respectively. A relation between frames is expressed as combinations, e.g. $\mathbf{R}^{cg}$ is the rotation from the gyro frame to the camera frame of reference. Vectors and matrices are expressed in bold ($\mathbf{x}$, $\boldsymbol{\Omega}$).

4.2 Sensor Calibration

The gyro provides angular velocity measurements for each of its three axes as the angular velocity vector $\omega(t^g) = (\omega_x, \omega_y, \omega_z)$. The RGB-D camera provides us with RGB images $\mathbf{I}(t^c)$ and depth images $\mathbf{D}(t^c)$.

To associate the data from the two sensors we must know the relation between their timestamps, t^g and t^c, as well as the relative pose, $\mathbf{R}^{gc}$, between their coordinate frames.

Our proposed pose calibration method only requires that the combined sensors are rotated at least once around two non-parallel axes. The time synchronization method only requires that the observed motion is not periodic, since it is based on correlation.

4.2.1 Synchronizing the Timestamps

Assuming that both timestamp generators provide timestamps that are linear in time, the two timestamps will be related via a linear function

$$t^g = m^{gc} \cdot t^c + d^g \,. \tag{4.1}$$

The multiplier m^{gc} will be constant when both timestamp generators are stable and do not drift. The time offset d^g depends on when each timestamp generator was initialized. Typically reinitialization can occur at any time due to e.g. a hardware reset, which makes it necessary to recompute d^g for every experiment.

Although the multiplier has to be known in order to calculate the offset, we will start by describing how to calculate the latter.

4.2.1.1 Finding the Offset

Since the offset, d^g, has to be computed for every experiment it should be fast to compute. To achieve this we divide the task into first finding a rough estimate which is then refined.

The rough offset is found based on the assumption that a rotation of the sensor platform will in some way be observable by both sensors. For the gyro this is trivial. For the RGB-D sensor we assume that the rotation is manifested in the optical flow magnitude between consecutive frames.

We begin by defining the gyro speed at sample n as

$$W(n) = \|\omega_n\| . \tag{4.2}$$

For convenience we will sometimes refer to $W(n)$ as the continuous function $W(t^g)$ where t^g is a timestamp expressed in the gyro time frame. When necessary this implies interpolation of $W(n)$.

The optical flow displacement magnitude for frame j is calculated as

$$F(j) = \frac{1}{N}\sum_{n}^{N} \|\mathbf{x}_{n,j} - \mathbf{x}_{n,j+1}\| , \quad \text{for} \quad j \in [1, J-1] , \tag{4.3}$$

which is the average optical flow at the frame j, where N is the total number of points tracked between frames j and $j+1$, and $\mathbf{x}_{n,j}$ are the image coordinates of the tracked point n in frame j. Like with the gyro speed, we define a continuous version of the flow magnitude, $F(t^c)$.

The assumed proportionality between $F(t^c)$ and $W(t^g)$, after mapping through (4.1), then becomes

$$W(t^g) \underset{\sim}{\propto} F\left(\frac{t^g - d^g}{m^{gc}}\right) . \tag{4.4}$$

Note that as the flow $F(j)$ is computed between frames, it is on average offset by half a frame compared to the image coordinate times. This offset is implicitly handled by the refinement step and can be safely ignored.

After applying the multiplier m^{gc} and resampling the signal with the lowest sampling rate to match the other, we can use cross-correlation to find the offset d^g.

An example of the proportionality between the optical flow magnitude and gyro speed can be seen in Figure 4.2.

It is important to note that the signal used for correlation must not be periodic, as this could make the cross-correlation fail. If we start the data collection from both sensors at approximately the same time, we can extract slices of the original signals which are known to contain a suitable motion.

4.2.1.2 Pyramid Speedup

The correlation that finds the offset d^g can be sped up by orders of magnitude, by using a coarse-to-fine search. This has the added benefit of allowing a larger set of samples to use for correlation. By using a larger part of the signal, the need for a dedicated synchronization movement decreases. We do still have to be careful as the non-periodicity constraint must still be met.

To do the course-to-fine search, we first successively sub-sample $W(n)$ and $F(j)$, by averaging neighbouring pairs of samples. This yields two scale pyramids $\{F_k\}_0^K$, and, $\{W_k\}_0^K$, where:

$$F_k(j) = 0.5(F_{k-1}(2j) + F_{k-1}(2j+1)) \text{ and } F_0(j) = F(j) . \tag{4.5}$$

The pyramid for $\{W_k\}_0^K$ is computed in the same way. We stop generating higher levels in the pyramids whenever the length of either F_k or W_k drops below 100. We then proceed by finding the offset at the coarsest scale by full correlation. As we only have about 75 samples in each sequence, this is very fast.

An offset at a coarse scale, d_k, can be converted to an offset at the next finer scale using the expression $d_{k-1} = 2d_k + 0.5$. To propagate the result through the pyramid, we use this as an initial guess, which we refine at each successive scale by trying a few neighbouring offsets. We use the range $d_{k-1} \in [2d_k - 1, 2d_k + 2]$, and we have found that in practise this always finds the same final offset as a full search at the finest scale.

Using the pyramid approach described above also allows us to replace plain correlation with the more expensive, but also more robust *zero-mean normalised cross correlation* (ZNCC).

4.2.1.3 Refining the Time Offset

The correlation-based time offset was found to be accurate to about ± 2 frames. Since we need sub-frame accuracy, the time offset must be refined further.

In [5] Hanning et al. describe a method to find an unknown offset between image timestamps and gyro timestamps. Points are tracked through an image sequence, and a grid search is used to find the time offset that best removes the rolling shutter effects.

Since we know that the offset is off by at most a few frames, our starting guess will be in the convex basin around the minimum of the cost function. This allows us to replace grid search with the much more efficient Brent's method [11].

4.2.1.4 Finding the Multiplier

To find the multiplier, the sensor platform is kept still except for two short and distinct movements, where the second movement is delayed sufficiently long.

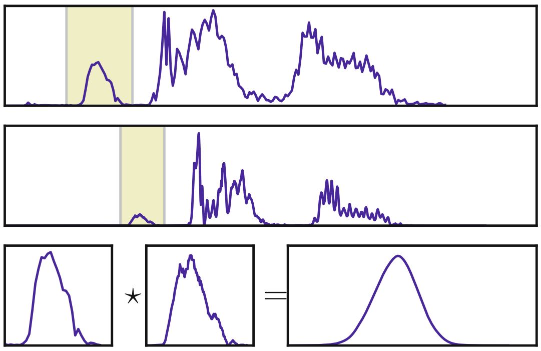


Fig. 4.2 Optical flow and gyro speed comparison. From top to bottom: Optical flow displacement magnitude $F(t^c)$, gyro angular speed $W(t^g)$, and correlation response. Correlation is calculated from slices known to contain the synchronization movement (highlighted).

We generate two short rotations of the sensor platform which will both be observable in both $W(t^g)$ and $F(t_j^c)$. The time between the two rotations, T, is measured in each sensor's frame of reference and the multiplier is calculated as the average of N such sequences

$$m^{gc} = \frac{1}{N}\sum_{n}^{N}\frac{T_n^g}{T_n^c}. \tag{4.6}$$

The calculated multiplier is then assumed to be valid for sequences of at least length T.

4.2.2 Relation of Coordinate Frames

No matter how carefully the IMU and camera are joined there will likely be some alignment error which could disturb the results [7].

A relative pose consists of a rotation and a translation from one coordinate frame to another. For our implementation only the gyro is used so the translation is not needed.

The basic idea of the relative pose estimation is that if we have two or more orientation vectors in one coordinate frame, and corresponding orientation vectors in the other coordinate frame we can find uniquely the rotation between them.

By rotating the sensor platform around at least two non-parallel axes we find the axes of rotation as seen by the camera coordinate frame and the IMU coordinate frame.

4.2.2.1 Gyro Coordinate Frame

Given a sequence where the gyro is rotating, we want to find the axis of rotation $\hat{\mathbf{r}}$. We do this by defining the following maximization problem:

$$\hat{\mathbf{r}} = \arg\max_{\mathbf{r}} J(\mathbf{r}) \tag{4.7}$$

$$J(\mathbf{r}) = \sum_{n=1}^{N} ||\mathbf{r}^T \omega_n||^2 \tag{4.8}$$

$$J(\mathbf{r}) = \sum_{n=1}^{N} \mathbf{r}^T \omega_n \omega_n^T \mathbf{r} = \mathbf{r}^T \underbrace{\left(\sum_{n=1}^{N} \omega_n \omega_n^T\right)}_{\Omega} \mathbf{r} \tag{4.9}$$

Here N is the total number of gyro samples in the chosen sequence.

The rationale of this cost function is that the principal axis of rotation should be parallel to ω, and large velocities should have a larger influence on the result. The solution $\hat{\mathbf{r}}$ is the eigenvector of Ω with the largest eigenvalue.

We are however not certain if we have found $\hat{\mathbf{r}}$ or $-\hat{\mathbf{r}}$ as both will give the same cost. The sign can be determined by testing whether the scalar product between the acquired $\hat{\mathbf{r}}$ and all ω_n is positive or negative such that

$$\hat{\mathbf{r}} \leftarrow \operatorname{sgn}\left(\sum_{n=1}^{N} \hat{\mathbf{r}}^T \omega_n\right) \hat{\mathbf{r}} . \tag{4.10}$$

4.2.2.2 Camera Coordinate Frame

The problem of finding the rotation axes of the camera can be formulated as an *Orthogonal Procrustes problem* [16, 4]. For a given rotation sequence we track a number of points from the first image frame to the last. To make sure the resulting point correspondences are of high quality the points are retracked from the last image to the first. Points are discarded if the distance from the original point is larger than 0.5 pixels. It is very important that the first and last frame of the sequence are captured when the sensor platform is not moving, otherwise rolling shutter effects would bias the result.

Using the camera calibration matrix $\mathbf{K}$, and a depth map $z(u,v)$ the 2D points are back projected to 3D using the equation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = z(u,v)\mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} . \tag{4.11}$$

If we denote the set of 3D points from the first and last image $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ...)$ and $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, ...)$ respectively, the problem of aligning them can be formulated as

$$\arg\min_{\mathbf{R},\mathbf{t}} ||\mathbf{X} - (\mathbf{R}\mathbf{Y} + \mathbf{t})||^2 \quad \text{s.t.} \quad \mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{4.12}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation.

Procrustes now gives us an estimate of $\mathbf{R}$ and $\mathbf{t}$ using the SVD

$$\mathbf{U}\mathbf{D}\mathbf{V}^T = \mathrm{SVD}[(\mathbf{X} - \mu_X)(\mathbf{Y} - \mu_Y)^T] \tag{4.13}$$

$$\mathbf{R} = \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U}\mathbf{V}^T) \end{pmatrix} \mathbf{V}^T \tag{4.14}$$

$$\mathbf{t} = \mu_X - \mathbf{R}\mu_Y . \tag{4.15}$$

Here μ_X and μ_Y are the means of the vectors $\mathbf{X}$ and $\mathbf{Y}$ respectively.

Although the Procrustes solution provides us with both rotation and translation, only the rotation is needed in our implementation.

The rotation matrix $\mathbf{R}$ can be written on axis-angle form as $\varphi\hat{\mathbf{n}}$ where $\hat{\mathbf{n}}$ is the principal axis of the rotation that we want to find.

To convert from matrix form to axis-angle form we use the method described by Hartley and Zisserman [6]

$$2\cos\varphi = \mathrm{trace}(\mathbf{R}) - 1 \tag{4.16}$$

$$2\sin\varphi\hat{\mathbf{n}} = \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix} . \tag{4.17}$$

It is worth noting that this representation forces the rotation angle φ to only take positive values. Therefore, if we have two rotations about the same axis $\varphi_1\hat{\mathbf{n}}$ and $\varphi_2\hat{\mathbf{n}}$ where $\varphi_2 = -\varphi_1$ we would measure the latter as $\varphi_1(-\hat{\mathbf{n}})$ which is a positive angle and a flipped rotation axis. This makes the result consistent with the behaviour of the gyro rotation axis calculation in (4.10).

4.2.2.3 Calculating the Relative Pose

Using the methods in sections 4.2.2.1 and 4.2.2.2 we can collect a set of corresponding rotation axes, $\mathbf{X}^g$ and $\mathbf{X}^c$, in the gyro coordinate frame and RGB-D camera coordinate frame respectively.

Once again we can formulate an Orthogonal Procrustes's Problem to find the relative sensor pose

$$\begin{aligned} \mathbf{R}^{cg}, \mathbf{t}^{cg} &= \arg\min_{\mathbf{R},\mathbf{t}} \|\mathbf{X}^c - (\mathbf{R}\mathbf{X}^g + \mathbf{t})\|^2 \\ \text{s.t.} \quad & \mathbf{R}\mathbf{R}^T = \mathbf{I} \end{aligned} \tag{4.18}$$

We typically use two forward-backward sequences, along two approximately orthogonal axes. This gives us four measurements in total, from which the relative pose may be determined.

4.3 Depth Map Rectification

4.3.1 Gyro Integration

We obtain the rotation of the sensor platform (relative to some initial orientation) by integrating the gyro angular velocity measurements. This accumulated rotation is later used for depth map rectification.

The accumulated rotation at time t is denoted by the unit quaternion

$$\mathbf{q}(t) = [\cos\frac{\varphi}{2}; \sin\frac{\varphi}{2}\mathbf{n}], \tag{4.19}$$

where the unit vector $\mathbf{n}$ is the axis of rotation and φ is the magnitude of the rotation.

Using the timestep Δt and angular velocity measurements ω, the integration becomes

$$\mathbf{q}(0) = [1; \mathbf{0}] \quad \text{and} \tag{4.20}$$

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t) \odot \mathbf{w}(\omega; \Delta t), \quad \text{where} \tag{4.21}$$

$$\mathbf{w}(\omega; \Delta t) = \left[\cos\left(\frac{\|\omega\|\Delta t}{2}\right); \frac{\sin\left(\frac{\|\omega\|\Delta t}{2}\right)}{\|\omega\|}\omega \right]. \tag{4.22}$$

Here $\odot$ is the quaternion multiplication operator. $\mathbf{q}$ is kept as a unit quaternion by renormalizing after each step.

4.3.2 *Rectification*

Once the camera and IMU are synchronized and their relative pose is known, we can perform the rectification. We are only measuring rotations, so the update equation for image coordinates from [12] simplifies to

$$\mathbf{x}' = \mathbf{K}\mathbf{R}(t_{\text{mid}})\mathbf{R}^T(t_{\text{row}})\mathbf{K}^{-1}\phi(\mathbf{x}; \lambda), \quad \text{where} \tag{4.23}$$

$$t_{\text{row}} = t_0 + t_r \frac{x_2}{N_{\text{rows}}} \quad \text{and} \tag{4.24}$$

$$t_{\text{mid}} = t_0 + \frac{t_r}{2}. \tag{4.25}$$

Here t_{row} and t_{mid} are the times when the current row and middle row were captured given the start of frame time, t_0, and readout time, t_r. $\mathbf{R}(t)$ is the rotation of the camera at time t, which is constructed as $\mathbf{R}(t) = \mathbf{R}^{cg}\mathbf{M}(\mathbf{q}(t))$, where

$$\mathbf{M}(\mathbf{q}) = \begin{pmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{pmatrix}, \tag{4.26}$$

transforms a unit quaternion to a rotation matrix [17]. $\mathbf{x}$ and $\mathbf{x}'$ are homogenous image coordinates before and after rolling shutter rectification. $\phi(\mathbf{x}; \lambda)$ is a function that removes lens distortion given the lens distortion parameters λ.

A way to interpret (4.23) is that a 2D point is back-projected to a 3D point, which is rotated back to the initial camera position, then rotated back to the time the middle row was captured, and finally projected again to a 2D point.

The inevitable drift accumulated in $\mathbf{q}(t)$ in previous frames is effectively cancelled out, as the combined rotation in (4.23) is relative to the middle row, and not to the start of the sequence. For online applications you can avoid integrating over the entire sequence, and instead calculate the rotation $\Delta\mathbf{R} = \mathbf{R}(t_{\text{mid}})\mathbf{R}^T(t_{\text{row}})$ using only the necessary samples.

Note that the fact that we are neglecting translations also means that the depth $z(\mathbf{x})$ can be ignored in the projections as it is now simply a scale factor in a homogenous equation.

Using (4.23) we can construct a forward mapping for each pixel coordinate in the original image. We use this to forward interpolate new rectified depth images.

In order to propagate also the the valid/invalid status of pixels each depth image is interpolated twice. First an interpolation using a weighted gaussian 3x3 kernel is used on all pixels that have valid depth values. Second we interpolate using nearest neighbour interpolation all pixels with invalid depth values. This avoids having invalid pixels become valid due to interpolation.

4.4 Experiments

We demonstrate the gain of using our method by examining three different types of sensor motions. These are *pan*, *tilt*, and *wobble* motions. By wobble we mean shaking the sensor to simulate the kind of motion you can expect if the sensor is, for example, handheld or attached to a moving vehicle. Another basic motion which we do not consider here, is *roll*. Our method can handle roll motions, but measuring and visualizing the effect is difficult.

As our method neglects translations, rolling shutter effects from translations are not handled. However, these effects appear only when the image plane is moving at high speeds which makes the impact much weaker than rotational movements.

We divide the three types of motions in two groups with two different evaluation strategies. For pan and tilt we measure features in the resulting meshes and compare to a *ground truth mesh*. In the case of wobble we instead make a visual evaluation of the resulting meshes.

For the pan and tilt experiments we also investigate the effect of compensating for lens distortion.

For each experiment we first recorded a scene while our sensor platform (RGB-D camera and gyroscope) was moving. The depth scans were then rectified to compensate for rolling shutter and/or lens distortion. For each set of data, the scene was reconstructed to a 3D mesh on which comparisons and evaluation was performed. To make time synchronization simpler, we began each recording by performing a short panning motion. This synchronization motion was cut from the recorded data before being used for reconstruction.

4.4.1 Experiment Setup

Our sensor platform consists of one Kinect RGB-D camera, to which is attached an ArduIMU gyro and accelerometer. The Kinect captures depth images at 29.97 Hz and RGB images at 30 Hz, while the ArduIMU provides gyroscope measurements at approximately 170 Hz. The discrepancy between the frame rates of the RGB and depth camera can safely be ignored during 3D reconstruction, since only depth images are used here. To capture the Kinect data we wrote a data logging application that makes sure that no frames are skipped and also uses the raw timestamps from the clock on board the Kinect. Using the raw timestamps is important, as it ensures that the timestamps of the depth images are consistent and not distorted by e.g. latencies due to the USB-transfer. It also has the added benefit that the depth and the RGB timestamps are in the same frame of time. This means we only have to synchronize using one of the cameras, because that result will be valid also for the second camera. The Kinect timestamps are 32-bit unsigned integers generated by a 60 MHz clock.

The intrinsic camera parameters and lens distortion parameters where calibrated using the method described in [20]. We chose to correct only for the radial lens distortion (three parameters) and ignored tangential distortions. To carry out the calibration we used the *calibrateCamera* function in OpenCV.

To construct the meshes from the depth scans we used the implementation of Kinect Fusion that is available in the Point Cloud Library under the name *KinFu* [15]. The reconstruction was run in offline mode, which means that every available frame of the recorded Kinect data was used. This means we get the same result (same mesh) every time we run KinFu. A small modification was made to make KinFu use our intrinsic camera parameters, instead of the default parameters.

4.4.2 Pan and Tilt Distortions

The scene we used is pictured in Figure 4.3. It consisted of a flat desk with objects of different sizes. Screens were placed on the edges of the scene to avoid background clutter.

With pan and tilt motions we expect the meshes to be distorted in a deterministic way. For panning motions we expect objects to become slanted, and for tilt we expect them to become either elongated or shortened. This is visualized in Figure 4.1. To measure these effects we looked at the reconstructed mesh of the large black box. The slant was measured by looking at the upper angles of the box, and the elongation by measuring its height. As ground truth we imaged the scene with a very slow motion of the sensor platform, to avoid rolling shutter effects, and repeated the same type of measurements. The ground truth measurements are available in Table 4.1a, for both original and lens corrected data.

The measurements on the reconstructed mesh (two upper angles and height of the box) were made in the following way: First we fit a plane to the front of the

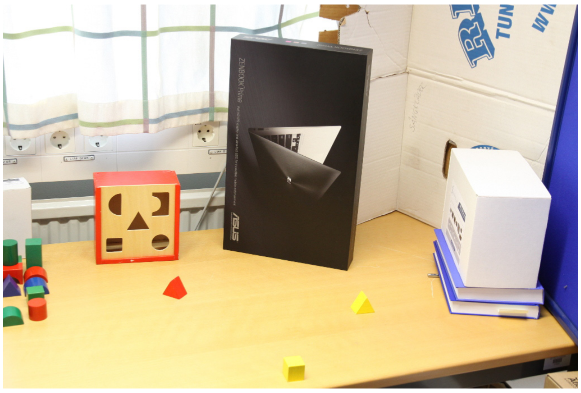


Fig. 4.3 Scene used for experiments. The large black box was used for measurements.

box using RANSAC. The user then clicks at the corner points of the box, and the corresponding points on the front plane are chosen. This plane restriction scheme is helpful since the actual corner is not always present in the mesh due to mesh distortions. Since clicking on a visualization of the mesh is prone to error, each measurement was performed five times. We then take the mean as the actual measurement, with the standard deviation providing some information about the size of the measurement error.

We did three pan and two tilt motion experiments to see how the results differ depending on the angular speed. For pan motions the sensor platform was panned from left to right. For tilt motions it was tilted from down to up. The recorded data was then processed to produce four sets:

1. Original set (no lens correction, no RS correction)
2. Lens correction only
3. RS correction only
4. RS and lens correction

4.4.2.1 Pan and Tilt Results

The results for the pan and tilt experiments can be found in Tables 4.1b and 4.1c respectively. Looking at the measurements it is clear that as the angular speed increases, so does the slant and height of the box. We can also clearly see that our method manages to rectify the mesh to a satisfactory result. Note that at angular speeds of about 2 rad/sec and above, the reconstructed meshes are bad due to large amounts of motion blur, which explains the overall bad result for the second angle in the last pan experiment.

Looking at the impact of correction of lens distortion, we can see that there is no clear trend. The reason for this is that, except for the experiment with the highest pan speed, any positive effects of lens distortion correction are smaller than the accuracy of our ground truth.

In Figure 4.4 we visualize the result of one pan motion experiment. The original and rectified meshes were aligned, and the outline of each mesh was drawn on top of the other mesh. Figure 4.5 shows a tilt experiment where you can see how the rectification effectively shrinks the mesh to get closer to its true size. To align the meshes we used the iterative closest point algorithm (ICP) [2]. The meshes are not

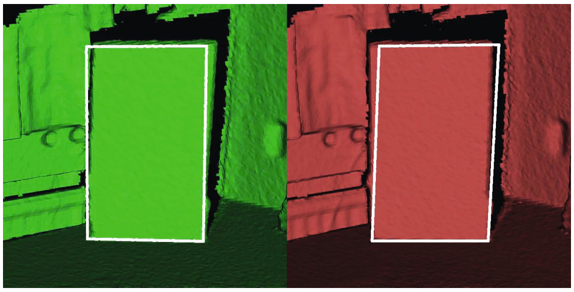


Fig. 4.4 Visualization of one pan experiment. The green mesh is from the original scans, and the red from rolling shutter rectified scans. The white border is the outline of the other mesh. The slant produced by rolling shutter is clearly visible.

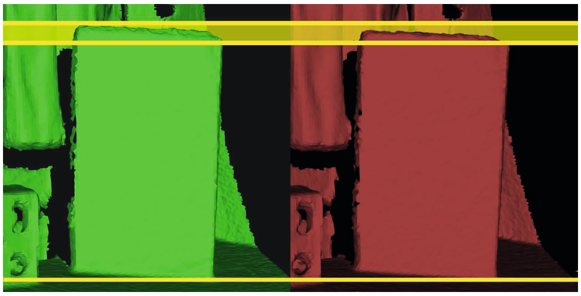


Fig. 4.5 Visualization of rectification on tilt motion experiment to show that the rectified mesh (red) shrinks compared to the original mesh (green).

Table 4.1 Measurements for different angular speeds (in rad/sec). The raw data is expressed as $\mu \pm \sigma$ where μ is the mean and σ is the standard deviation of the measurement. The error columns of tables (b) and (c) are deviations from the ground truths in table (a). Errors marked in bold are the smallest errors for a particular experiment.

(a)

	Original	Lens corrected
Pan	$89.8° \pm 0.1$	$89.4° \pm 0.1$
	$90.1° \pm 0.1$	$90.4° \pm 0.2$
Tilt	42.8 ± 0.1	43.0 ± 0.1

(b)

$\|\omega\|$	Original	Error	Lens only	Error	RS only	Error	RS + lens	Error
0.5	$89.2° \pm 0.2$	$-0.6°$	$88.9° \pm 0.4$	$-0.6°$	$90.0° \pm 0.2$	$0.2°$	$89.6° \pm 0.2$	**$0.1°$**
	$91.4° \pm 0.2$	$1.3°$	$90.8° \pm 0.4$	$0.4°$	$90.2° \pm 0.1$	**$0.2°$**	$90.7° \pm 0.2$	$0.3°$
1.1	$88.5° \pm 0.1$	$-1.4°$	$88.1° \pm 0.2$	$-1.4°$	$90.0° \pm 0.2$	**$0.2°$**	$89.8° \pm 0.4$	$0.4°$
	$91.8° \pm 0.2$	$1.7°$	$92.2° \pm 0.2$	$1.8°$	$89.8° \pm 0.3$	**$-0.2°$**	$90.1° \pm 0.4$	$-0.3°$
2.5	$86.9° \pm 0.3$	$-2.9°$	$86.3° \pm 0.7$	$-3.1°$	$90.5° \pm 0.2$	**$0.7°$**	$90.2° \pm 0.5$	**$0.7°$**
	$98.0° \pm 0.2$	$8.0°$	$97.7° \pm 0.6$	$7.3°$	$94.2° \pm 0.2$	$4.2°$	$94.1° \pm 0.2$	**$3.7°$**

(c) Tilt measurements. Height of rectangular box, measured in cm.

$\|\omega\|$	Original	Error	Lens only	Error	RS only	Error	RS + lens	Error
0.7	43.8 ± 0.2	1.0	44.2 ± 0.1	1.2	43.0 ± 0.0	**0.2**	43.4 ± 0.1	0.4
1.3	43.6 ± 0.3	0.8	42.5 ± 0.1	-0.5	42.5 ± 0.2	-0.3	43.2 ± 0.4	**0.1**

related through a simple rigid transformation, so the entire mesh can not be used for ICP alignment. Since our scene had a flat ground surface we instead opted to align the meshes such that this ground plane was aligned as well as possible. We selected points on the desk surface and close to distinct objects in both meshes, and applied ICP to this smaller point set.

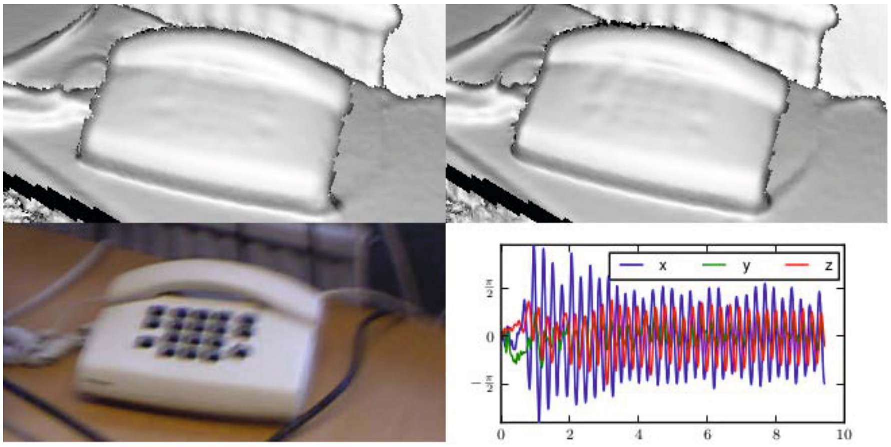


Fig. 4.6 Wobble experiment. Zoom in on raycasted mesh. Top row: Original mesh (left), rectified mesh (right). Bottom row: one selected RGB image from the sequence (left), and gyro measurements (right).

4.4.3 Wobble Distortions

The wobble experiment was carried out by keeping an object (in our case, a telephone) in view of the sensor while shaking the sensor platform. In contrast to the pan and tilt experiment, with wobble we do not expect the general shape of the objects in the scene to change. However, since the Kinect Fusion algorithm integrates measurements over time, we do expect rolling shutter wobble to blur out smaller details. In Figure 4.6 we show a zoomed in view of the telephone, and one can see that e.g. the buttons and cables are more pronounced in the rectified version than in the original version.

We examined the frequency of the wobble by applying the FFT to each axis, and calculated a conservative estimate of the combined frequency content as

$$G(f) = \sqrt{|\omega_x(f)|^2 + |\omega_y(f)|^2 + |\omega_z(f)|^2}\,. \tag{4.27}$$

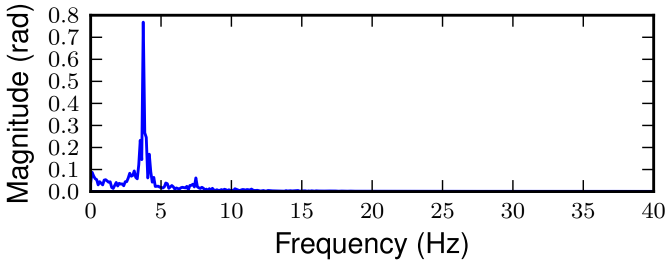


Fig. 4.7 Composite frequency content, $G(f)$, during the wobble experiment

The result in Figure 4.7 shows that the frequency of our handheld wobble was approximately 4 Hz. We can also see that the energy beyond 15 Hz is negligible, which implies that our sampling rate of 170 Hz is sufficient.

4.5 Concluding Remarks

In this chapter we have shown that the rolling shutter effect will create notable errors in 3D reconstructions from the Kinect Fusion algorithm. We also show that these errors can be mitigated by applying rolling shutter rectification on the depth data before 3D reconstruction.

As for lens distortion, we could see no clear improvement when using it. The lens distortion on the Kinect is however quite small. It is likely that even though lens distortion correction will improve the results, our ground truth measurements are simply not accurate enough to see such small improvements.

We have also introduced a simple scheme for calibrating the time synhronization and relative orientation between a gyroscope and a rolling shutter RGB-D sensor.

In the future we would like to improve the depth map rectification scheme. Our current approach, while avoiding interpolation of bad depth values, sometimes produces jagged edges due to nearest neighbour interpolation.

We would also like the to investigate the possibility of performing the sensor synchronization and calibration in a less constrained fashion, without the need for special synchronization motions.

Acknowledgements. This work was supported by the Swedish Research Council through a grant for the project *Embodied Visual Object Recognition*, and by Linköping University.

References

1. Baker, S., Bennett, E., Kang, S.B., Szeliski, R.: Removing rolling shutter wobble. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, San Francisco (2010)
2. Besl, P., McKay, H.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2), 239–256 (1992)
3. Geyer, C., Meingast, M., Sastry, S.: Geometric models of rolling-shutter cameras. In: 6th OmniVis WS (2005)
4. Golub, G.H., van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore (1983)
5. Hanning, G., Forslöw, N., Forssén, P.E., Ringaby, E., Törnqvist, D., Callmer, J.: Stabilizing cell phone video using inertial measurement sensors. In: The Second IEEE International Workshop on Mobile Vision. IEEE, Barcelona (2011)
6. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004)
7. Hol, J.D., Schön, T.B., Gustafsson, F.: Modeling and calibration of inertial and vision sensors. International Journal of Robotics Research 29(2), 231–244 (2010)

8. Karpenko, A., Jacobs, D., Baek, J., Levoy, M.: Digital video stabilization and rolling shutter correction using gyroscopes. Tech. Rep. CSTR 2011-03, Stanford University Computer Science (2011)
9. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011, Basel, Switzerland (2011)
10. Ovrén, H., Forssén, P.E., Törnqvist, D.: Why would i want a gyroscope on my RGB-D sensor? In: Proceedings of IEEE Winter Vision Meetings, Workshop on Robot Vision (WoRV 2013). IEEE, Clearwater (2013)
11. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes in C: the art of scientific computing, 2nd edn. Cambridge University Press, New York (1992)
12. Ringaby, E., Forssén, P.E.: Scan rectification for structured light range sensors with rolling shutters. In: IEEE International Conference on Computer Vision. IEEE Computer Society Press, Barcelona (2011)
13. Ringaby, E., Forssén, P.E.: Efficient video rectification and stabilisation for cell-phones. International Journal of Computer Vision 96(3), 335–352 (2012)
14. Roth, H., Vona, M.: Moving volume kinectfusion. In: British Machine Vision Conference (BMVC 2012). BMVA, University of Surrey, UK (2012), `http://dx.doi.org/10.5244/C.26.112`
15. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China (2011)
16. Schönemann, P.: A generalized solution of the orthogonal procrustes problem. Psychometrika 31(1), 1–10 (1966)
17. Shoemake, K.: Animating rotation with quaternion curves. In: Int. Conf. on CGIT, pp. 245–254 (1985)
18. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: Proc. of the International Conference on Intelligent Robot Systems, IROS (2012)
19. Whelan, T., McDonald, J., Kaess, M., Fallon, M., Johannsson, H., Leonard, J.J.: Kintinuous: Spatially extended kinectfusion. In: RSS 2012 Workshop on RGB-D Cameras, Sydney (2012)
20. Zhang, Z.: A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(11), 1330–1334 (2000)

Chapter 5
Modeling Paired Objects and Their Interaction

Yu Sun and Yun Lin

5.1 Introduction

Object categorization and human action recognition are two important capabilities for an intelligent robot. Traditionally, they are treated separately. Recently, more researchers started to model the object features, object affordance, and human action at the same time. Most of the works build a relation model between single object features and human action or object affordance and uses the models to improve object recognition accuracies [16, 21, 12].

In our daily life, it is natural that we not only pay our attentions to the objects we hold and manipulate, but also the interactive relationship between the objects. We also select our motions according to the intended interaction we want, which is mostly defined by both objects. For example, when a person holds a pen, there could be many different kinds of motions. However, if the pen is associated to a piece of paper, the human motion with the pen is significantly confined. Most likely, a writing motion will occur. Likewise, if we want to detect the type of object in a human hand, and we have detected a human writing motion and a piece of paper, we have more confidence to believe that the object is a pen than without detecting the writing motion or the paper. There are many similar examples such as a book and a schoolbag, and a teapot and a cup. The interactive motions performed by the humans have strong relationship with both objects. Therefore, the motion information can enhance our belief of the recognition results of the objects. If we can detect a stirring motion and recognize a cup, we can enhance our belief that the object in the human's hand is a spoon. Figure 5.1 shows several objects on a table that have inter-object relationship: a CD and a CD case, a pen and a piece of paper, a spoon and a cup, and a cup and a teapot.

Yu Sun · Yun Lin
Computer Science and Engineering, University of South Florida, Tampa, FL 33620, U.S.A.
e-mail: yusun@cse.usf.edu, yunlin@mail.usf.edu

Y. Sun et al. (eds.), *New Development in Robot Vision*,
Cognitive Systems Monographs 23, DOI: 10.1007/978-3-662-43859-6_5

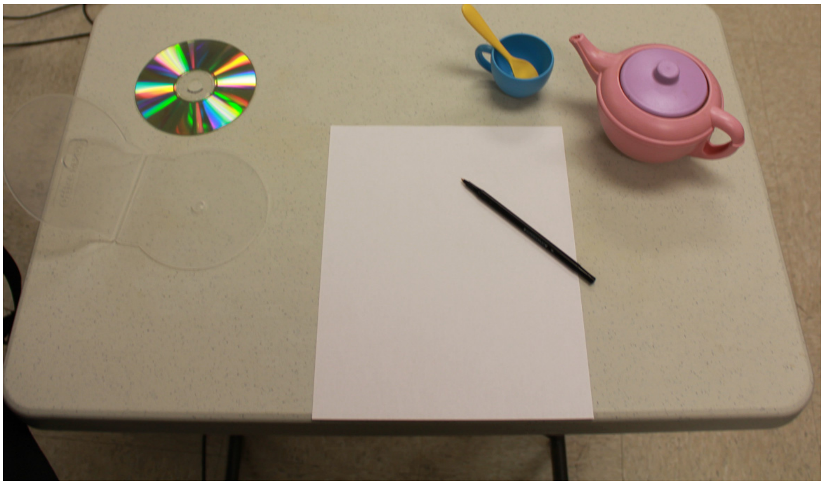


Fig. 5.1 Several objects on a table have inter-object relationships: pen-paper, teapot-cup, cup-spoon, CD-CD case

The connection between the visual recognition and motor action has been studied in neuroscience and cognitive science recently. The concept of objects' affordance has been around since 1977 [14]. Only lately studies on objects' affordance [28, 29, 24] indicated that the mirror neurons in human brains congregated visual and motor responses. In the studies, the researchers found that mirror neurons in the F5 sector of the macaque ventral premotor cortex fired both during observation of interacting with an object and during action execution, but did not discharge in response to simply observing an object [9, 13]. More close to the human-object-object interaction affordance idea, Yoon et al. [32] studied the affordances associated to pairs of objects positioned for action and found an interesting so-called "paired object affordance effect." The effect was that the response time by right-handed participants was faster if the two objects were used together when the active object (supposed to be manipulated) was to the right of the other object. Borghi et al. [3] further studied the functional relationship between paired objects and compared it with the spatial relationship and found that both the position and functional context were important and related to the motion; however, the motor action response was faster and more accurate with the functional context than the spatial context. The study results in neuroscience and cognitive science indicate that there are strong connections between the observation and the motion, and functional relationships between objects are directly associated with the motor actions.

Based on the new findings in neuroscience and cognitive science, we propose to link a pair of objects with their interaction motion directly and we call the interaction motion instead of the functionalities of the object as the inter-object affordance. In this chapter, we attempt to capitalize the strong relationship between paired objects and interactive motion by building an object relation model and associating it to human action model in the human-object-object way to characterize inter-object affordance.

In robotics and related fields, object affordance has only been explored recently in limited works that mainly model the object affordance with the interaction between single object and a human user, and then use the mutual relation to improve the recognition of each other. For example, Gupta and Davis [16] recently achieved inspiring success in using single object action to improve the recognition rate of both the object and human motion. Kjellstrom et. al. [21] used conditional random field (CRF) and factorial conditional random field (FCRF) to model the relationship between object type and human action, in which the 3D hand pose was estimated to represent human action including open, hammer, and pour actions. Most recently, Gall, et. al. [12] have recovered the human action from a set of depth images and then represented object's function and affordance with the human action. In their work, objects were classified according to the involved human action in an unsupervised way base on high-level features.

Another recent approach in literature is to derive the objects' affordance from their low level features or 3D shapes. Stark et. al. [30] obtained the object affordance cues from human hand and object interaction in the training images, and then they detected an object and determine the objects functions according to the objects affordance cue features. Grabner et. al. [15] proposed a novel way to determine object affordance using computer graphical simulation. The system 'imagines' or simulates an actor performing actions on the objects to compute the objects affordances from the object's 3D shape.

In robotics community, there are several existing works on obtaining and using object-action relation. In [1], objects were categorized solely according to object interaction sequences (motion features), but the geometry appearance features of the objects was not considered. First, the objects were segmented out from the background in a number of video sequences, then the space interaction relationship between objects were represented with an undirected semantic graph. Their work was able to represent the object temporal and spatial interactions in an event with a sequence of such graphs.

In summary, most of the existing works focus on object-action interaction, or object geometry-related affordance features. This chapter based on our previous publications [27, 31], introduces our new works on modeling the affordance relationship between objects for object recognition and presents a way to model the inter-object affordance, and then use the inter-object affordance relationship to improve object recognition.

In this chapter, we describe a design of a graphical model that composes of two objects and the human motions that relate both objects. The graphical model contains the inter-object affordance that can be learned to represent the interaction relationship between paired objects, such as teapot-cup, and pen-paper. A Bayesian Network is structured to integrate the paired objects, their interaction, and the consequence of the object interaction. After the description of the Bayesian Network graphical model, we introduce an approach to recognize the paired objects by analyzing and classifying the interactive motions with the statistical knowledge learned

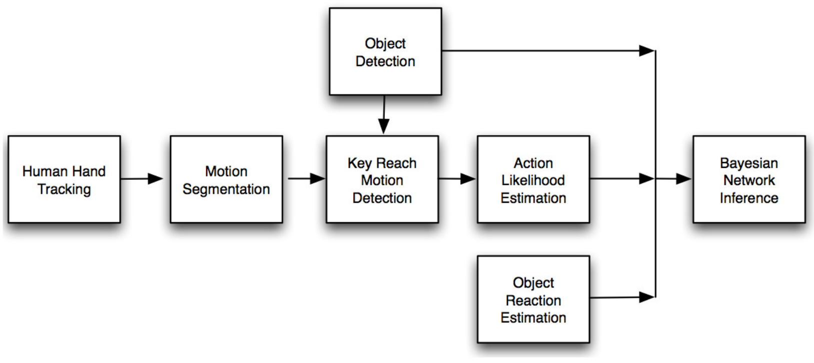

Object Detection
Human Hand Tracking
Motion Segmentation
Key Reach Motion Detection
Action Likelihood Estimation
Bayesian Network Inference
Object Reaction Estimation

Fig. 5.2 The workflow of building the human-object-object interaction model starts with object detection, human hand tracking, and object reaction estimation. In the end the likelihoods are used to build a Bayesian inference network.

from training data. In addition, at the end of the chapter, we extend this approach to leverage the object recognition accuracy from videos with the interactive motion recognition and demonstrate the benefit of the approach with results in several experiments, which show that the detection accuracy of the interactive objects was significantly improved with the introduced approach.

5.2 Human-Object-Object-Interaction Modeling

The workflow to build the human-object-object interaction model is illustrated in Figure 5.2. First, the initial likelihood of the objects' manipulation and reaction is computed. The object initial likelihoods were estimated with a sliding window object detector, which is based on the Histogram of Oriented Gradients (HoG). The initial likelihood of human action is estimated based on the feature of human hand motion trajectory. The human hand was tracked in the whole process, and the hand motion was segmented according to the velocity changing. With motion segmentation and possible object locations, the interactive object pairs were detected in the step of key reach motion detection. The start time of the manipulation was estimated based on the object pair locations and hand motion trajectory. Then, the initial belief of the manipulation was computed.

Object interaction usually leads to a state change of the associated objects. For example, if a CD is put into a CD case, the color of the CD case probably will change. The likelihood of object reaction was estimated by comparing with the training datasets. Finally, the belief in each node was updated with the inference algorithm for Bayesian Networks.

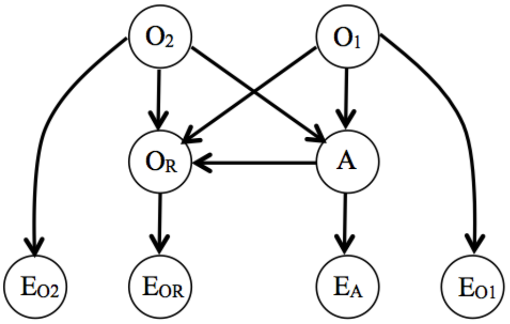


Fig. 5.3 The Bayesian network model used to represent objects, actions and object interactions. O_1 and O_2 represent the two interacting objects, A denotes hand manipulation motion, and O_R is the object reaction.

5.2.1 Bayesian Network Model for HOO Interaction

$$\begin{aligned} P(O_1,O_2,A,O_R|e) \propto\ & P(O_1|e_{O_1})P(O_2|e_{O_2}) \\ & P(A|O_1,O_2)P(A|e_A) \\ & P(O_R|O_1,O_2,A)P(O_R|e_{O_R}) \end{aligned} \tag{5.1}$$

Bayesian network is chosen to model the HOO interaction because it is a powerful inference tool for decision making in the observation of several or many interrelated factors. As illustrated in Figure 5.3, the Bayesian network introduced here has eight nodes. The two interactive objects are represented as node O_1 and node O_2. Node A denotes hand manipulation action, also represents the inter-object affordance. The node O_R represents the object reaction that reflects the change of object state after the interaction. The rest notes are the evidences $e = \{e_{O_1}, e_{O_2}, e_A, e_{O_R}\}$, and they represent the evidence for O_1, O_2, A, and O_R respectively. The nodes are connected according to their conditional dependencies. Since node A is determined by the two interacting objects (O_1 and O_2), they are the parents of node A. Similarly, since the object reaction is the consequence of the two objects and the manipulation, it is the child of those three nodes. The belief for each node can be updated with the messages from the corresponding evidence node. According to the Bayesian rule and conditional independence relations, the joint probability distribution of the paired objects, inter-action, and reaction can be represented with Equation 5.1.

The Bayesian network model can be scaled up by increasing the number of variables for object and action in each node without changing the graphical model structure. Alternatively, we can combine multiple Bayesian networks to form a large-scale graphical model if there are inter-connections between different pairs of objects.

5.2.2 Object Detection

To estimate the initial likelihood of the objects, a detector similar to [7] was designed. The detector works in the sliding window manner, and uses a variant of the HoG feature from [10] to represent the object local features. At each pixel, the color channel with the largest gradient magnitude was used to represent the gradient orientation and magnitude. In each detecting window, the image was divided into 8x8 pixel cells and, for each cell, the pixel level feature was aggregated to a feature map.

Objects were modeled as object type and object location. We computed the object likelihoods:

$P(O_1 = \{obj_1, l^{O_1}\} | e_{O_1})$ and
$P(O_2 = \{obj_2, l^{O_2}\} | e_{O_2})$

for each sliding window with the SVM estimation, in which l^{O_1} is the location of start object and l^{O_2} is the location of the end object . Figure 5.4 shows a sample of the detection results using training image images from the Image-Net [8] and Google Image Search. All of the training images were labeled. For each object, 50 positive and 70 negative examples were used to train an SVM (Support Vector Machine) classifier. The window size and aspect ratio were learned from the training data set. The LibSVM library [4] was used to obtain the probability of the classification for each window.

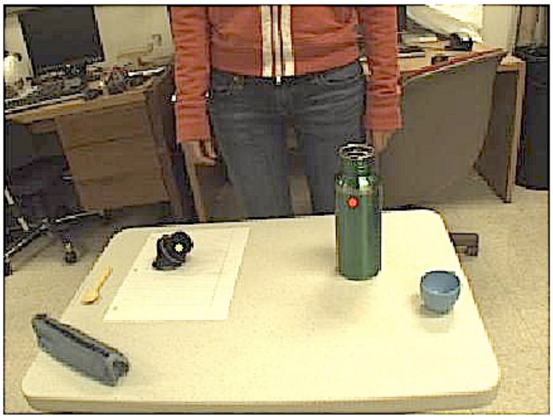


Fig. 5.4 Example result of object detection with SVM classifier using HoG features. Dots indicate detected object centers.

5.2.3 Motion Analysis

The object detector in the previous section can only give us the possible object locations with their types. Since the inter-object affordance is represented by the object interaction, that affordance should be modeled with motion features. To represent the inter-object action – the affordance of the pair, it is necessary to detect and analyze the hand motion that is associated with one of both of the objects. Here the trajectories are segmented and the motion segments are used to represent and recognize the motion types. Generally, there are two kinds of object interactive motion – putting an object into a container and manipulating one of the objects relative to the other [18, 19]. In this chapter, these two kinds of motion are treated the same, although they are considered different in cognition science.

5.2.3.1 Human Hand Tracking in 2D

It is difficult to track an arbitrary hand in a daily-living environment with various background solely based on the hand's shape as a hand can have many different shapes for different gestures. To simplify the discussion, this chapter describe an approach using the human skin color as tracking features since it is much more stable and has been used successfully in previous works [2]. In addition, the skin color model in [5] and the TLD object tracker [20] are combined to build a stable hand tracker. In this approach, the hands in the initial several frames are located using optical flow and the skin color. Then for each additional frame, the hand location is updated according to the color information around the previous hand location and the shape features from TLD tracker. Figure 5.5(a) shows one example of the tracking result and the Figure 5.5(b) shows the tracked trajectory for a whole inter-action motion – putting a CD into its case.

5.2.3.2 Motion Segmentation

From the tracked hand motion trajectory, motion features should be extracted to represent the motion. Here, the obtained trajectories can then be segmented into several pieces according to the velocity and represented with the motion features in the segments. According to [26], there are two kinds of human limb motions: ballistic motion and mass spring motion. In those two kinds of motions, the velocity provides natural indications of the motion segments. The local minimal points in their velocity curves are used to segment the trajectories, and then these small pieces can be either merged or segmented further into possible ballistic and mass spring segments. Similar to the method in [26], the segments are classified into ballistic and mass sprint types according to their velocity features. The features used in this chapter include the maximum velocity, average velocity, number of local minimum point, standard deviation, and motion distance etc. Figure 5.5(c) shows the motion segments in velocity for one motion that is putting a pencil into a pencil case. Similar motion analysis approaches exist in neuroscience and cognitive science to classify and represent motion segments with action chains [11, 17].

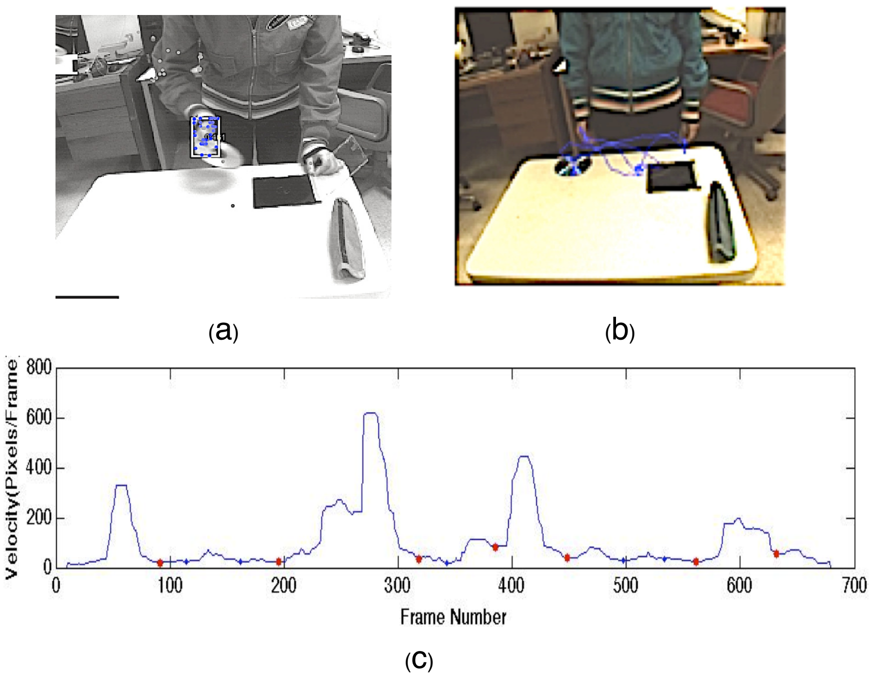
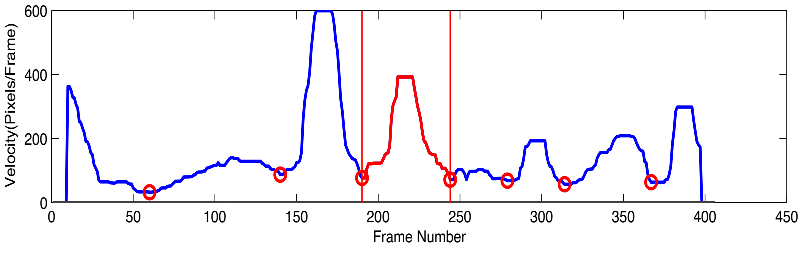


Fig. 5.5 Hand tracking and motion segmentation: (a) right-hand motion tracking; (b) right-hand motion trajectory; (c) motion segmentation with velocity – horizontal axis is time (frame number), and vertical axis represents velocity (pixels per frame). Red circles are detected motion segment boundaries.

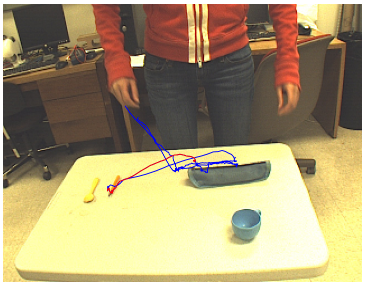
5.2.3.3 Key Reach Motion Detection

In each object interaction process, a human hand carries one object to the location of another object. For example, in the stirring water example, a human hand carries a spoon and moves it to the cup. This reach motion is called the key reach motion. There could be several reach motions in one action. For example, in a process of putting a book into a schoolbag, there are three reach motions. A person first opens the schoolbag, the first reach motion; reach to the book, the second reach motion; and then take the book to the schoolbag to put into it, the third reach motion. However, only the taking the book to the schoolbag is defined as the key reach motion for this interaction as only this reach motion involves both objects. Therefore we name the book as the start object and the schoolbag as the end object as object1 and object2 respectively in the graphical model.

The ballistic segments are then further classified into reach motion and non-reach motion according to motion features including the velocity during acceleration and deceleration, time duration, average velocity, and stand deviation of the velocity. However, it is difficult to segment out the key reach motion only based on the hand



(a)



(b)

Fig. 5.6 Key reach motion detection: (a) red velocity segment represents key reach motion in velocity graph. The red circles are detected motion segment boundaries; (b) The red curve shows key reach motion in image.

motion and to detect if a hand is carrying object or not if the object is small. Instead, we rely on the motion of the object since it is easy to detect the object state around the start and end location of the reach motion. The key reach motion starts from one location (l^a_{r1}), and ends at another location (l^a_{r2}). The distance between the location of start object (l^{O_1}) and the start of the key reach motion location l^a_{r1} is modeled with a normal distribution, $N(|l^a_{r1}l^{O_1}|, \mu_r^{O_1}, \sigma_r^{O_1})$. Likewise, the distance between the location of the end object (l^{O_2}) and l^a_{r2} is modeled with $N(|l^a_{r2}l^{O_2}|, \mu^{O_2}, \sigma_a^{O_2})$. The start and end locations for each reach motion are obtained in the tracking. Then, the start object, end object, and the key reach motion are detected at the same time, according to the two distributions values. Here $\mu_r^{O_1}$, $\sigma_r^{O_1}$, $\mu_a^{O_2}$, and $\sigma_a^{O_2}$ are learned from the training data set. In the key reach motion, human hand carries object1 from location l^{O_1} to location l^{O_2} , so the belief of the key reach motion can be further enhanced by checking if the detected start object (object1) is removed or not. This can be carried out by comparing the likelihood value of object1 at location l^{O_1} before and after the key reach motion. Figure 5.6 shows the key reach motion segment detected (marked as red) from the entire motion that put a pencil into a pencil case.

5.2.3.4 Manipulation Motion Estimation

A manipulation action can be modeled with the features in the human hand trajectory. The features are the start time (t_s^a), the end time (t_e^a), the two reach locations (l_{r1}^a, l_{r2}^a), and the manipulation type (T^a). According to Equation 1, we model the conditional probability $P(A|O_1O_2)$, and the initial likelihood of A, $P(A|e_A)$. $P(A|O_1O_2)$ can be computed with Equation refeq:moo1. If we define l_s^a as the hand location for the start time t_e^a, we can model $P(t_s^a, t_e^a|O_1O_2)$ with $N(|l_s^a l^O|, \mu_r^O, \sigma_r^O)$, and O is either O_1 or O_2. μ_r^O is the mean of the grasping distance for the object O, while σ_r^O is the variance, which can be learned from the training data. $P(l_{r1}^a|O_1)$ and $P(l_{r2}^a|O_2)$ are modeled as normal distributions $N(|l_{r1}^a l^{O_1}|, \mu_r^{O_1}, \sigma_r^{O_1})$ and $N(|l_{r2}^a l^{O_2}|, \mu_a^{O_2}, \sigma_a^{O_2})$, which have been discussed in Section 5.2.3.3. $P(T^A|obj_1, obj_2)$ is computed according to the occurrence of manipulation type and object type in the training data.

$$P(A|O_1O_2) = P(t_s^a, t_e^a|O_1O_2)P(l_{r1}^a|O_1) \\ P(l_{r2}^a|O_2)P(T^a|obj1, obj2) \tag{5.2}$$

We estimate the likelihood $P(A|e_A)$ with the features from the hand motion trajectory. Based on the segmentation results in Section 5.2.3.2, the ballistic and mass spring segments are replaced with labels. The manipulation motions are classified according to the numbers of ballistic and mass spring segments, the translation rate of the two segments, and time duration etc. Linear SVM is trained as the classifier and gives the likelihood of the manipulation.

5.2.4 Object Reaction

The object reaction node is modeled with two parameters: reaction type (T^R) and reaction location (l^R). It is difficult to fully model the object reaction. Therefore, we only consider the state change of the object2 after the interaction. Similar to [4], we use the color histogram at the object2 to represent the object reaction. We estimate $P(O_R|e_{O_R})$ by comparing the histogram of the object2 with the histogram of the training instances from the training data set. Then we model the prior $P(O_R|O_1, O_2, A)$ according to Equation (5.3). $P(l^R|O_2)$ is model with $N(|l^R l^{O_2}|, \mu^R, \sigma^R)$, and parameters μ^R and σ^R are learned from the training data. $P(T^R|O_1, O_2, A)$ is learned from the training data set by counting the occurrence of T^R, O_1, O_2 and A.

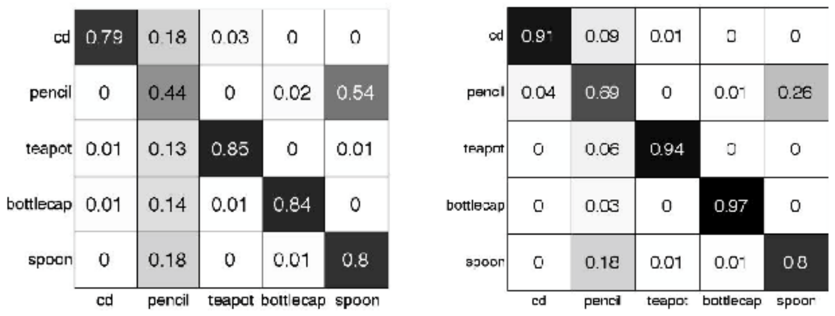
$$P(O_R|O_1, O_2, A) = P(l^R|O_2)P(T^R|O_1, O_2, A) \tag{5.3}$$

5.2.5 Bayesian Network Inference

After getting the key reach motion and the interaction object pair locations, we estimate the parameters for A and O_R according to Sections 5.2.3.3 and 5.2.3.4. We perform the inference with Pearls algorithm [25] once all of the initial likelihoods for O_1, O_2, A, and O_R are estimated. The Bayesian Network, the object classifier and the manipulation classifier are trained with fully-labeled data.

5.3 Experiments and Results

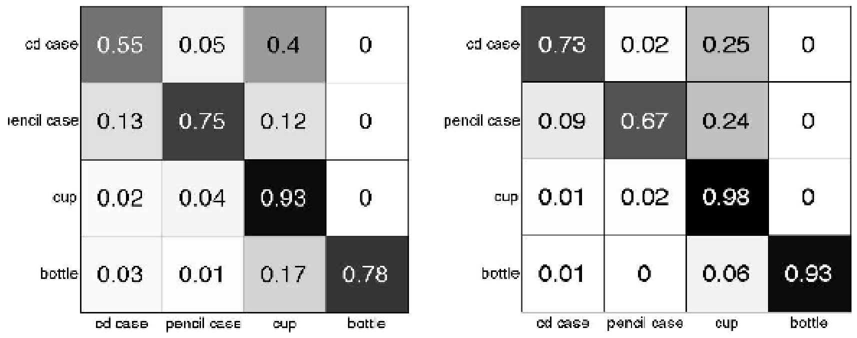
The following experiment and evaluation results demonstrate how this approach is used and its performance. A dataset was collected from six subjects who performed five types of interactions of five pairs of objects. The interaction object pairs included teapot-cup, pencil-pencil case, bottle cap-bottle, CD-CD case and spoon-cup. The actions for these object pairs were pouring water from a teapot to a cup, putting a pencil into a pencil case, screwing on a bottle cap, putting a CD into the CD case and stirring a spoon in a cup. All of these objects and actions were chosen because they are very common in everyday life, and they are representative for different inter-object affordance relationships. The data from four subjects were used



	cd	pencil	teapot	bottlecap	spoon
cd	0.79	0.18	0.03	0	0
pencil	0	0.44	0	0.02	0.54
teapot	0.01	0.13	0.85	0	0.01
bottlecap	0.01	0.14	0.01	0.84	0
spoon	0	0.18	0	0.01	0.8

	cd	pencil	teapot	bottlecap	spoon
cd	0.91	0.09	0.01	0	0
pencil	0.04	0.69	0	0.01	0.26
teapot	0	0.06	0.94	0	0
bottlecap	0	0.03	0	0.97	0
spoon	0	0.18	0.01	0.01	0.8

(a)



	cd case	pencil case	cup	bottle
cd case	0.55	0.05	0.4	0
pencil case	0.13	0.75	0.12	0
cup	0.02	0.04	0.93	0
bottle	0.03	0.01	0.17	0.78

	cd case	pencil case	cup	bottle
cd case	0.73	0.02	0.25	0
pencil case	0.09	0.67	0.24	0
cup	0.01	0.02	0.98	0
bottle	0.01	0	0.06	0.93

(b)

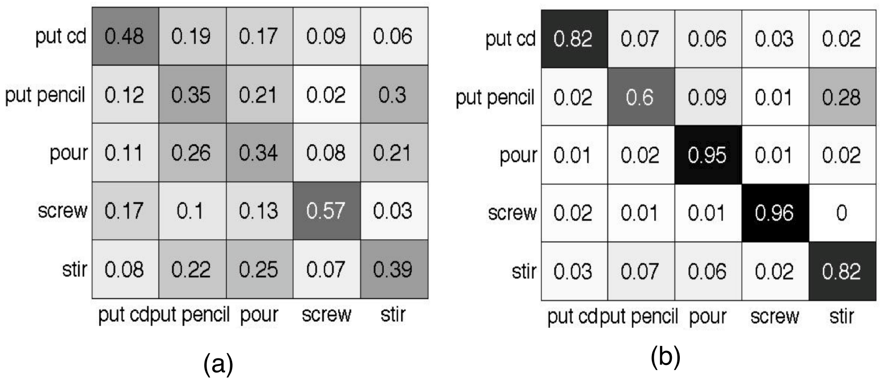
Fig. 5.7 Results comparison: (a) Object1 likelihood confusion matrix. The left one shows the result using HoG detector. The right shows the result using the described approach; (b) Object2 likelihood confusion matrix. The left one shows the result using HoG detector. The right shows the result using our framework.

for training, and the data from the rest two subjects were used for testing. Each subject performed every action for two or three trials.

The object classifier, the action classifier and the Bayesian Network were trained in supervised manner. As stated before, the training images for the object classifier were collected from the ImageNet [8] and Google Image Search. The training data for the action classifier and the Bayesian Network were collected from manually labeled video sequences taken in our experiments. Fifty videos sequences that performed by four subjects were used for training. In each training video sequence, object locations, reach locations and action type and the start frame of the manipulation were manually labeled.

The test data set are video sequences that contain the action sequences performed by the other two subjects. Figure 5.7(a) shows the object classification confusion matrixes for object1 for the testing data, which is the object at the beginning of the key reach motion. Figure 5.7(b) presents the likelihood confusion matrixes for object2 that is the object at the end of the key reach motion. In each of the confusion matrices, the *i*th row represents the likelihood value when the *i*th type of object presents. For object1, as we can see from the confusion matrices, it was difficult to distinguish a pencil from a spoon only based on the appearance, which is consistent with the fact that they have the similar shape and both of them are small. With our approach, by including the context of human-object-object interaction, our Bayesian network was able to distinguish and recognize the spoon and the pencil more much accurately. The average recognition success rate of our approach for object1 was improved from 72.6% to 86.0% and improved from 75.3% to 82.8% for object2.

Among the five actions studied, if based only on motion features, it was difficult to distinguish putting a CD into a CD case, putting a pencil into a pencil case, pouring water into a cup, and stirring water in a cup because they had the similar motion patterns. With the human-object-object interaction framework, they could be distinguished. Figure 5.8(a) shows the likelihood confusion matrix that was estimated



(a)

	put cd	put pencil	pour	screw	stir
put cd	0.48	0.19	0.17	0.09	0.06
put pencil	0.12	0.35	0.21	0.02	0.3
pour	0.11	0.26	0.34	0.08	0.21
screw	0.17	0.1	0.13	0.57	0.03
stir	0.08	0.22	0.25	0.07	0.39

(b)

	put cd	put pencil	pour	screw	stir
put cd	0.82	0.07	0.06	0.03	0.02
put pencil	0.02	0.6	0.09	0.01	0.28
pour	0.01	0.02	0.95	0.01	0.02
screw	0.02	0.01	0.01	0.96	0
stir	0.03	0.07	0.06	0.02	0.82

Fig. 5.8 Action likelihood confusion matrix: (a) result using only motion features; (b) result using framework. The *i*th row shows likelihood value when *i*th action is categorized.

with only hand motion features. Figure 5.8(b) shows the action confusion matrix using human-object-object interaction framework. We can see that the overall average recognition rate across all objects improved from 42.6% to 83.0%.

5.4 Conclusions

This chapter described a recent investigation on modeling the human-object-object interaction with Bayesian network. The object categorization and action recognition are linked using human-object-object-interaction affordance framework. The knowledge of object affordance is learned from labeled video sequences, and represented with a Bayesian Network. The elements of the Bayesian Network include objects, human action and object reaction. The experiments with six subjects and about 70 video sequences have shown that with human-object-object-interaction affordance knowledge, the object classification rate, and especially the action recognition rate were significantly improved.

The learned affordance knowledge represented in the Bayesian network can also help us to learn affordance motion more precisely and apply the learned motion to guide and control robot motions in a learning from demonstration framework such as in [22], since the interaction affordance knowledge can suggest proper actions that the robot should perform. The interaction motion can also be used to compute a feasible and stable manipulation-task oriented grasp planning [23] with the help of the object categorization. The motion analysis presented in this chapter is only one of many approaches. A functional motion analysis could be applied (similar to [6]) to capture more dynamic features and represent the motion in a lower dimensional space.

References

1. Aksoy, E., Abramov, A., Worgotter, F., Dellen, B.: Categorizing object-action relations from semantic scene graphs. In: IEEE Intl. Conference on Robotics and Automation, pp. 398–405 (2010)
2. Argyros, A.A., Lourakis, M.I.A.: Real-time tracking of multiple skin-colored objects with a possibly moving camera. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004. LNCS, vol. 3023, pp. 368–379. Springer, Heidelberg (2004)
3. Borghi, A., Flumini, A., Natraj, N., Wheaton, L.: One hand, two objects: emergence of affordance in contexts. Brain and Cognition 80(1), 64–73 (2012)
4. Chang, C., Lin, C.: Libsvm: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2(3), 1–27 (2011)
5. Conaire, C., O'Connor, N.E., Smeaton, A.F.: Detector adaptation by maximising agreement between independent data sources. In: IEEE International Workshop on Object Tracking and Classification Beyond the Visible Spectrum, pp. 1–6 (2007)
6. Dai, W., Sun, Y., Qian, X.: Functional analysis of grasping motion. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–7 (2013)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition, pp. 886–893 (2005)

8. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Int. Conf. on Computer Vision and Pattern Recognition, pp. 248–255 (2009)
9. Di Pellegrino, G., Fadiga, L., Fogassi, L., Gallese, V., Rizzolatti, G.: Understanding motor events: A neurophysiological study. Exp. Brain Res. 91, 176–180 (1992)
10. Felzenszwalb, P.F., McAllester, D., Ramanan, D.: A discriminatively trained, multi-scale, deformable part model. In: Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
11. Fogassi, L., et al.: Parietal lobe: From action organization to intention understanding. Science 29(308), 662–667 (2005)
12. Gall, J., Fossati, A., Gool, L.: Functional categorization of objects using real-time markerless motion capture. In: Conference on Computer Vision and Pattern Recognition, pp. 1969–1976 (2011)
13. Gallese, V., Fogassi, L., Fadiga, L., Rizzolatti, G.: Action representation and the inferior parietal lobule. In: Prinz, W., Hommel, B. (eds.) Attention and Performance XIX. Common mechanisms in perception and action. Oxford University Press, Oxford (2002)
14. Gibson, J.: The theory of affordances. In: Shaw, R., Bransford, J. (eds.) Perceiving, Acting and Knowing. Erlbaum, Hillsdale (1977)
15. Grabner, H., Gall, J., Van Gool, L.: What makes a chair a chair? In: Conference on Computer Vision and Pattern Recognition, pp. 1529–1536 (2011)
16. Gupta, A., Davis, L.: Objects in action: An approach for combining action understanding and object perception. In: Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
17. Hamilton, A., Grafton, S.: The motor hierarchy: from kinematics to goals and intentions. In: Haggard, P., Rosetti, Y., Kawato, M. (eds.) Attention and Performance, ch. 22 (2007)
18. Iacoboni, M., Molnar-Szakacs, I., Gallese, V., Buccino, G., Mazziotta, J., et al.: Grasping the intentions of others with one's own mirror neuron system. PLoS Biol. 3(3) (2005)
19. Jax, S.A., Buxbaum, L.J.: Response interference between functional and structural actions linked to the same familiar object. Cognition 115(2), 350–355 (2010)
20. Kala, Z., Matas, J., Mikolajczyk, K.: P-n learning: Bootstrapping binary classifiers by structural constraints. In: Conference on Computer Vision and Pattern Recognition, pp. 49–56 (2010)
21. Kjellstrom, H., Romero, J., Kragic, D.: Visual object-action recognition: Inferring object affordances from human demonstration. Computer Vision and Image Understanding 115, 81–90 (2010)
22. Lin, Y., Shaogang, R., Clevenger, M., Sun, Y.: Learning grasping force from demonstration. In: IEEE Intl. Conference on Robotics and Automation, pp. 1526–1531 (2012)
23. Lin, Y., Sun, Y.: Task-oriented grasp planning based on disturbance distribution. In: International Symposium on Robotics Research (ISRR), pp. 1–16 (2013)
24. Oztop, E., Kawato, M., Arbib, M.: Mirror neurons and imitation: a computationally guided review. Epub Neural Networks 19, 254–271 (2006)
25. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Network and Plausible Inference. Morgan Kaufmann (1988)
26. Prasad, V.S.N., Kellokumpu, V., Davis, L.S.: Ballistic hand movements. In: Perales, F.J., Fisher, R.B. (eds.) AMDO 2006. LNCS, vol. 4069, pp. 153–164. Springer, Heidelberg (2006)

27. Ren, S., Sun, Y.: Human-object-object-interaction affordance. In: IEEE Workshop in Robot Vision, pp. 1–6 (2013)
28. Rizzolatti, G., Craighero, L.: The mirror neuron system. Ann. Rev. Neurosci. 27, 169–192 (2004)
29. Rizzolatti, G., Craighero, L.: Mirror neuron: A neurological approach to empathy. In: Changeux, J.P., Damasio, A.R., Singer, W., Christen, Y. (eds.) Neurobiology of Human Values, Springer, Heidelberg (2005)
30. Stark, M., Lies, P., Zillich, M., Wyatt, J.C., Schiele, B.: Functional object class detection based on learned affordance cues. In: Gasteratos, A., Vincze, M., Tsotsos, J.K. (eds.) ICVS 2008. LNCS, vol. 5008, pp. 435–444. Springer, Heidelberg (2008)
31. Sun, Y., Ren, S., Lin, Y.: Object-object interaction affordance learning. Robotics and Autonomous Systems (in Press)
32. Yoon, E., Humphreys, W., Riddoch, M.: The paired-object affordance effect. J. Exp. Psychol. Human 36, 812–824 (2010)

Chapter 6
Probabilistic Active Recognition of Multiple Objects Using Hough-Based Geometric Matching Features

Natasha Govender, Philip Torr, Mogomotsi Keaikitse,
Fred Nicolls, and Jonathan Warrell

6.1 Introduction

3D Object recognition is an important task for mobile platforms to dynamically interact in human environments. This computer vision task also plays a fundamental role in the areas of automated surveillance, Simultaneous Localization and Mapping (SLAM) applications for robots and video retrieval. The recognition of objects in realistic circumstances, where multiple objects may appear together with significant occlusions and clutter from distracter objects, is a complicated and challenging problem. Particularly in such situations multiple viewpoints are necessary for recognition [17] as single viewpoints may be of poor quality and not contain sufficient information to reliably recognise or verify all objects' identities unambiguously.

Here, we consider the restricted task of recognizing which of a set of known objects are present in a small isolated circular test area. A mobile platform may take images from a set of evenly spaced viewpoints around the perimeter of this area. This task forms a multi-object extension of the single-object 1 degree-of-freedom (1 DoF) active recognition tasks investigated in [1] and [10]. A challenge particular to our setting is to find an effective object representation that will be robust to occlusions and clutter. Further challenges for both single and multiple object settings

Natasha Govender · Mogomotsi Keaikitse
Mobile Intelligent Autonomous Systems, CSIR, South Africa
e-mail: {ngovender,mkeaikitse}@csir.co.za

Philip Torr
Department of Engineering Science, Oxford University
e-mail: philip.torr@eng.ox.ac.uk

Fred Nicolls
Department of Electrical Engineering, University of Cape Town, South Africa
e-mail: fred.nicolls@uct.ac.za

Jonathan Warrell
Biosciences, CSIR, South Africa
e-mail: jwarrell@csir.co.za

Y. Sun et al. (eds.), *New Development in Robot Vision*,
Cognitive Systems Monographs 23, DOI: 10.1007/978-3-662-43859-6_6

involve providing a means of combining data across viewpoints, while maintaining information about uncertainty; and choosing a mechanism for selecting the order in which to capture new data, typically depending on the expected informativeness of a new viewpoint. We introduce our own dataset specifically to explore this multiple object recognition task.

With respect to representation, a number of previous methods have sought to recognize 3D objects in cluttered environments by matching features extracted at sparse interest points to training images of an object at multiple viewpoints [12][6][14]. This approach naturally allows for the matching of both local features and global geometric relations under rigid transformations. We build on such work in our approach to create an image representation suitable for our active recognition setting. Particularly, we use the Scale Invariant Feature Transform (SIFT) [13] detector and descriptor to extract relevant object features, and match each test viewpoint to all training viewpoints seen at training time by allowing matching SIFT features to vote for rigid transformations in a Hough-voting scheme similar to [12]. We show how representing a test image by a feature vector containing the best matching counts from all training views under this process contains sufficient information to build effective probabilistic models for active recognition in the multi-object scenario above, and further develop efficient viewpoint selection strategies. We are also able to outperform existing active recognition methods which are similarly based on SIFT-features, but which do not incorporate geometric matching [10].

With respect to data integration, we build on a number of works which have explored Bayesian methods of integration across a range of active sensing tasks. These include both general frameworks for active sensing as in [5], as well as specific models for scene exploration and tracking from surveillance videos [18], medical diagnostics [19], and object recognition from a mobile platform [1][3]. In many of these cases however, attention is paid to the general problems of optimal methods for fusing data (using Bayes Theorem) and planning sensing strategies while assuming that a probabilistic model for the phenomenon of interest (object/environment/diagnostic features) is given. Particularly in the case of active object recognition[1][3], simple probabilistic models and highly controlled datasets are used in order to highlight general approaches to data fusion. While adopting a Bayesian framework for data fusion similar to those mentioned, we explore a number of probabilistic models based on the Hough-matching feature representation specifically designed to represent hypotheses about the presence of both single and multiple objects, as well as the poses of all objects that are present. This allows us to cope effectively with more complex test data of the kind discussed. In the case of matching multiple objects, we show how effective probabilistic models can be built by using empirical distributions of matching counts for each object/viewpoint. We demonstrate empirically the gains that can be achieved through using such multi-object models over simpler single-object models (both ours and other methods) in our test scenario.

Finally, we provide an extensive evaluation of a viewpoint selection mechanism introduced in our previous work [7] for the case of active recognition of single objects, extended here to the multiple-object case. This algorithm uses a vocabulary tree data structure [15] to cluster all SIFT vectors from our training set, and builds a *uniqueness map* for each object which summarizes the uniqueness of each object viewpoint by summing a Term Frequency Inverse Document Frequency (TFIDF) metric across the counts of the leaf-node clusters appearing at that viewpoint. These uniqueness maps are then used to select the next viewpoint in an active scenario, based on the current belief about which object/objects are present. The approach is particularly efficient compared to mutual information, as is commonly used in Bayesian models [5][18][11][19], since it does not require the averaging of entropy scores for every possible outcome. Further, it can also be used in non-Bayesian active contexts, such as the model of [10], where it is more efficient to evaluate than the expected activation. We compare this selection mechanism with previous methods in both Bayesian and non-Bayesian contexts, showing it to perform well in terms of efficiency and accuracy in the multi-object setting compared to mutual information and expected activation.

In summary, we develop an active recognition pipeline specifically to handle the realistic situation of simultaneously recognizing multiple objects in close proximity, which may be subject to extensive occlusions and clutter from distracter objects. Within our approach, we highlight three main contributions:

- We propose an image representation based on Hough-based geometric matching counts to exemplar images to cope with clutter and occlusions.
- We develop a Bayesian model for data fusion which maintains a distribution over multiple object and pose hypotheses.
- We extend the viewpoint selection mechanism in [7] to multiple objects, and provide an extensive evaluation, comparing it with alternative mechanisms.

The structure of the chapter is as follows. Section 6.2 begins by discussing related work. Section 6.3 uses the scenario of a single object/pose hypothesis to introduce our Hough-based geometric matching features and the viewpoint selection mechanism of [7] in the context of a simpler probabilistic model. Section 6.4 then extends the approach outlined to the multiple-object scenario, detailing the probabilistic model used, and associated adaptations to other elements of the algorithm. In Section 6.5 we compare the complexity of our proposed viewpoint selection mechanism with the use of mutual information as applied to our model. We then demonstrate the efficiency and accuracy of our approach as detailed above in Section 6.6, comparing our approach as a whole using both single and multiple object hypotheses to that of [10], and further comparing our viewpoint selection mechanism with mutual information [5][18] and expected activation [10] mechanisms. Finally, we conclude with a discussion in Section 6.7.

6.2 Related Work

A number of methods have considered feature representations similar to ours outside of the active vision setting. Our Hough-based matching feature representation is inspired by [12], who adopt a similar matching process for locating 3D-object views in single images. Our Hough-voting procedure has slight differences to [12] (we do not construct a full multi-dimensional vote space and neglect the final least-squares validation step). Also, [12] use matching counts along with a variety of other features to build a probabilistic model to accept or reject the presence of an object in a image while we construct a variety of distributions for active recognition scenarios using the matching counts alone as features. Our simpler representation however proves sufficient for the active setting. Further non-active methods using geometric SIFT-based matching to cope with cluttered scenes include [6], who consider both single and multiple viewpoints from static cameras at test time, and [14], who consider the matching of isolated pairs of viewpoints.

A wide range of general frameworks for active vision and active sensing have been explored, including information theoretic and Bayesian approaches [5][3][18][1], discriminative approaches [10][9], and approaches based on other theoretical models such as possibilistic and Dempster-Shafer theory [8],[2]. In Borotschnig et. al. [2] a comparison was conducted between probabilistic (Bayesian), possibilistic and Dempster-Shafer theory approaches to data fusion. They concluded that the probabilistic approach worked best for 3D active object recognition, although all these methods use test images with a single object in an uncluttered environment with no occlusions.

Our framework follows that of [5] and [1] in terms of the general Bayesian form of our updates. However, [5] and [1] consider only recognizing single objects in uncluttered environments, allowing them to use a global eigenspace model as their image representation (which is sensitive to clutter). Further, [5] and [1] consider only the case of a single object/pose hypothesis, and do not consider the Bayesian updates that are required in the multiple object/pose hypothesis case we consider. In addition, [5] proposes the mutual information criterion as a viewpoint selection mechanism, which has been subsequently used/proposed by [18][11][19]. As noted, this is expensive to calculate, and requires the collection of extensive statistics at training time, although as [5] discuss, it provides the optimal strategy provided the underlying models are correct.

Although not a fully Bayesian method, the active recognition approach of [10] resembles ours in terms of underlying representations, where [10] extract SIFT features at sparse interest points in both test and training images, and match test points to training points using nearest neighbour matching. An activation score for each object/viewpoint hypothesis is formed based on the counts of the test points matching to the corresponding training image. The main differences from our method are that we also incorporate a Hough-based geometric matching step when calculating the matching counts; we estimate probabilistic models using these counts instead of summing them directly; and that we provide a framework for updating multiple object/pose hypotheses in place of a single hypothesis. We show empirically that these factors provide advantages for our method over [10] in our test scenario. Further, [10] propose a viewpoint selection strategy based on expected increase in the best activation score for their method. We show empirically that our proposed selection method outperforms this strategy, thus showing the utility of our mechanism in a non-Bayesian context.

Other related active recognition methods include Callari et. al.[3], who estimate Bayesian probabilities with neural nets and minimize the expected ambiguity measured by Shannon entropy, and the boosting and support vector machines strategy of [9]. Both approaches use substantially uncluttered test data, and estimate only a single object hypothesis (in the case of [3] without a pose estimation).

Finally, we make use of a vocabulary tree representation [15] in our proposed viewpoint selection strategy. This has been shown to be effective in both 2D and 3D recognition tasks [15], and also in Simultaneous Localization and Mapping (SLAM) approaches for matching similar images and for loop closure [16].

6.3 Active Recognition of a Single Object

We first describe a Bayesian approach to active recognition of a single object at unknown orientation. Although we give specific forms for the image representation, likelihood model and viewpoint selection rules below, the framework is general, and different choices can be substituted for these.

Problem Statement: The active recognition task for a single object can be defined as follows. At training time, for each object $o \in \{1...N_o\} = \mathscr{O}$ we capture set of images, one at each of a series of N_θ regularly spaced training views around the object indexed by their viewing angle, for example $\theta \in \{0°, 20°, 40°, ...340°\} = \mathscr{V}$, $N_\theta = |\mathscr{V}|$. For simplicity, we consider only varying the viewing angle around one axis (e.g. vertical), although minimal changes are necessary to incorporate viewpoints from across a viewing sphere. We thus have a training image $I^{\text{train}}_{o,\theta}$ for each object/view pair.

At test time, we are presented again with one of the training objects, and must identify a) the object present $o^\star$, and b) the orientation of the object, which may be specified by the training viewpoint $\theta^\star$ corresponding to a reference test view. We are allowed to capture images of the test object at a sequence of test views, $\delta_1, \delta_2, ... \in \mathscr{V}$, where the angles δ_t can be in any order. We label the image corresponding to the t'th test view $I^{\text{test}}_{\delta_t}$, and treat $\delta_1 = 0°$ as a reference view (i.e. $I^{\text{train}}_{o^\star,\theta^\star}$ will denote the training view we believe corresponds to $I^{\text{test}}_{\delta_1}$). Typically, an active object recognition algorithm will include the following components: an *update strategy* for incorporating new information from each viewpoint as it is seen and using it to update a belief/score for the correct $o^\star$ and $\theta^\star$; a *viewpoint selection strategy* for choosing the sequence of test views $\delta_1, \delta_2, ...$; and a *stopping criterion* to decide when to stop capturing further viewpoints and generate the output. We outline a Bayesian algorithm below for single object active recognition which incorporates these elements. These are listed in turn, following a description of our image representation.

Image Representation: For a given test image, I^{test}_δ, we apply the method of [13] to generate a sparse set of SIFT descriptors to represent the image. We index these by $\mathscr{I}^{\text{test}}_\delta = \{1...N^{\text{test}}_\delta\}$, where N^{test}_δ is the number of descriptors found for test image I^{test}_δ. Each descriptor index is associated with a 128-dimensional SIFT descriptor, a location, scale and orientation, as can be expressed by introducing functions d^{test}_δ : $\mathscr{I}^{\text{test}}_\delta \to \mathbb{R}^{128}$, $x^{\text{test}}_\delta : \mathscr{I}^{\text{test}}_\delta \to \mathbb{R}$, $y^{\text{test}}_\delta : \mathscr{I}^{\text{test}}_\delta \to \mathbb{R}$, $s^{\text{test}}_\delta : \mathscr{I}^{\text{test}}_\delta \to \mathbb{R}$ and $\phi^{\text{test}}_\delta$: $\mathscr{I}^{\text{test}}_\delta \to [0\ 360)$. Similarly, we can form a sparse representation for a given training image, $I^{\text{train}}_{o,\theta}$, by introducing the functions $d^{\text{train}}_{o,\theta}$, $x^{\text{train}}_{o,\theta}$, $y^{\text{train}}_{o,\theta}$, $s^{\text{train}}_{o,\theta}$ and $\phi^{\text{train}}_{o,\theta}$ over index set $\mathscr{I}^{\text{train}}_{o,\theta}$.

We now consider the set of *matched pairs* of descriptors between training image $I^{\text{train}}_{o,\theta}$ and test image I^{test}_{δ}. which we write $\mathscr{M}^{o,\theta}_{\delta}$. This consists of all pairs of descriptors whose distance falls below a certain threshold, τ^{match}:

$$\begin{aligned}\mathscr{M}^{o,\theta}_{\delta} &= \{(n_1,n_2) \in \mathscr{I}^{\text{train}}_{o,\theta} \times \mathscr{I}^{\text{test}}_{\delta}| \\ &\quad |d^{\text{train}}_{o,\theta}(n_1) - d^{\text{test}}_{\delta}(n_2)|_2 < \tau^{\text{match}}\}\end{aligned} \tag{6.1}$$

These matches are then inputted to a Hough transform voting procedure to assist in removing any randomly occurring SIFT matches. This is achieved by allowing each match to vote for an approximate translation, scaling and rotation of the object. Given $\mathscr{B}_1$ (x-translation), $\mathscr{B}_2$ (y-translation), $\mathscr{B}_3$ (rotation) and $\mathscr{B}_4$ (scale) for the sets of bins used in the Hough transform, for each $m \in \mathscr{M}^{o,\theta}_{\delta}$ we generate votes $v_1(m) : \mathscr{M}^{o,\theta}_{\delta} \to \mathscr{B}_1$ ... $v_4(m) : \mathscr{M}^{o,\theta}_{\delta} \to \mathscr{B}_4$ as follows. For the scale and rotation votes, we simply quantize the rotation/scale differences/ratios of the matched pair of descriptors to the nearest bin: $v_3(m) = v_3(n_1,n_2) = \text{round}_{\mathscr{B}_3}(\phi^{\text{test}}_{\delta}(n_2) - \phi^{\text{train}}_{o,\theta}(n_1))$ and $v_4(m) = v_4(n_1,n_2) = \text{round}_{\mathscr{B}_4}(s^{\text{test}}_{\delta}(n_2)/s^{\text{train}}_{o,\theta}(n_1))$ (writing $\text{round}_{\mathscr{B}_3}$ and $\text{round}_{\mathscr{B}_4}$ for functions which return the corresponding bin for a given rotation/scale differences/ratios). To generate the translation votes, we solve for the similarity transform that will map $(x^{\text{train}}_{o,\theta}(n_1), y^{\text{train}}_{o,\theta}(n_1))$ to $(x^{\text{test}}_{\delta}(n_2), y^{\text{test}}_{\delta}(n_2))$ using the known scaling and rotation above with an unknown translation (tx_m, ty_m) (details of this calculation are given in [12]). We then set $v_1(m) = \text{round}_{\mathscr{B}_1}(tx_m)$ and $v_2(m) = \text{round}_{\mathscr{B}_2}(ty_m)$, with $\text{round}_{\mathscr{B}_1}, \text{round}_{\mathscr{B}_2}$ defined similarly to above.

Having generated the required votes, we find:

$$\begin{aligned} b^*_1 &= \text{argmax}_{b\in\mathscr{B}_1} \sum_m [v_1(m) = b] \\ &\dots \\ b^*_4 &= \text{argmax}_{b\in\mathscr{B}_4} \sum_m [v_4(m) = b] \end{aligned} \tag{6.2}$$

that is, the bins which separately accumulated the most votes (where [.] is 1 for a true condition and 0 otherwise). We then define the *Hough-matching score* for $I^{\text{train}}_{o,\theta}$ and I^{test}_{δ} to be the number of matched descriptors voting simultaneously for these bins:

$$H^{o,\theta}_{\delta} = |\{m \in \mathscr{M}^{o,\theta}_{\delta_t}| \bigwedge_{n=1..4} v_n(m) = b^*_n\} \tag{6.3}$$

We then form a feature vector for a given test image $\mathbf{f}^{\text{test}}_{\delta}$ by concatenating these scores across all training images:

$$\mathbf{f}^{\text{test}}_{\delta} = [[H^{o_1,\theta_1}_{\delta}; H^{o_1,\theta_2}_{\delta}; ...]; [H^{o_2,\theta_1}_{\delta}; H^{o_2,\theta_2}_{\delta}; ...]; ...] \tag{6.4}$$

where [.;.] denotes vertical concatenation.

Update Strategy: We outline here a Bayesian update strategy which maintains a distribution over object and pose random variables, O and Θ_0 (taking values in $\mathscr{O}$ and $\mathscr{V}$ respectively), given the images observed up to a given time-step t. This can be expressed as $P_t(o,\theta_0) = P(o,\theta_0|\mathbf{f}^{\text{test}}_{\delta_1},...\mathbf{f}^{\text{test}}_{\delta_t})$, where $\mathbf{f}^{\text{test}}_{\delta_t}$ is as above, and we denote by $P_t(o,\theta_0)$ the probability at time-step t that the test object is o and the test view at the reference test viewpoint $\delta_1 = 0°$ corresponds to training view $\theta_0 \in \mathscr{V}$. For simplicity, we assume that the images we see at different viewpoints are generated independently given o and θ_0. In general, this will not be the case, since we expect there to be high correlations between the images we see for instance at neighboring viewpoints. However, granting this assumption allows us to build a separate probability model for each object/viewpoint combination $P(\mathbf{f}^{\text{test}}_{\delta}|o,\Theta_\delta = \theta_\delta)$, where the random variable Θ_δ corresponds to the training view seen at a particular δ, which stands in the deterministic relation to Θ_0, $\Theta_\delta = \Theta_0 + \delta$ modulo 360°. Treating these probabilities as likelihood terms, we can recursively estimate $P_t(o,\theta_0)$ as:

$$P_t(o,\theta_0) = \frac{P(\mathbf{f}^{\text{test}}_{\delta_t}|o,\theta_0+\delta_t)P_{t-1}(o,\theta_0)}{\sum_{o,\theta_0} P(\mathbf{f}^{\text{test}}_{\delta_t}|o,\theta_0+\delta_t)P_{t-1}(o,\theta_0)} \tag{6.5}$$

By default, a uniform prior can be selected for $P_0(o,\theta_0)$. If we are primarily interested in identifying the correct test object, we can further calculate:

$$P_t(o) = \sum_{\theta_0} P_t(o,\theta_0). \tag{6.6}$$

It remains to specify fully the likelihood model. Our model depends on three parameters, p_a, $p_b \in \mathbb{R}$ ($p_a > p_b$), and $M \in \mathbb{N}$:

$$P(\mathbf{f}^{\text{test}}_{\delta}|o,\theta_\delta) \propto \begin{cases} 0 & \text{if } \max_{o',\theta'}(f^{\text{test}}_{\delta}(o',\theta')) \geq M \\ p_a & \text{if } \max_{o',\theta'}(f^{\text{test}}_{\delta}(o',\theta')) < M \text{ and} \\ & (o,\theta_\delta) \in \operatorname{argmax}_{o',\theta'}(f^{\text{test}}_{\delta}(o',\theta')) \\ p_b & \text{otherwise} \end{cases} \tag{6.7}$$

where argmax returns the subset of arguments attaining the maximum value. The parameter M may be set arbitrarily high, and simply allows the model to be normalized (specifying a number of matches that cannot be exceeded). By specifying $p_a > p_b$, we ensure that when we are looking at object o and viewpoint θ, we expect to see feature vectors generated where $f^{\text{test}}_{\delta}(o,\theta)$ takes the maximum value in the vector.

Viewpoint Selection Strategy: The viewpoint selection strategy determines the 'next best viewpoint', δ_{t+1}, that will provide the most relevant information to complete the recognition process in an optimal manner given the previous views we have visited $\delta_{1...t}$, and our current belief about the object/pose, $P_t(o,\theta_0)$. For our viewpoint selection strategy, we use a *uniqueness map* $w_o(\theta)$ which takes values in $\mathbb{R}^+$, specifying the uniqueness of the viewpoint θ for object o on some scale. We specify how this is assigned below. Assuming we have such a uniqueness map, we set:

$$\delta_{t+1} = \text{argmax}_{\delta' \in \mathscr{V} \setminus \{\delta_1 \dots \delta_t\}} \sum_{o,\theta_0} \omega_t(o,\theta_0) \cdot w_o(\theta_0 + \delta') \tag{6.8}$$

Here, the factors $\omega_t(o,\theta)$ allow us to specify how the uniqueness maps are to be combined. We use a simple additive combination based on our current best guess for the orientation of each possible object:

$$\omega_t(o,\theta_0) = \begin{cases} 1 & \text{if } \theta_0 \in \text{argmax}_{\theta_0'} f_{\delta_t}^{\text{test}}(o,\theta_0') > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6.9}$$

To set the uniqueness maps, $w_o(\theta)$, we first build offline a vocabulary tree data structure. Starting with all the extracted SIFT descriptors from the training set, we cluster these using hierarchical K-means clustering where each group of descriptors at a particular level consists of descriptors closest to a particular cluster centre at the level above, which are then clustered to form cluster centres at the current level. Full details of this process can be found in [15]. The vocabulary tree consists of a collection of nodes $\mathscr{N}$, with each node associated with a cluster center (SIFT vector), $c(n) : \mathscr{N} \to \mathbb{R}^{128}$, a root node $\text{rt} \in \mathscr{N}$, parent and children functions $\text{pa}(n) : \mathscr{N} \to \mathscr{N}$, $\text{ch}(n) : \mathscr{N} \to \mathscr{P}(\mathscr{N})$ (with $\mathscr{P}(.)$ the powerset operator) with the usual tree-structure constraints on rt, pa and ch. An arbitrary given SIFT descriptor d is associated with a path from the root to a leaf node in the tree $\mathscr{N}_d \subset \mathscr{N}$ formed by starting at root, and successively adding the child node to $\mathscr{N}_d$ whose cluster center is closest, $n_{i+1} = \text{argmin}_{n \in \text{ch}(n_i)} |c(n) - d|$, where n_i is the node added to $\mathscr{N}_d$ at the i'th step along the path, and $n_1 = \text{rt}$.

For each node n in the tree a TFIDF-like (Term Frequency Inverse Document Frequency) metric is calculated to capture the node's uniqueness:

$$q_n = \ln \frac{N}{N_n} \tag{6.10}$$

where N is the total number of images in the database and N_n is the number images in the database with at least one feature that passes through node n ($N_n = \sum_{o,\theta} [\sum_{i \in \mathscr{I}_{o,\theta}^{\text{train}}} [n \in \mathscr{N}_{d_{o,\theta}^{\text{train}}(i)}] > 0]$).

We use this TFIDF metric to calculate an arbitrary feature's uniqueness. The feature's path through the vocabulary tree is determined as above. A measure of uniqueness is then given by the sum of all the TFIDF numbers, or weights, of the nodes it passes through. The higher the weighting, the more unique we rate the feature. The uniqueness of each viewpoint is then calculated by summing these totals for all the SIFT features extracted from that viewpoint. Hence,

$$w_o(\theta) = \sum_{n \in \mathscr{I}_{o,\theta}^{\text{train}}} \sum_{i \in \mathscr{N}_{d_{o,\theta}^{\text{train}}(n)}} q_i. \tag{6.11}$$

Stopping Criterion: As our stopping criterion, we consider when our belief that we are observing a particular object reaches a certain threshold, τ^{stop}: for the first t at which $\max_o(P_t(o)) > \tau^{\text{stop}}$, we halt, and output $o^* = \text{argmax}(P_t(o))$ and $\theta^* = \text{argmax}_{\theta_0}(P_t(o^\star, \theta_0))$.

6.4 Active Recognition of Multiple Objects

The algorithm outlined in Section 6.3 assumes that we are viewing a single object at test time from a variety of angles. However, in natural scenes we rarely encounter single objects isolated from each other, and more typically see collections of objects which occlude each other, and may contain cluttering objects that we are not trained to recognize. The Bayesian framework presented above is readily adapted to recognize collections of objects (and their orientations) in place of single objects as above.

Problem Statement: We assume that we have access to similar training data to Section 6.3, including uncluttered images of each of the objects we wish to recognize captured from a set of viewing angles. As before, we write $I^{\text{train}}_{o,\theta}$ for the training image of object o at viewing angle θ. Instead of being presented with a single object at test time, we now assume we are viewing a collection of objects, which may include objects from our training set, and unknown objects. Our task is to identify a) which out of the set of known objects are present and how often, and b) for every object present, its orientation with respect to a reference viewpoint. This output may be expressed by an $N_o \times N_\theta$ matrix, Z^*, with entries in $\mathbb{Z}^+$, whose entry $Z^*(o, \theta_0)$ denotes the number of occurrences of object o at orientation θ_0 in the test collection. As in the single object case, we are allowed to capture a sequence of images of the test collection at viewing angles $\delta_1, \delta_2, ... \in \{0°, 20°, 40°, ...340°\}$ (with respect to rotation about the center of the collection) and we treat $\delta_1 = 0°$ as the reference viewpoint to label the orientation of the objects present. As in the single object case, we specify our algorithm in terms of the image representation we use, our update strategy, viewpoint selection strategy and stopping criterion, which are outlined below.

We make two simplifying assumptions in the model presented (which are respected in our experimental data). First, that the camera positions when viewing the test collection are such that all object centers project close to the center of the image (i.e. the collection is not too dispersed), and thus that we do not need to compensate for projection effects when identifying the orientations of objects at different positions in the collection (implying that we have approximately an orthogonal projection over the collection). Second, we assume the same object does not occur more than once at the same orientation, and thus that the matrix Z to be estimated is binary.[1]

[1] Our experimental data in fact allows the stronger assumption that the same object does not occur more than once. This can easily be incorporated, although for simplicity we outline the model here without this assumption, which we did not find to provide significant gains in practice.

Image Representation: We use the same image representation as in the single object recognition case. Hence, given a new test image, I^{test}_{δ}, we calculate the matching descriptor sets $\mathscr{M}^{o,\theta}_{\delta}$ (Equation 6.1) for each (o,θ) pair, and the corresponding Hough-matching scores $H^{o,\theta}_{\delta}$ (Equation 6.3). The feature representation $\mathbf{f}^{\text{test}}_{\delta}$ is again formed by concatenating the Hough scores as in Equation 6.4.

Update Strategy: Since we are interested in estimating a collection of objects and associated poses under the assumption that no object appears multiple times at the same pose, we introduce a binary random variable, $Z_0(o,\theta_0)$ for each (o,θ_0) pair, which will take the value 1 if object o is present in the test collection at orientation θ_0, and 0 otherwise. Making the simplifying assumption that appearances of object/orientation pairs in the test collection are independent, we maintain a separate distribution for each of these binary variables, indicating our belief that object o is present at orientation θ_0 in the test collection given the images observed up to time-step t, which may be expressed: $P_t(z_0(o,\theta_0)) = P_t(z_0(o,\theta_0)|\mathbf{f}^{\text{test}}_{\delta_1},...\mathbf{f}^{\text{test}}_{\delta_t})$, where $z_0(o,\theta_0) \in \{0,1\}$ is the value taken by $Z_0(o,\theta_0)$. We update in parallel each of these distributions using a Bayesian update strategy. As in Section 6.3, we assume that images at different test viewpoints are generated independently given the test collection. We thus require a likelihood model for the generation of a feature vector given a collection of objects at specific offsets. We can express this as $P(\mathbf{f}^{\text{test}}_{\delta}|\{Z_{\delta}(o,\theta_{\delta}) = z_{\delta}(o,\theta_{\delta}), o = 1...N_o, \theta \in N_{\theta}\})$, where we have the deterministic relation $Z_{\delta}(o,\theta_{\delta}) = Z_0(o,\theta_0+\delta)$. We will assume that this likelihood factorizes as follows (and give a form below that does so):

$$P(\mathbf{f}^{\text{test}}_{\delta}|\{z_{\delta}(o,\theta_{\delta}), o = 1...N_o, \theta \in N_{\theta}\}) = \prod_{o,\theta} P(f^{\text{test}}_{\delta}(o,\theta_{\delta})|z_{\delta}(o,\theta_{\delta})). \tag{6.12}$$

This allows us to express the required Bayesian updates as:

$$P_t(z_0(o,\theta_0)) = \frac{P(f^{\text{test}}_{\delta_t}(o,\theta_{\delta_t})|z_{\delta_t}(o,\theta_{\delta_t}))P_{t-1}(z_0(o,\theta_0))}{\sum_{z_0(o,\theta_0)=\{0\,1\}} P(f^{\text{test}}_{\delta_t}(o,\theta_{\delta_t})|z_{\delta_t}(o,\theta_{\delta_t}))P_{t-1}(z_0(o,\theta_0))} \tag{6.13}$$

To estimate the probability that a particular object is present at any orientation, we can evaluate:

$$P_t(z(o)) = 1 - \prod_{\theta_0} P_t(z_0(o,\theta_0) = 0). \tag{6.14}$$

By Equation 6.12, we can express our likelihood model directly in terms of $P(f^{\text{test}}_{\delta}(o,\theta_{\delta})|z_{\delta}(o,\theta_{\delta}))$. For this purpose, we assume we have access to counts (from a sample of validation images with statistics similar to our test data) $\kappa^0_{o,n}$, $\kappa^1_{o,n}$, $\kappa^0_{o,\theta,n}$, $\kappa^1_{o,\theta,n}$, $n = 0...M$, where M is a maximum value used for normalization as in Section 6.3. The counts $\kappa^1_{o,\theta,n}$ indicate how many times we observe a Hough-score of n between a validation image containing viewpoint θ of object o and training image $\mathscr{I}^{\text{train}}_{o,\theta}$, while $\kappa^0_{o,\theta,n}$ indicates how many times we observe a Hough-score of n between such a validation image and all other training images. Similarly, $\kappa^1_{o,n}$ indicates how many times we observe a Hough-score of n between a validation image containing o and any training image containing o regardless of orientation, while $\kappa^0_{o,n}$ indicates how many times we observe a Hough-score of n between such a validation image and training images not containing o. We also introduce a smoothing constant β, and a constant $M' < M$ (which is used to pool together less frequently occurring larger values of n). Our general likelihood model can then be expressed as:

$$P(f^{\text{test}}_{\delta}(o,\theta_\delta)|z_\delta(o,\theta_\delta)) \propto \begin{cases} 0 & \text{if } f^{\text{test}}_{\delta}(o,\theta_\delta) \geq M \\ \lambda^{z_\delta(o,\theta_\delta)}_{o,\theta_\delta,f^{\text{test}}_{\delta}(o,\theta_\delta)} + \beta & \text{if } f^{\text{test}}_{\delta}(o,\theta_\delta) < M' \\ \frac{(\sum_{n=M'...M} \lambda^{z_\delta(o,\theta_\delta)}_{o,\theta_\delta,n}) + \beta}{M-M'} & \text{otherwise} \end{cases} \tag{6.15}$$

We consider two cases. For the first (which we call *likelihood model 1*) we let $\lambda^b_{o,\theta,n} = \kappa^b_{o,n}$, where $b \in \{0\ 1\}$. This allows a different distribution of Hough matching scores for each object, but does not distinguish between different viewpoints. For the second, (*likelihood model 2*) we let $\lambda^b_{o,\theta,n} = \kappa^b_{o,\theta,n}$, hence giving a different distribution of Hough matching scores for each viewpoint and orientation.

Viewpoint Selection Strategy: The same viewpoint selection strategy may be applied as in Section 6.3, since it depends only on the feature vector representation, and not on the probability model. Implicitly, this selection strategy respects an assumption that the same object is not present in the collection at multiple orientations. This is appropriate in our experimental setting, but may not be appropriate in general.

Stopping Criterion: We adapt the stopping criterion of Section 6.3 to cope with the multiple object hypothesis. Here, we fix a value N_{obj}, and stop when at least N_{obj} objects reach a belief of τ^{stop} of being present: for the first t at which $\sum_o [P_t(z(o)) > \tau^{\text{stop}}] \geq N_{\text{obj}}$, we halt, and output Z^*, where $Z^*(o,\theta_0) = 1$ if $P_t(Z_0(o,\theta_0) = 1) > 0.5$ and $Z^*(o,\theta_0) = 0$ otherwise. This stopping criterion implicitly assumes at least N_{obj} objects will be present in a collection, which is appropriate in our experimental setting, but may not be in general.

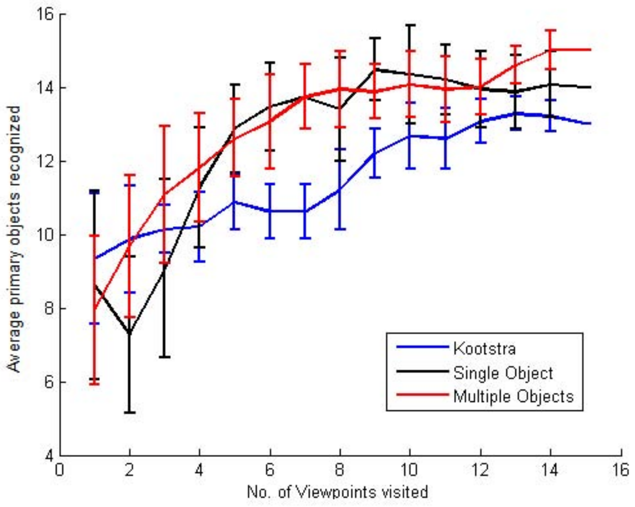


Fig. 6.1 Comparing Single and Multiple Object Recognition Algorithms:Figure 6.1 shows the average number of primary objects recognized after a given number of viewpoints for our single and multi-object algorithms, and the approach of Kootstra *et al.* [10]

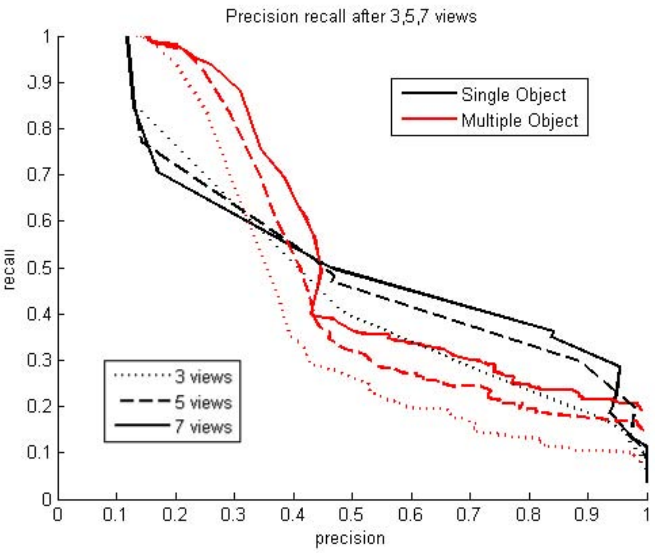


Fig. 6.2 This graph plots the average precision and recall curves for both primary and secondary objects for our single and multi-object algorithms after 3,5 and 7 viewpoints respectively

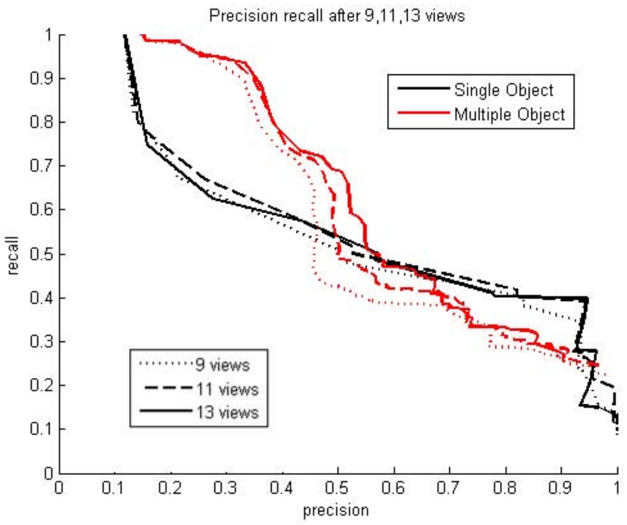


Fig. 6.3 This graph displays the average precision and recall curves for both primary and secondary objects for our single and multi-object algorithms after 9,11 and 13 viewpoints respectively

6.5 Relationship to Mutual Information

We briefly discuss here the relationship between our uniqueness map viewpoint selection algorithm introduced in Section 6.3 and an approach to viewpoint selection based on mutual information. As we note, our approach is typically more efficient in terms of complexity.

The use of mutual information for data selection in active sensing tasks has been proposed by [5], and again more recently in [18] and [11]. In the terms of our multiple object problem, this approach would direct us to select the next test viewpoint on the basis of which has the highest mutual information with the random variables $Z_0(o,\theta_0)$ we are interested in. Hence, we search for:

$$\delta_{t+1} = \text{argmax}_{\delta' \in \mathscr{V} \setminus \{\delta_1 \ldots \delta_t\}} MI(\mathbf{f}^{\text{test}}_{\delta'}; \mathbf{z}_0) \tag{6.16}$$

where the mutual information is defined as:

$$MI(\mathbf{f}^{\text{test}}_{\delta'}; \mathbf{z}_0) = H(\mathbf{z}_0) - H(\mathbf{z}_0 | \mathbf{f}^{\text{test}}_{\delta'}) \tag{6.17}$$

with $H(.)$ the Shannon entropy, and $H(.|.)$ the conditional entropy. Given that $H(\mathbf{z}_0)$ is independent of δ', maximizing Equation 6.16 is equivalent to minimizing the conditional entropy:

$$\delta_{t+1} = \underset{\delta' \in \mathscr{V} \setminus \{\delta_1 \ldots \delta_t\}}{\text{argmin}} H(\mathbf{z}_0 | \mathbf{f}^{\text{test}}_{\delta'}) \tag{6.18}$$

Given the factorization of the likelihood function in Equations 6.12 and 6.15, we can evaluate the conditional entropy as:

$$\begin{aligned} H(\mathbf{z}_0|\mathbf{f}_\delta^{\text{test}}) &= -\sum_{\mathbf{f}_\delta^{\text{test}},\mathbf{z}_0} P(\mathbf{z}_0)P(\mathbf{f}_\delta^{\text{test}}|\mathbf{z}_\delta)\cdot\log(P(\mathbf{z}_\delta|\mathbf{f}_\delta^{\text{test}})) \\ &= -\sum_{o,\theta_0}\sum_{f_\delta^{\text{test}}(o,\theta_\delta),z_0(o,\theta_0)} P(z_0(o,\theta_0))\cdot \\ &\quad P(f_\delta^{\text{test}}(o,\theta_\delta)|z_\delta(o,\theta_\delta))\cdot \\ &\quad \log(P(z_\delta(o,\theta_\delta)|f_\delta^{\text{test}}(o,\theta_\delta))) \end{aligned} \tag{6.19}$$

For Equation 6.15 with likelihood model 1 ($\lambda_{o,\theta,n}^b = \kappa_{o,n}^b$), we have that

- $P(f_{\delta'}^{\text{test}}(o,\theta_{\delta'})|z_{\delta'}(o,\theta_{\delta'})) = P(f_{\delta''}^{\text{test}}(o,\theta_{\delta''})|z_{\delta''}(o,\theta_{\delta''}))$ when
- $f_{\delta'}^{\text{test}}(o,\theta_{\delta'}) = f_{\delta''}^{\text{test}}(o,\theta_{\delta''})$ and
- $z_{\delta'}(o,\theta_{\delta'}) = z_{\delta''}(o,\theta_{\delta''})$,

and thus *MI* cannot be used with this model as all viewpoints will give rise to the same conditional entropy. A similar problem affects the single object model in Section 6.3, since Equation 6.7 is symmetrical across o and θ. However, since Equation 6.15 with likelihood model 2 ($\lambda_{o,\theta,n}^b = \kappa_{o,\theta,n}^b$) results in distinct matching score distributions for each object/orientation combination, the *MI* selection strategy above can be applied with this model.

The fact that we can only apply an *MI* selection strategy when we have distinct distributions for each viewpoint highlights the reliance of the *MI* strategy on extensive training/validation statistics. By contrast, our uniqueness map can be used in cases where we do not estimate these. Further, the evaluation of the required conditional entropies in Equation 6.18 is $O(N_o N_\theta K)$, where K is the complexity of evaluating the expectation across the feature space (in general $K = |\mathscr{F}|$, where $\mathbf{f}_\delta^{\text{test}} \in \mathscr{F}$, and for our likelihood model 2 we have $K = M'$ due to the form of Equation 6.15). In contrast, our uniqueness map requires only $O(N_O N_\theta)$ using the selection rule described in Equations 6.8 and 6.9. We note though that, if accurate probability models are available for each viewpoint, the *MI* selection rule is optimal in the sense of achieving the lowest expected misclassification loss for a given number of viewpoints [5].

6.6 Experimentation

Dataset: For our experiments, we use the active recognition dataset introduced by [7]. The training data consists of 23 everyday objects such as cereal boxes, ornaments, and spice bottles. The full list of training objects is: All bran box, Battery, Can 1, Can 2, Curry 1, Curry 2, Elephant, Handbag 3, Jewelry box 1, Jewelry box 2, Lemon bottle, Mr Min, Robotcop, Salad, Sauce 1, Sauce 2, Spice 1, Spice 2, Spice 3, Spray can, Teddy, Toy and Wall E. Images were captured at every 20 degrees for each object against a plain background on a turntable using a Prosilica GE1900C

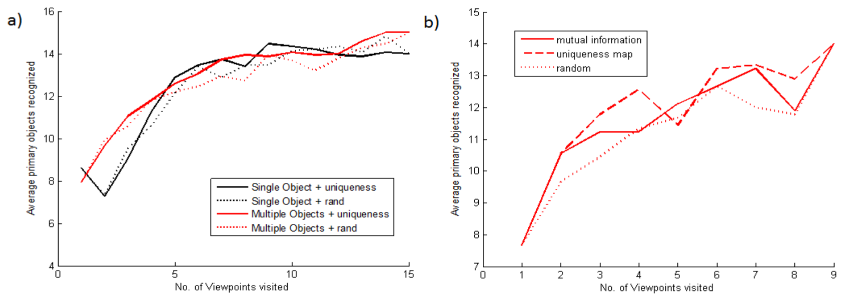


Fig. 6.4 Comparing Viewpoint Selection Algorithms 1: Graph (a) shows the average number of primary objects recognized for our single and multi-object algorithms, comparing the performance using the uniqueness map viewpoint selection strategy, and the performance with a random selection strategy. Graph(b) compares performance of our multi-object algorithm with likelihood model 2 using the uniqueness map, a random, and a mutual information-based viewpoint selection strategy.

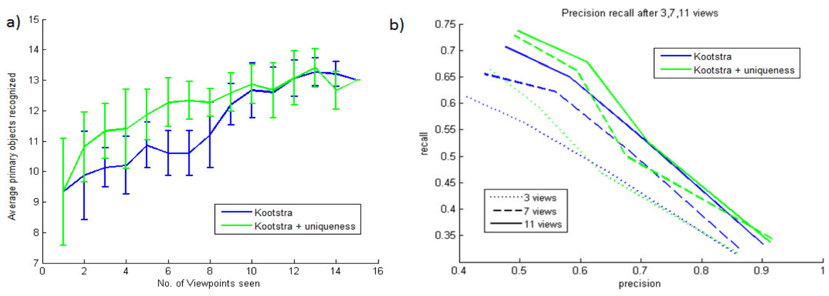


Fig. 6.5 Comparing Viewpoint Selection Algorithms 2: Graph (a) shows the average number of primary objects recognized for the approach of Kootstra *et al.* [10], comparing the performance using the original expected activation viewpoint selection strategy (from [10]), and with our uniqueness map strategy substituted. Graph(b) compares average precision and recall curves across all primary and secondary objects for the same methods when highest ranking 1-4 objects are selected.

camera. This means that there were 18 training images captured for each object in the database. Example images from the training set are shown in Figure 6.6.

For the test set, the same objects used in the training data were captured at every 20 degrees in a cluttered environment with significant occlusion by other objects. A 'primary' object was placed in the centre of the turntable with 'secondary' objects, which or may not belong to the training set, surrounding it. The distracter objects include other everyday objects such as towel, pencil box, thyme bottle and bracelet. Example images from the test set are shown in Figure 6.7. These images are used in the experiments for testing either the performance against a single object hypothesis, where we desire recognition of the central object, or multiple objects hypotheses, where we

Table 6.1 Comparing Timing and Stopping Criteria: The table compares average performance of various methods with different stopping criteria. Shown are: the average number of viewpoints, average time taken (s), average primary object score, average precision/recall across primary and secondary objects when taking the top ranked object, and the top 4 ranked objects, and the stopping criterion applied.

Method	Views	time(s)	score	prec(1)	recall (1)	prec(4)	recall(4)	Stopping criteria
Kootstra	6.2	587.1	12.3	0.901	0.334	0.430	0.639	stopping ratio 1.5
Kootstra+our viewpoint	6.7	636.5	12.6	0.901	0.334	0.444	0.659	stopping ratio 1.5
Ours (single)	8.9	439.7	14.1	0.919	0.341	0.354	0.525	threshold 0.5
Ours(single)	11.6	573.0	14.1	0.916	0.340	0.376	0.558	threshold 0.8
Ours (multiple)	9.7	479.2	13.5	0.835	0.310	0.509	0.756	threshold 0.5, 2 objs
Ours (multiple)	11.0	543.4	13.6	0.844	0.313	0.530	0.786	threshold 0.6, 2 objs

desire recognition of all objects appearing in the training set. For both training and test data, images are captured around the y-axis, which represents 1 DoF.

Experiment 1: Comparing Single and Multiple Object Recognition Algorithms Our first experiment compares the performance of the two algorithms outlined, using single and multiple object hypotheses. We consider two tasks: recognizing the primary objects in the test sequences, and recognizing all objects. For the primary object task, we take the object with highest probability for both single and multiple object hypotheses ($\text{argmax}_o P_t(o)$ and $\text{argmax}_o P_t(z(o))$ respectively). For the all object task, we generate precision recall curves by thresholding both the single object and multi-object posteriors ($P_t(o)$ and $P_t(z(o))$). We note that these operations are valid probabilistically only for the primary-task/single-object and all-object-task/multi-object combinations respectively. We also compare the performance of our algorithms with the method of Kootstra *et al.* on the primary object task. For the multi-object algorithm we use the likelihood 1 model. We select 3 evenly spaced viewpoints (5, 11 and 17) from all test sequences to form the validation set from which to record the counts $\kappa_{o,n}^{\{0,1\}}$ used in this model. We restrict all models to the 15 remaining viewpoints at test time for a fair comparison. We record the performance of the algorithms on both tasks after 1...15 viewpoints, and we average across all 15 starting viewpoints for all performance measures to generate a robust comparison. Results are shown in Figures 6.1 6.2 and 6.3 (error bars are shown at 1 standard deviation).

As shown in Figure 6.1, both the single and multiple object algorithms outperform Kootstra's method on the primary-object recognition task, validating our claim that the geometric matching step of our algorithm is important in cluttered scenes. The single and multiple object hypothesis models perform comparably here, which seems to indicate that the bias towards the primary object is similar in both these methods: as more data is seen, the single object model latches onto the primary object hypothesis in over half the cases , while this hypothesis becomes the highest scoring in the multi-object model. Figures 6.2 and 6.3 show that the two models have different performance characteristics, though, when we consider recognition of all objects present (the second task). The single object hypothesis approach generally has higher precision when the recall is low i.e. < 0.5. The multiple object hypothesis approach, though, has higher recall when the precision is low. This appears to indicate that we can only take advantage of the more accurate probabilistic model in the lower precision range on our test data. The single object model looses out in this range, as it will tend to suppress hypotheses if they are significantly weaker than the most dominant hypothesis. In contrast, this behaviour seems to help in the high precision range, since it is more likely to suppress spurious hypotheses. As shown by comparing Figures 6.2 and 6.3, the latter effect becomes less pronounced as more data are gathered, and the range in which the multi-object model dominates expands.

Experiment 2: Comparing Viewpoint Selection Algorithms
Our second experiment runs several tests to compare our proposed viewpoint selection strategy based on the uniqueness map with alternative strategies.

First, we test this strategy against a random selection mechanism (simply choosing at random one of the remaining viewpoints at each time-step). We use the same set-up as above, and compare the primary object recognition rate for single and multi-object algorithms with uniqueness map and random viewpoint selection rules in all combinations. As shown in Figure 6.4(a), the single and multiple object hypothesis approaches using our viewpoint selection algorithm outperform those using random selection across a large range of viewpoints.

Second, we test our uniqueness map strategy against random selection and mutual information using the multi-object algorithm with likelihood model 2. The set-up is as above, but we now select all even numbered viewpoints as our validation set to gather the statistics $\kappa^{\{0,1\}}_{o,\theta,n}$, and treat the remaining 9 odd numbered viewpoints as our test set (we let $\kappa^{\{0,1\}}_{o,\theta,n} = 0.5 * \sum_{m=\{n-1\; n+1\,\mathrm{mod} N_\theta\}} \kappa^{\{0,1\}}_{o,\theta,m})$ for odd n. As noted, these more extensive statistics (compared to likelihood model 1) are necessary in order to evaluate the mutual information per viewpoint. Figure 6.4(b) com-

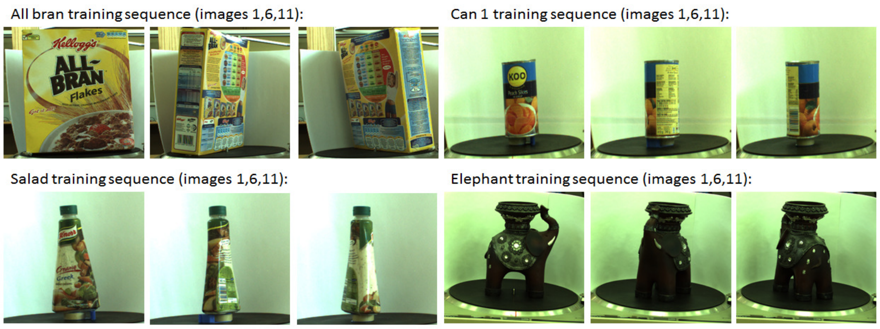

All bran training sequence (images 1,6,11):
Can 1 training sequence (images 1,6,11):
Salad training sequence (images 1,6,11):
Elephant training sequence (images 1,6,11):

Fig. 6.6 Example images from our training set

pares the performance of likelihood model 2 with 3 viewpoint selection rules: random, mutual information and our uniqueness map. As shown, both the uniqueness map and mutual information outperform random selection across most viewpoints. The graph also shows that our uniqueness map performs competitively with mutual information, and in fact outperforms it on a range of viewpoints. This is despite the fact that the mutual information can be shown to be the optimal strategy if the probabilistic model is accurate [5]. The low performance of mutual information may thus be taken to indicate that insufficient data was available to estimate accurate statistics for likelihood model 2 in this setting (indeed, the performance is in general comparable to likelihood model 1, as can be seen by comparing graphs (a) and (b), despite having access to more fine-grained validation statistics). The results however generally support our claim that the uniqueness map provides an effective alternative to previous viewpoint selection mechanisms, which is robust to inaccuracies in the underlying model.

Finally, we test our uniqueness map strategy in a non-Bayesian context against the expected increase in activation strategy used in [10]. We substitute the weighting:

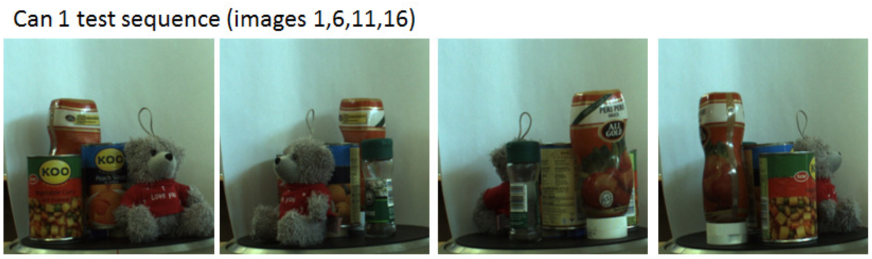
$$\omega_t(o,\theta_0) = \begin{cases} 1 & \text{if } \theta_0 = \text{argmax}_{\theta_0'} a^{\text{test}}_{\delta_t}(o,\theta_0') \\ 0 & \text{otherwise.} \end{cases} \tag{6.20}$$

for Equation 6.9, where $a^{\text{test}}_{\delta_t}(o,\theta_0)$ is the activation for object o at orientation θ_0 and time t in the method of [10]. Figure 6.5 compares the performance of [10] with the original expected activation selection mechanism, and with our proposed mechanism substituted. Graph (a) shows the average number of primary objects identified, and graph (b) shows the precision/recall curves for all objects present at a range of viewpoints. In (b), the curves are generated from 4 points, found by successively selecting the 1-4 objects with the highest ranked activations summed across viewpoints. As shown, our viewpoint selection mechanism performs better across most viewpoint ranges than the expected activation in graph (a). In graph (b) when the recall is higher than approximately 0.5, [10] with our viewpoint selection algorithm has higher precision. When the recall falls below 0.5, the precision values for both methods are comparable. The results thus validate our method's effectiveness in a non-Bayesian context.

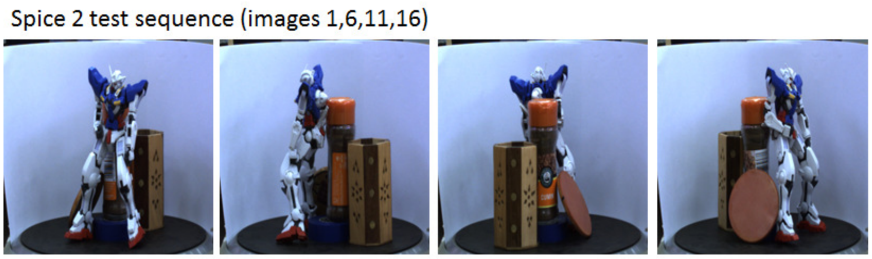
Experiment 3: Comparing Timing and Stopping Criteria

In our final experiment, we compare the performance of our single and multi-object algorithms along with the method of Kootstra [10] under various settings of the stopping condition. In general, we can use the stopping criteria to manipulate the trade-off between the overall accuracy of each algorithm and its expected overall timing. Table 6.1 gives a number of performance measures for all algorithms under a variety of settings of the stopping criterion (the criterion for [10] is the ratio between largest and second largest activations, and for our methods we manipulate τ^{stop} for both methods and N_{obj} for the multi-object method). The timings for [10] are derived from our own implementation of the method. The performance measurements are averaged across all 15 starting viewpoints, with ‘score’ being the average number of objects recognized in the primary object setting, and precision/recall values given when the top 1 and 4 ranked objects are selected.

As shown, we can achieve similar or better performance metrics to [10] at faster absolute timings, although in general our methods use a larger average number of viewpoints to achieve this. Further, in agreement with the results of Experiment 1, we see that our single object method has slightly better performance than the multi-object method in the high-precision/low-recall range, while the multi-object model is substantially better in the low(er)-precision/high-recall range (as seen by comparing the results across methods in the prec(4) and rec(4) columns). We note



Ground Truth	Kootstra	Ours (Single Object)	Ours (Multi Object)
can1	sauce1: 1.1781 +	can2: 0.51366 +	can2: 0.98922 +
can2	can2: 0.9814 +	sauce1: 0.032653 +	can1: 0.93986 +
sauce1	spice3: 0.74646 -	robocop: 0.028842 -	sauce1: 0.71543 +
teddyBear	handbag3: 0.57768 -	sprayCan: 0.028831 -	sauce2: 0.59597 -
	teddyBear: 0.53794 +	salad: 0.024364 -	teddyBear: 0.50077 +
	robocop: 0.53436 -	Toy: 0.023257 -	jewelryBox2: 0.23187 -
	time: 291.3333	time: 442.309	time: 470.2307



Ground Truth	Kootstra	Ours (Single Object)	Ours (Multi Object)
robocop	elephant: 5.8869 -	spice2: 0.45797 +	spice2: 0.95604 +
spice2	robocop: 5.6327 +	sprayCan: 0.049506 -	robocop: 0.8573 +
	battery: 5.4467 -	salad: 0.046528 -	can1: 0.73769 -
	Toy: 5.2443 -	elephant: 0.035167 -	salad: 0.58172 -
	time: 1425	time: 442.309	time: 470.2307

Fig. 6.7 Example images from our testing set. Shown below are the ground truth objects in the sequence, the highest ranking objects predicted by our single object and multi-object algorithms, along with the approach of Kootstra *et al.* [10] using the stopping criteria from rows 1, 3 and 5 of Table 6.1. The final probabilities/scores of the predicted objects are shown, along with +/- to indicate if the object is present or not, and the time taken (in seconds) to process the sequence.

that these results are only suggestive, since they may be affected by our particular implementations, and also in a realistic active vision situation factors other than processing time may be involved (such as time to move to a new viewpoint). Outputs corresponding to rows 1, 3 and 5 for two test sequences are shown in Figure 6.7.

6.7 Discussion

In summary, we have investigated active object recognition in the context of identifying groups of objects in cluttered scenes. We have shown that features formed from Hough-based geometric matching counts provide sufficient information to perform object recognition in this context, and that using geometric information allows us to achieve better performance against methods such as [10] which do not. Further, we have developed a Bayesian framework which accurately models the assumptions of this testing context, and have shown that providing such an accurate probabilistic model can provide enhanced performance in certain circumstances. Finally, we have provided an extensive empirical evaluation of a multi-object version of the uniqueness map viewpoint selection strategy introduced in [7], and shown this to provide an efficient and accurate alternative to strategies such as mutual information, and non-Bayesian approaches.

In general, our results suggest that techniques for coping with object recognition amongst clutter in 2D can be used effectively in an active vision context. Indeed, our results show that the added robustness of the active vision setting allows simpler overall representations to be used, as seen by comparing our count-based features with the approach of [12]. The incorporation of additional statistics into our representation, as in [12], as well as more complex object models, including for instance texture and shape, are promising directions for future work. Further, our results suggest that further refining our overall probabilistic models to accurately reflect the assumptions of the test data can provide performance gains in a Bayesian active vision setting. A slight caveat here, though, is that we must have access to sufficient training/validation data in order to accurately estimate such models to see significant effects. Possibilities for future work along these lines include investigation of probabilistic active vision models which incorporate knowledge of correlations between objects (which objects are seen often/seldom together) and enhanced occlusion reasoning.

References

1. Borotschnig, H., Paletta, L., Prantl, M., Pinz, A.: Active object recognition in parametric eigenspace. In: British Machine Vision Conference (BMVC), pp. 629–638 (1998)
2. Borotschnig, H., Paletta, L., Prantl, M., Pinz, A.: A comparison of probabilistic, possibilistic and evidence theoretic fusion schemes for active object recognition. Computing, 293–319 (1999)

3. Callari, F., Ferrie, F.: Active object recognition: Looking for differences. International Journal of Computer Vision, 189–204 (2001)
4. Collet, A., Berenson, D., Srinivasa, S., Ferguson, D.: Object recognition and full pose registration from a single image for robotic manipulation. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 48–55 (2009)
5. Denzler, J., Brown, C.M.: Information theoretic sensor data selection for active object recognition and state estimation. IEEE Transactions on PAMI, 145–157 (2002)
6. Ferrari, V., Tuytelaars, T., Gool, L.: Simultaneous object recognition and segmentation from single or multiple views. International Journal of Computer Vision, 159–188 (2006)
7. Govender, N., Claassens, J., Torr, P., Warrell, J.: Active object recognition using vocabulary trees. IEEE Workshop on Robot Vision (2013)
8. Hutchinson, S.A., Kak, A.C.: Planning sensing strategies in a robot work cell with multisensor capabilities. IEEE Transactions on Robotics and Automation, 765–783 (1989)
9. Jia, Z.: Active view selection for object and pose recognition. In: International Conference on Computer Vision (ICCV) 3D Object Recognition Workshop, 641–648 (2009)
10. Kootstra, G., Ypma, J., de Boer, B.: Active exploration and keypoint clustering for object recognition. In: IEEE International Conference on Robotics and Automation, 1005–1010 (2008)
11. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in Gaussian Processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research, 235–284 (2008)
12. Lowe, D.: Local feature view clustering for 3d object recognition. In: International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 682–688 (2001)
13. Lowe, D.: Distinctive image features from scale invariant keypoints. International Journal of Computer Vision, 91–110 (2004)
14. Moreels, P., Perona, P.: Evaluation of feature detectors and descriptors based on 3d objects. International Journal of Computer Vision, 263–284 (2007)
15. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2161–2168 (2006)
16. Sabatta, D., Scaramuzza, D., Siegwart, R.: Improved appearance-based matching in similar and dynamic environments using a vocabulary tree. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1008–1013 (2010)
17. Selinger, A., Nelson, R.: Appearance-based object recognition using multiple views. In: Conference on Computer Vision and Pattern Recognition (2001)
18. Sommerlade, E., Reid, I.: Information-theoretic active scene exploration. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
19. Yu, S., Krishnapuram, B., Rosales, R., Rao, R.: Active Sensing. In: IEEE International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 639–646 (2009)

Chapter 7
Incremental Light Bundle Adjustment: Probabilistic Analysis and Application to Robotic Navigation

Vadim Indelman and Frank Dellaert

Abstract. This paper focuses on incremental light bundle adjustment (iLBA), a recently introduced [13] structureless bundle adjustment method, that reduces computational complexity by algebraic elimination of camera-observed 3D points and using incremental smoothing to efficiently optimize only the camera poses. We consider the probability distribution that corresponds to the iLBA cost function, and analyze how well it represents the true density of the camera poses given the image measurements. The latter can be exactly calculated in bundle adjustment (BA) by marginalizing out the 3D points from the joint distribution of camera poses and 3D points. We present a theoretical analysis of the differences in the way that light bundle adjustment and bundle adjustment use measurement information. Using indoor and outdoor datasets we show that the first two moments of the iLBA and the true probability distributions are very similar in practice. Moreover, we present an extension of iLBA to robotic navigation, considering information fusion between high-rate IMU and a monocular camera sensor while avoiding explicit estimation of 3D points. We evaluate the performance of this method in a realistic synthetic aerial scenario and show that iLBA and incremental BA result in comparable navigation state estimation accuracy, while computational time is significantly reduced in the former case.

7.1 Introduction

Bundle adjustment (BA) plays a key role in many applications in mobile vision and robotics. The basic problem can be described as follows: given a sequence of images, determine the maximum a posteriori (MAP) estimate of camera poses and the observed 3D points (or another representation of the observed structure). Fast and reliable calculation of this MAP estimate is important in particular in mobile robotic

Vadim Indelman · Frank Dellaert
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA
e-mail: {indelman,dellaert}@cc.gatech.edu

applications and has been continuously addressed by the research community in recent years.

This paper[1] focuses on the recently developed incremental light bundle adjustment (iLBA) approach [13], that belongs to "structure-less" BA methods [34, 31, 10, 13], in which the camera poses are optimized without including structure parameters into the iterative optimization procedure. iLBA reduces computational complexity by algebraic elimination of 3D points and using an efficient incremental optimization that is based on incremental smoothing (iSAM2) [17, 18].

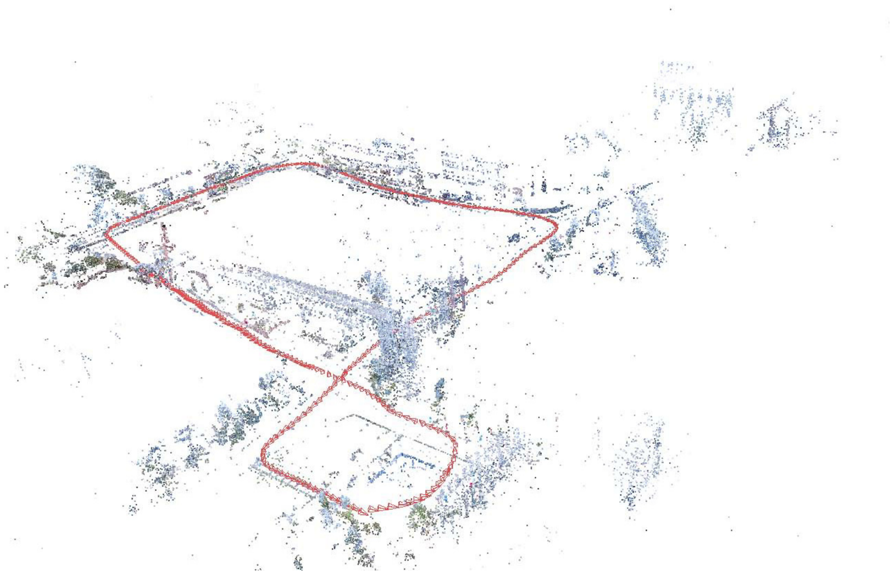
This paper presents a probabilistic analysis of iLBA, analyzing how well the probability distribution corresponding to the iLBA cost function agrees with the true probability distribution of the camera poses. Accurate and reliable maximum a posteriori and uncertainty estimates are important in many structure from motion and robotic applications, yet to the best of our knowledge this is the *first* time that such an analysis is conducted for structure-less BA methods. This theoretical analysis, which is also valid for other structure-less BA methods, reveals the root effects that cause the iLBA distribution to be an approximation of the true distribution. Using real imagery datasets (see Figures 7.1 and 7.2) we show that in practice the two distributions are close to each other.

In the second part of the paper an extension of iLBA to robotic navigation is presented. We argue that iLBA is in particular suitable for navigation problems, as it facilitates estimation of the navigation state without explicit 3D reconstruction, which is typically not required in the navigation context. Furthermore, it supports loop closure observations (re-observations of 3D points) that are essential for maintaining accurate performance over time. We consider state estimation in the challenging configuration of a monocular camera and high-rate inertial navigation sensors (IMU) and use incremental smoothing to fuse information from these sensors. Similarly to [16], we adopt a recently developed technique [26] for IMU pre-integration to summarize IMU information, significantly reducing the number of variables in the optimization. Finally, we analyze the performance of the proposed method in a synthetic aerial scenario.

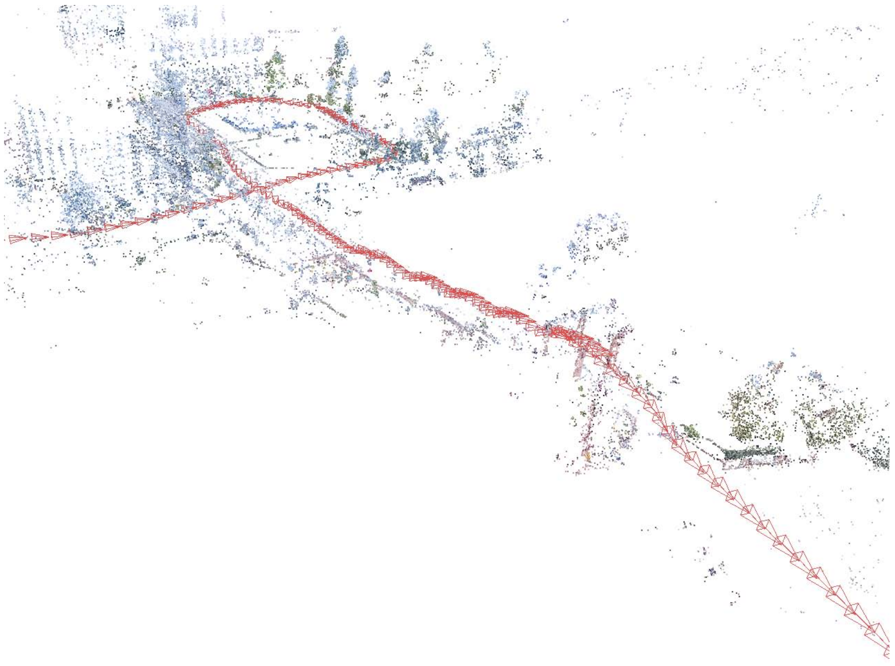
Consequently, this paper makes three contributions: 1) probabilistic analysis of iLBA; 2) extension of iLBA to robotic navigation; 3) Evaluation of thereof using real-imagery datasets and a synthetic aerial scenario.

The remainder of this paper is organized as follows. After discussing related work, Section 7.3 overviews the main component of iLBA. Section 7.4 presents a probabilistic analysis of iLBA and evaluation using real-imagery datasets. In Section 7.5 iLBA is extended to robot navigation; this section also includes performance evaluation in a realistic synthetic aerial scenario. Section 7.6 concludes the discussion and suggests directions for future research.

[1] Part of the material discussed in this paper was presented in the conference papers [14] and [12].



(a) Top view



(b) Zoom in on the stairway area

Fig. 7.1 Estimated camera trajectory and sparse 3D reconstruction in the *Outdoor* dataset that is used for evaluating LBA and BA probability distributions

Fig. 7.2 Estimated camera poses and sparse 3D reconstruction in the *Indoor* dataset that is used for evaluating LBA and BA probability distributions

7.2 Related Work

We organize this section into two parts, discussing first research related to computational efficient bundle adjustment and then reviewing related methods from simultaneous localization and mapping (SLAM) and vision-aided navigation literature.

7.2.1 Computationally Efficient Bundle Adjustment

Development of computationally efficient bundle adjustment approaches, in particular for large-scale scenarios, has become an active research area in the past few years. The developed approaches include methods that exploit sparsity of the involved matrices in the optimization [24, 21], decoupling the BA problem into several submaps that can be efficiently optimized in parallel [30], constructing a skeletal graph using a small subset of images and incorporating the rest of the images using pose estimation [33], solving a reduced version of the non-linear system [22], and finding a coarse initial solution using a discrete-continuous optimization followed by a BA refinement [2].

Another family of recently suggested methods is structure-less BA [34, 31, 10, 13], in which the camera poses are optimized without including structure parameters into the iterative optimization procedure. The first structure-less BA method was introduced, to the best of our knowledge, by Steffen et al. [34] who optimized the corrections of image observations subject to satisfying trifocal tensor constraints [7]. A similar concept was developed in [10] using three-view constraints [11] instead of the trifocal tensor. Rodrguez et al. [31] obtained reduced computational complexity by reformulating the optimized cost function and refraining from correcting the pixels. Another significant gain in computational complexity was obtained in incremental light bundle adjustment [13], that applied a recently developed technique for incremental smoothing [17, 18] to structure-less BA.

7.2.2 SLAM and Vision-Aided Navigation

Current SLAM algorithms can be divided into two main categories: feature- and view-based SLAM. In feature-based SLAM, both the observed 3D points and the robot's past and current poses are optimized. Several efficient optimization methods that exploit the sparsity of typical structure from motion and SLAM problems have been developed in recent years, some of which are discussed in Section 7.2.1.

The second SLAM category is view-based SLAM [25, 4], or pose-SLAM, in which, similar to iLBA, only the current and past robot's poses are maintained. In pose-SLAM approaches, pairs of poses are linked using relative pose constraints that are straightforward to estimate in a stereo camera setup [8, 22], but become more challenging when relying only on a single camera. In the latter case, the relative constraints can be estimated only up to a scale, which encodes the magnitude of the relative translation [4]. This scale parameter can be set based on the previous frames as in [1]. However, to avoid scale drift the scale parameters should be part of the optimization as well [6]. In contrast to conventional pose-SLAM, iLBA formulates multi-view geometry constraints for *each* feature match, thereby not requiring uncertainty estimates from an intermediate (and separate) process of image-based relative-pose constraints estimation.

In the context of navigation-aiding, despite the close relation to SLAM, only a few methods have been presented in recent years that are capable of incorporating loop closure measurements. These include [28] where visual observations are incorporated into the navigation solution using an EKF formulation with a sliding window of past poses. In a later work [29], the authors applied a conventional batch BA that involved explicit structure estimation in order to handle loop closure measurements. More recently, incremental smoothing [18] was proposed for inertial navigation systems in [15, 16] and a method was developed to incorporate loop closures while maintaining a real time navigation solution [20]. The extension of iLBA to robotic navigation that is described in this paper is formulated within the same framework of [20, 15, 16] but replaces the explicit estimation of 3D points with a set of 2-view and 3-view constraints.

7.3 Incremental Light Bundle Adjustment

Incremental light bundle adjustment (iLBA) [13] combines the following two key-ideas: algebraic elimination of 3D points, and incremental smoothing. In this section we review each of these concepts, first introducing notations and standard bundle adjustment formulation.

7.3.1 Bundle Adjustment

Consider a sequence of N views observing M 3D points, and denote the i^{th} camera pose by x_i and the measured image observation of the j^{th} 3D point l_j by z_i^j. Let also $X \doteq \left\{x_1^T,\ldots,x_N^T\right\}^T$ and $L \doteq \left\{l_1^T,\ldots,l_M^T\right\}^T$.

The joint pdf $p\left(X,L|Z\right)$ can be explicitly written in terms of the prior information and the actual measurement models:

$$p\left(X,L|Z\right) = priors \cdot \prod_i \prod_{j\in\mathscr{M}_i} p\left(z_i^j|x_i,l_j\right), \tag{7.1}$$

where $p\left(z_i^j|x_i,l_j\right)$ is the measurement model corresponding to the probability density of observing the 3D point l_j from a camera pose x_i at the pixel location z_i^j, and $\mathscr{M}_i$ represents the indices of all 3D points observed from the ith camera. Assuming Gaussian distributions, the maximum a posteriori (MAP) estimation

$$X^*,L^* = \arg\max_{X,L} p\left(X,L|Z\right),$$

corresponds to minimizing the following nonlinear cost function (omitting prior terms for clarity)

$$J_{BA}\left(X,L\right) = \sum_i \sum_{j\in\mathscr{M}_i} \left\|z_i^j - \pi\left(x_i,l_j\right)\right\|_\Sigma^2, \tag{7.2}$$

where $\pi\left(\cdot\right)$ is the projection function [7] for a standard pinhole camera model, and $\|a\|_\Sigma^2 \doteq a^T\Sigma^{-1}a$ is the squared Mahalanobis distance with the measurement covariance matrix Σ. Each term in the cost function J_{BA} corresponds to the re-projection error between the measured and predicted image observations (see Figure 7.3).

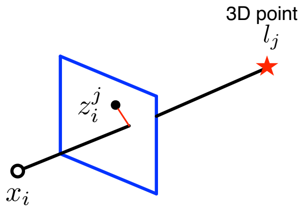


Fig. 7.3 Illustration of re-projection error: difference between projection of the 3D point l_j onto the camera frame using camera pose x_i, and the image observation z_i^j

7.3.2 Algebraic Elimination of 3D Points Using Three-View Constraints

Performing inference over the joint pdf $p(X,L|Z)$ involves optimizing $6N+3M$ variables, with N and M denoting camera frames and observed 3D points, respectively. Instead, in this section we reduce the number of variables to only $6N$ by algebraically eliminating all the 3D points.

Considering all the camera frames that observe some 3D point l_j and writing down all the appropriate projection equations, it is possible to algebraically eliminate l_j, leading to multiple view constraints that involve up to triplets of cameras [27, 35]. One possible formulation of these constraints, recently developed in the context of vision-aided navigation [9, 11], is the three-view constraints. The close relation between these constraints and the well-known trifocal tensor was reported in [13]. The three view constraints, for a triplet of camera frames k,l and m are given by (see Figure 7.4)

$$g_{2v}\left(x_k,x_l,z_k^j,z_l^j\right) = q_k \cdot (t_{k\to l} \times q_l) \tag{7.3}$$

$$g_{2v}\left(x_l,x_m,z_l^j,z_m^j\right) = q_l \cdot (t_{l\to m} \times q_m) \tag{7.4}$$

$$g_{3v}\left(x_k,x_l,x_m,z_k^j,z_l^j,z_m^j\right) = (q_l \times q_k)\cdot(q_m \times t_{l\to m}) - (q_k \times t_{k\to l})\cdot(q_m \times q_l) \tag{7.5}$$

where q_i is the line of sight expressed in a global frame, $q_i \doteq R_i^T K_i^{-1} z$, for any view i and image observation z, K_i is the calibration matrix of this view, R_i represents the rotation matrix from the global frame to the i^{th} view's frame, and $t_{i\to j}$ denotes the translation vector from view i to view j, expressed in the global frame. The first two constraints are the two-view constraints g_{2v} between appropriate pairs of views, also known as the epipolar constraints. Given matching image observations,

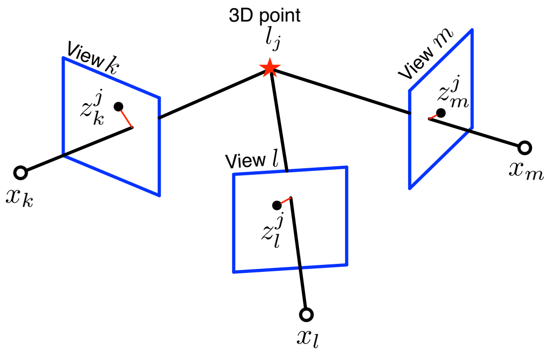


Fig. 7.4 Three camera frames k,l and m observing the same 3D point l_j. Explicit estimation of l_j can be avoided by algebraic elimination using three-view constraints (7.3)-(7.5).

these constraints allow to recover the relative rotation and up-to-scale translation motion between camera pairs. The third constraint, g_{3v}, involves all the three views and enforces a consistent scale of the translation vectors $t_{k\to l}$ and $t_{l\to m}$. Thus, if the magnitude of the former is known, the magnitude of the later can be determined.

When a 3D point is observed by more than three views, a single two-view and three-view constraint between each new view m and past views k,l is added [10]. Determining which past views to use is still an open question that we currently investigate. In the results reported in this paper, we heuristically choose the past views to be the earliest camera frame k observing the 3D point and set l to be the middle camera frame such that the translation vectors $t_{k\to l}$ and $t_{l\to m}$ are of similar magnitudes[2].

In practice, in order to avoid the trivial solution of zero translation, we normalize each of the constraints g_{2v} and g_{3v} by a translation vector and modify the Jacobian matrices accordingly.

Algebraically eliminating all the 3D points using the three-view constraints (7.3)-(7.5) leads to the following cost function that is expressed in terms of these constraints, instead of re-projection errors as in bundle adjustment (Eq. (7.2)):

$$J_{LBA}(X) \doteq \sum_{i=1}^{N_h} \|h_i(X_i, Z_i)\|^2_{\Sigma_i}, \tag{7.6}$$

where $h_i \in \{g_{2v}, g_{3v}\}$ represents a single two- or three-view constraint that is a function of several camera poses $X_i \subset X$ and image observations Z_i in the appropriate views, and N_h is the overall number of such constraints. One can observe that the cost function J_{LBA} does not contain any 3D points as variables.

Note that an exact marginalization of 3D points $p(X|Z) = \int_L p(X,L|Z)\,dL$ is another alternative, however, as discussed in the sequel, it densifies the matrices and therefore will not necessarily result in computational gain. Algebraic elimination of 3D points following the approach discussed above avoids some of this densification by discarding a certain amount of information. As a result, the inference is performed only over the camera poses and is typically significantly faster than conventional bundle adjustment at the cost of a certain degradation in accuracy. The analysis of this tradeoff is further discussed in Section 7.4.

7.3.3 Incremental Smoothing

The second component in iLBA is the recently-developed incremental smoothing [19, 18], an efficient nonlinear optimization technique that exploits sparsity and re-uses calculations when possible. Below we review the main concepts of this technique, and refer the reader to [19, 18] for further details.

[2] When adding a new camera into the optimization we initialize its pose using a three-view constraint g_{3v}, while keeping the poses of the other two cameras fixed.

7.3.3.1 Factor Graph Representation

The incremental smoothing technique uses the factor graph graphical model [23] to represent a given factorization of the joint probability distribution function (pdf). Formally, a factor graph is a bipartite graph $G = (\mathcal{X}, \mathcal{F}, \mathcal{E})$ with $\mathcal{X}, \mathcal{F}$ being variable and factor nodes and $\mathcal{E}$ consisting of edges that connect between these two variable groups. Each ith probabilistic term $p(.)$ in the factorization of the joint pdf is represented by a *factor* node $f \in \mathcal{F}$ that is connected by edges $e \in \mathcal{E}$ to *variable* nodes $\mathcal{X}_i \subset \mathcal{X}$ that are involved in $p(.)$.

In case of bundle adjustment, the factorization of the joint pdf $p(X, L|Z)$ is given by Eq.(7.1), and defining the projection factor for some view x, landmark l and image observation z as

$$f_{proj}(x,l) \doteq \exp\left(-\frac{1}{2}\|z - proj(x,l)\|_{\Sigma}^2\right),$$

the factor graph formulation can be trivially written as

$$p(X, L|Z) \propto \prod_i \prod_{j \in \mathcal{M}_i} f_{proj}(x_i, l_j).$$

In order to represent LBA in a factor graph, a suitable joint probability distribution function $p_{LBA}(X|Z)$ should be formulated first. Since the residual errors of three-view constraints (7.3)-(7.5) have been shown [13] to be of Gaussian distribution, the LBA pdf and the two- and three-view factors can be defined as

$$p_{LBA}(X|Z) \propto \prod_{i=1}^{N_h} f_{2v/3v}(X_i, Z_i), \tag{7.7}$$

and

$$f_{2v}(x_k, x_l) \doteq \exp\left(-\frac{1}{2}\|g_{2v}(x_k, x_l, z_k, z_l)\|_{\Sigma_{2v}}^2\right) \tag{7.8}$$

$$f_{3v}(x_k, x_l, x_m) \doteq \exp\left(-\frac{1}{2}\|g_{3v}(x_k, x_l, x_m, z_k, z_l, z_m)\|_{\Sigma_{3v}}^2\right), \tag{7.9}$$

where the covariance matrices Σ_{2v} and Σ_{3v} are given in [13].

Figure 7.5b illustrate factor graphs that represent $p(X, L|Z)$ and $p_{LBA}(X|Z)$ in a simple case of 4 camera poses observing 2 different 3D points. Each method uses different factors as discussed above.

7.3.3.2 Incremental Inference

Computing a MAP estimate of a given joint pdf typically involves a nonlinear optimization, where in each iteration a linearized system of the form $\Delta^* = \text{argmin}_{\Delta}(A\Delta - b)$ is solved. To that end, the *large sparse* Jacobian matrix A is factorized, e.g. using QR factorization, and an equivalent system $R\Delta - d$, with $d \doteq Q^T b$ is obtained.

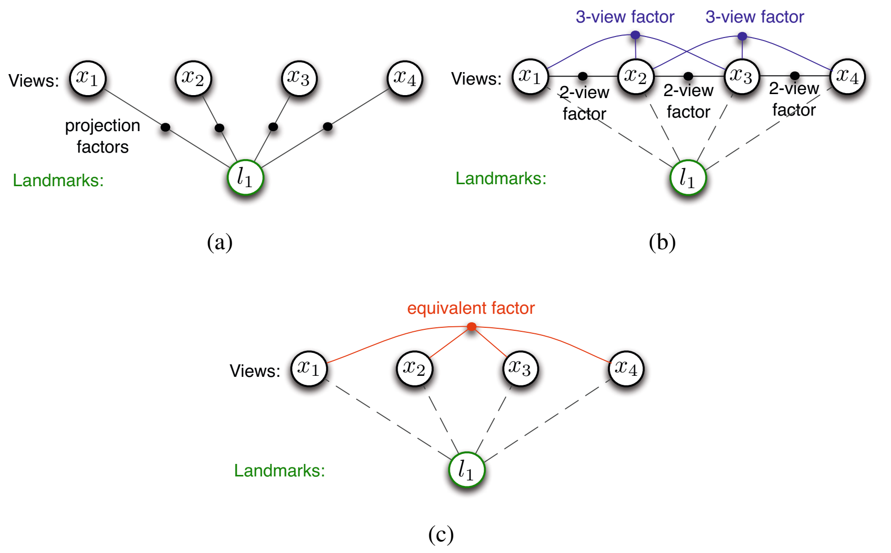


Fig. 7.5 Factor graph formulation for (a) BA and (b) LBA. (c) Factor graph after marginalizing out the landmark l_1.

Since R is upper triangular, this system can be easily solved, in a process known as back-substitution.

Instead of calculating a factorization of the Jacobian A from scratch, the matrix R from the previous factorization can be updated with new information. Incremental smoothing performs this operation very efficiently using graphical models [19, 18]: The factor graph is eliminated into a Bayes net using a calculated elimination order, and can be also converted into Bayes tree. Both graphical models represent the sparse factorized matrix R (which is the square root information matrix). Updating a factorization involves identifying what parts in the Bayes net (tree) are affected and re-eliminating only these variables. Additionally, tracking the validity of linearization point of each variable allows to perform selective re-linearization, instead of always re-linearizing all variables, while still recovering the MAP estimate up to a tolerance [19, 18].

7.4 Probabilistic Analysis of Light Bundle Adjustment

This section analyzes how well the LBA distribution $p_{LBA}(X|Z)$ represents the true density $p(X|Z)$. An exact calculation of the latter would marginalize the landmarks from the joint $p(X,L|Z)$

$$p(X|Z) = \int_L p(X,L|Z)\,dL.$$

While in practice, LBA represents a similar probability density over cameras as BA, there are two root effects that cause the LBA distribution to be an approximation of the true density: First, *LBA discards some mutual information in large camera cliques*, by considering only the mutual information between camera pairs and triplets introduced by them observing the same landmark. Bundle adjustment, on the other hand, induces mutual information between *all* cameras observing the same landmark. Second, *LBA duplicates some information for image measurements used in multiple factors*, double-counting measurements that appear in multiple two- or three-view factors.

As an example of both of these effects, consider observing a landmark l by four views x_1, x_2, x_3 and x_4, as illustrated in Figure 7.5. The joint pdf is given by

$$p\left(X_4, l | Z_4\right) \propto \prod_{i=1}^{4} f_{proj}\left(x_i, l\right), \tag{7.10}$$

where X_4 and Z_4 denote the four camera poses and the four image observations, respectively. On the other hand, the LBA pdf is

$$p_{LBA}\left(X_4 | Z_4\right) \propto f_{2v}\left(x_1, x_2\right) f_{2v}\left(x_2, x_3\right) f_{3v}\left(x_1, x_2, x_3\right) \\ f_{2v}\left(x_3, x_4\right) f_{3v}\left(x_2, x_3, x_4\right) \tag{7.11}$$

which corresponds to the set of two- and three-view factors, as shown in Figure 7.5.

The first effect, discarding of mutual information, can be seen when comparing the LBA pdf with the pdf resulting from eliminating the landmarks from the BA pdf,

$$p\left(X_4 | Z_4\right) = \int_{X_4} p\left(X, L | Z\right) dX_4 = p\left(x_1, x_2, x_3, x_4 | z_1, z_2, z_3, z_4\right) \tag{7.12}$$

The result in the case of BA is a single clique over all cameras. In general, there is no way to exactly factor such a dense clique in a way that reduces complexity. The multiple factors of LBA over pairs and triplets (Eq. (7.11)) reduce complexity instead by discarding some "links" that would otherwise be introduced between cameras.

The second effect, duplication of some image measurement information, can be seen in the sharing of cameras between LBA factors in Eq. (7.11). Any two factors sharing a camera in common both use the information from the shared camera, effectively duplicating it. For example, $f_{2v}\left(x_1, x_2\right)$ and $f_{2v}\left(x_2, x_3\right)$ both use the information from the measurements in camera 2.

This duplication of information happens since the two- and three-view factors were assumed to have independent noise models, represented by the covariance matrix Σ_i for the ith factor, and therefore could be separately written in the LBA pdf (7.7). This assumption is violated for factors that share measurements, as in the example above. One approach to avoid double counting is therefore to augment such factors, while accounting for the fact that the same measurement is involved by appropriate cross-covariance terms in the (augmented) covariance matrix.

For example, for any two factors representing constraints g_a and g_b, we can define $f_{aug} \doteq \exp\left(-\frac{1}{2}\left\|g_{aug}\right\|^2_{\Sigma_{aug}}\right)$, with $g_{aug} \doteq \begin{bmatrix} g_a & g_b \end{bmatrix}^T$ and the augmented covariance matrix Σ_{aug} :

$$\Sigma_{aug} \doteq \begin{bmatrix} \Sigma_a & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_b \end{bmatrix} \tag{7.13}$$

where the cross-covariance terms Σ_{ab} are non-zero when the constraints share measurements. However, since multiple factors are combined into a single multi-dimensional factor that involves all the variables in these individual factors, the factor graph becomes denser. Therefore, such an approach is expected to have considerable impact on computational complexity.

As we show in the next section, despite the above two aspects, the actual LBA distribution is very similar to the true distribution $p(X|Z)$. It is worth mentioning that the presented probabilistic analysis is valid for other existing structure-less BA methods [34, 31, 10] as well.

7.4.1 Datasets for Evaluation and Implementation

We use two datasets to evaluate how well the iLBA distribution $p_{LBA}(X|Z)$ represents the true density $p(X|Z)$. In the first dataset (*Cubicle*) the camera observes a cubicle desk in an open space environment from different viewpoints and distances (see Figure 7.2). In the second dataset, *Outdoor*, the camera follows a trajectory encircling a courtyard and building and performing loop closures as shown in Figures 7.1 and 7.6. Figure 7.7 shows typical images from these two datasets, while

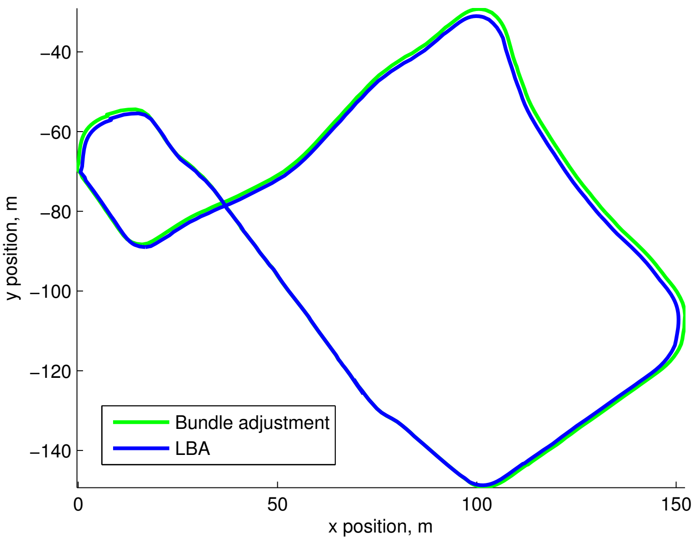


Fig. 7.6 Estimated trajectory in *Outdoor* dataset. LBA and conventional BA produce very similar results.

Table 7.1 Dataset details and performance of iLBA and BA: Re-projection errors and computational cost using incremental smoothing in all methods

Dataset	N, M, #Obsrv	Avg. reproj. error [pix]		Overall time [s]	
		iLBA	iBA	iLBA	iBA
Cubicle	148, 31910, 164358	0.552 pix	0.533 pix	581	6024
Outdoor	308, 74070, 316696	0.418 pix	0.405 pix	3163	26414

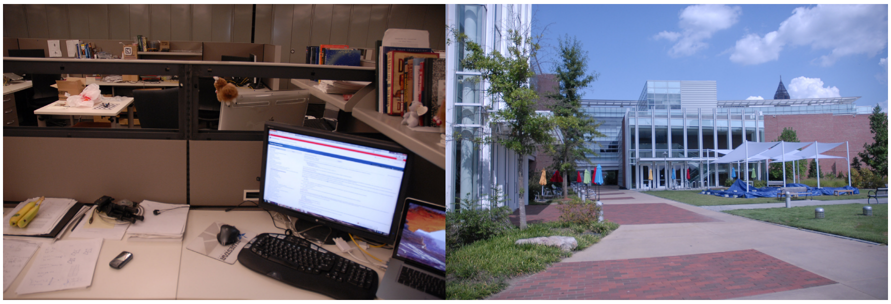

(a) (b)

Fig. 7.7 Typical images in the *Cubicle (a)* and *Outdoor* (b) datasets

Table 7.1 provides further details regarding the number of views (N) and 3D points (M), as well as the number of total observations in the two datasets.

All methods were implemented using the GTSAM factor graph optimization library[3] [3, 18]. Incremental smoothing was used in all cases, denoted by the prefix i (i.e. iLBA and iBA). Image correspondences, as well as the camera calibration matrices, were obtained by first running Bundler[4] [32] on each dataset. Additional implementation details can be found in [13].

7.4.2 Evaluation

In this section we compare the distributions of iLBA and incremental BA using two real imagery datasets. We first discuss how this comparison is made, present MAP estimate and computational cost of each method in Section 7.4.2.2 and then focus on estimated uncertainties by the two approaches in Section 7.4.2.3.

[3] `https://borg.cc.gatech.edu/.`

[4] `http://phototour.cs.washington.edu/bundler.`

7.4.2.1 Method for Comparing the PDFs of LBA and BA

Because computing the true marginal over cameras for BA $p(X|Z)$ is not tractable in closed form, we use an alternate method to compare the pdfs of LBA and BA. This method evaluates how well LBA and BA agree in both the absolute uncertainty of each camera in a global frame, and the relative uncertainty between all pairs of cameras.

In order to compare uncertainties, we first assume that $p_{LBA}(X|Z)$ and $p(X|Z)$ both are well-approximated as multivariate Gaussian distributions about their MAP estimates

$$p_{LBA}(X|Z) = N(\mu_{LBA}, \Sigma_{LBA})$$
$$p(X|Z) = N(\mu, \Sigma).$$

In order to compare relative uncertainty between cameras, we compare conditional densities $p(x_i|x_j, Z)$ between all pairs of cameras. This calculation quantifies how well LBA agrees with BA in relative uncertainty, while avoiding calculating the full covariance matrix on all cameras, which quickly becomes intractable for large numbers of cameras. The conditionals are obtained by integrating out all variables other than x_i and x_j,

$$p(x_i|x_j, Z) = \int\limits_{X\setminus\{x_i, x_j\}, L} p(X, L|Z) / p(x_j|Z).$$

In practice, we do this analytically by approximating the joint as a Gaussian around its MAP estimate, and applying sparse factorization,

$$p(X, L|Z) = p\left(X\setminus\{x_i, x_j\}, L|x_i, x_j, Z\right) p(x_i|x_j, Z) p(x_j|Z) \tag{7.14}$$

from which the desired conditional $p(x_i|x_j, Z)$ can be read off.

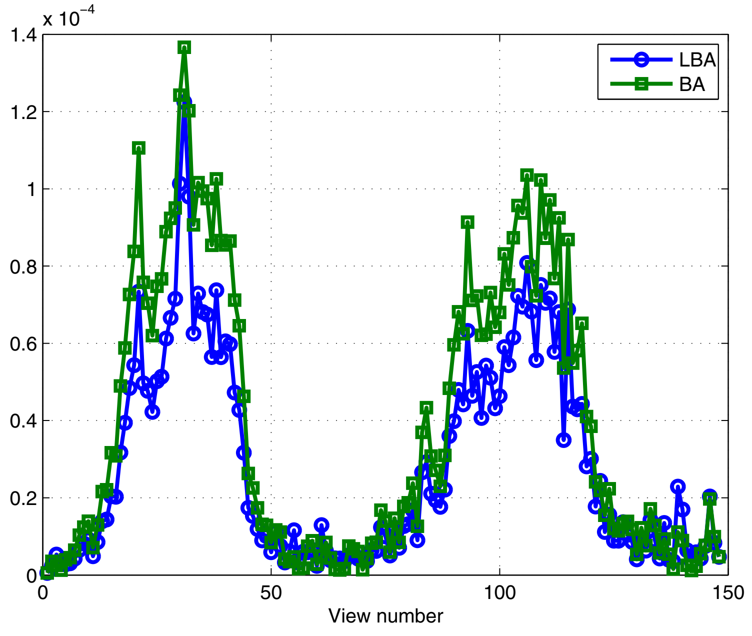
7.4.2.2 MAP Estimate and Computational Cost

Before discussing probabilistic aspects, we show performance results, in terms of accuracy of the MAP estimate and computational complexity. As seen in Table 7.1 and Figure 7.6, while iLBA yields a similar, but a bit degraded accuracy, the computational cost of iLBA is *8-10 times faster* than incremental BA.

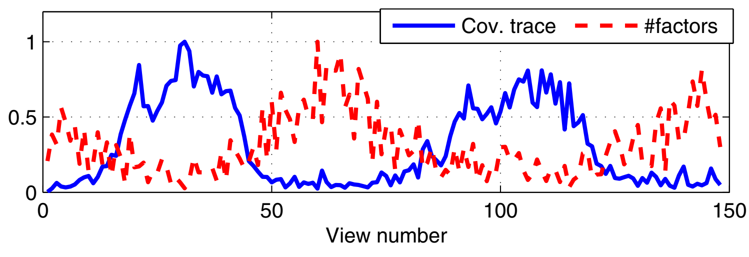
7.4.2.3 Estimated Camera Pose Uncertainty

We compare the probability density of the cameras estimated by iLBA to that of incremental BA by comparing their discrepancy both in the marginal uncertainty of each camera, and in relative uncertainty between each camera pair, as described in Section 7.4.2.1. We provide details as to how this comparison was made in the Appendix.

A comparison of the absolute uncertainty for the *Cubicle* dataset is given in Figure 7.8 and Figures 7.9a-7.9b. Figure 7.8a compares, for each camera pose i, between the covariance trace of Σ^i_{LBA} and Σ^i_{BA}. As seen, the initial uncertainty is very small and it increases as the camera moves around the cubicle deck and drops to low values when the camera captures previously-observed areas thereby providing loop-closure measurements. Figure 7.8b describes the interaction between the uncertainty of each view and the number of factors that involve this view. As expected, it can be seen that the covariance is higher when less factors are involved and vice versa.



(a)



(b)

Fig. 7.8 *Cubicle* dataset: (a) Covariance trace of each camera pose. (b) Trace of covariance and number of factors in LBA formulation, both are normalized to 1.

Overall, the absolute uncertainties in LBA and BA are very similar. This can be also observed in Figures 7.9a-7.9b that show a histogram of the discrepancy (7.19) both for position and orientation terms. Typical position discrepancies are near -10^{-4} meters. The discrepancies for relative uncertainties are given in Figures 7.9c-7.9d for position and orientation terms.

Figure 7.10 shows the discrepancy histograms for the *Outdoor* dataset. The absolute and relative discrepancies between LBA and BA are small, e.g. less than 5 centimeters in the absolute position for a trajectory that spans an area of 120×150 meters (cf. Figure 7.6), and on the order of 10^{-4} radians for the absolute rotation uncertainty.

Our conclusion from this evaluation is that the uncertainties estimated using iLBA represent well the uncertainties of incremental BA and therefore can be used instead of the later, e.g. in the context of establishing data association.

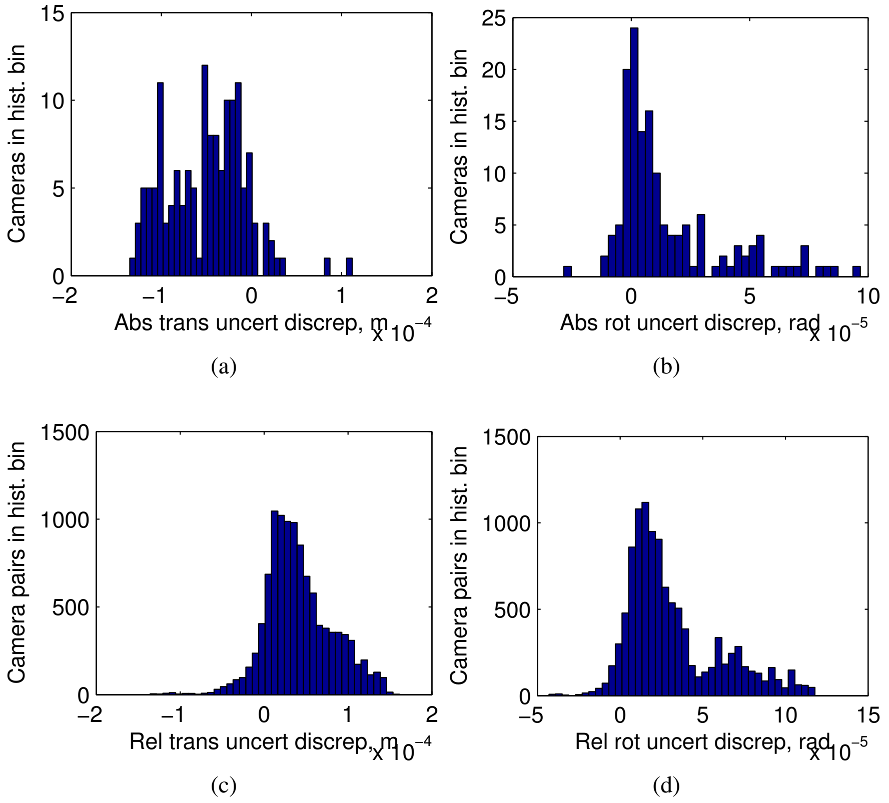


Fig. 7.9 Discrepancy histograms for the *Cubicle* dataset: Absolute position (a) and orientation (b); Relative position (c) and orientation (d) between every camera pair in the sequence

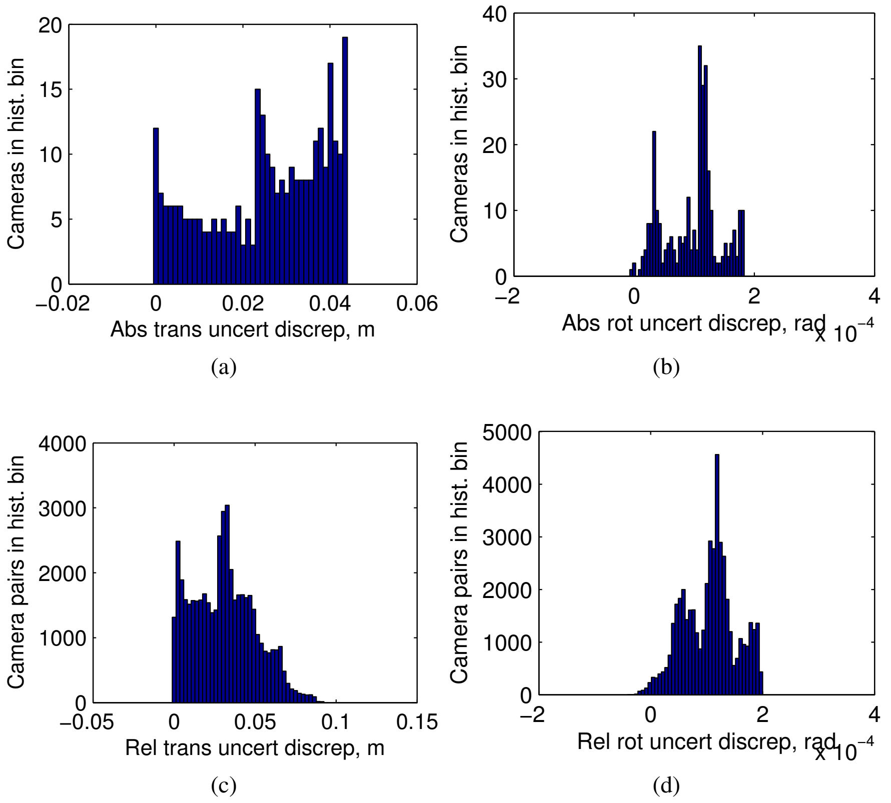


Fig. 7.10 Discrepancy histograms for the *Outdoor* dataset: Absolute position (a) and orientation (b); Relative position (c) and orientation (d) between every camera pair in the sequence

7.5 Application iLBA to Robotic Navigation

While incremental light bundle adjustment has been discussed thus far in the context of structure from motion problems, it is particularly attractive also to robotic navigation where different sensors are often available.

In this section we extend iLBA to robotic navigation and consider the challenging configuration of a robot equipped only with high-rate inertial navigation sensors (IMU) and a monocular camera. We show that this information fusion problem can be solved using incremental smoothing and adapt a recently-developed technique [26] for summarizing consecutive IMU measurements to obtain high-rate performance. We present proof-of-concept results using a synthetic aerial scenario.

Slightly abusing the previous notation, we redefine x to be the navigation state, comprising robot pose (position and orientation) and velocity. The IMU calibration parameters are denoted by b; although in this section no specific parametrization is assumed, we will refer to b as IMU bias.

7.5.1 Formulation

As standard in navigation literature [5], we use the probabilistic motion model

$$p\left(x_{k+1}|x_k,b_k,z_k^{IMU}\right) \propto \exp\left(-\frac{1}{2}\left\|x_{k+1}-h^{IMU}\left(x_k,b_k,z_k^{IMU}\right)\right\|_{\Sigma_{IMU}}^2\right) \doteq f^{IMU}\left(x_{k+1},x_k,b_k\right) \tag{7.15}$$

to represent the distribution over the state x_{k+1} given previous state x_k, an IMU measurement z_k^{IMU} and IMU calibration, that we will refer to as bias, b_k. The function h^{IMU} represents the nonlinear discrete inertial navigation equations [5]. The time evolution of IMU bias is modeled using some dynamics function h^b and is expressed probabilistically as

$$p\left(b_{k+1}|b_k\right) \propto \exp\left(-\frac{1}{2}\left\|b_{k+1}-h^b\left(b_k\right)\right\|_{\Sigma_b}^2\right) \doteq f^{bias}\left(b_{k+1},b_k\right). \tag{7.16}$$

Fusing information from IMU and camera sensors using LBA framework then involves calculating the MAP estimate of the following joint probability distribution function:

$$p\left(X_k,B_k|Z_k\right) \propto \prod_{s=0}^{k-1}\left[f^{IMU}\left(x_{s+1},x_s,b_s\right)f^{bias}\left(b_{s+1},b_s\right)\prod_{i=1}^{n_s}f_{2v/3v}\left(X_{s_i}\right)\right], \tag{7.17}$$

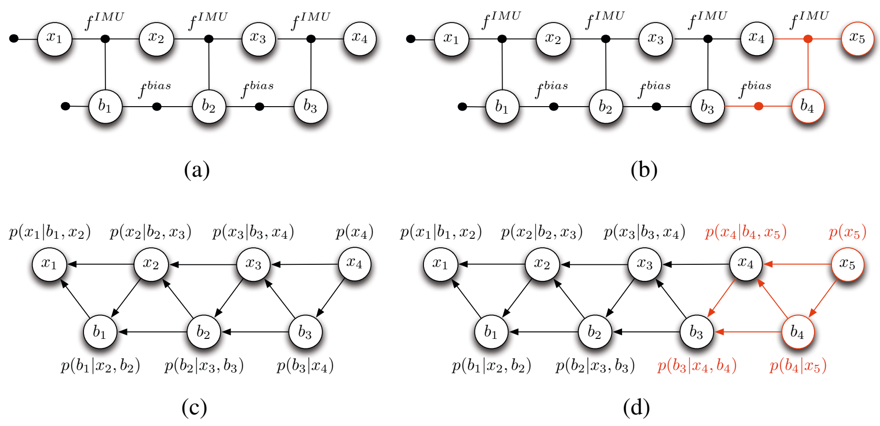
where we used the IMU and bias factors f^{IMU} and f^{bias} defined in Eqs. (7.15)-(7.16), with Σ_{IMU} and Σ_b representing the corresponding process covariance matrices, and the overall set of IMU biases denoted by $B_k \doteq \left[b_1^T \cdots b_k^T\right]^T$; in practice, since these tend to only have slow dynamics, it makes sense to describe this process in some lower rate [15, 16]. In Eq (7.17), n_{s+1} is the number of two- and three-view factors that are added between each current state x_{s+1} and past states. Thus, if $x_a \in X_{s_i}$ then $a \leq s+1$.

While the MAP estimate $X_k^*,B_k^* = \arg\max_{X_k,B_k} p\left(X_k,B_k|Z_k\right)$ can be calculated using incremental smoothing, high-rate performance becomes infeasible: Number of variables in the optimization rapidly increases as a new navigation state is introduced at IMU rate (for each new IMU factor). Moreover, number of variables that need to be re-calculated rapidly increases when new two- and three-view factors are added to the graph.

To get a better understanding of this aspect, it is beneficial to first consider only IMU observations. Adding new IMU and bias factors involve only re-eliminating the two past navigation and bias states *regardless* to the graph size. This is illustrated in Figure 7.11 for two consecutive time instances t_4 and t_5. The figure shows both factor graphs and bayes nets with the latter representing the square root information matrix R. The nodes that were modified from the previous Bayes net are shown in Figure 7.11d in red.

Considering now also a camera sensor, adding two- and three-view factors (or projection factors for bundle adjustment) would require re-eliminating many more variables. While the exact number depends on variable elimination ordering, typically at least the variables in between the variables involved in the new factors will have to be re-eliminated. For example, adding a single two-view factor (Figure 7.12a) most probably will involve re-eliminating the majority of the variables $x_1 - x_5$ and $b_1 - b_4$. Adding other multi-view factors that involve additional navigation states will require re-eliminating many more variables, and thus high-rate performance is only possible for a limited time.

In the next section we discuss a solution to this problem.



f^{IMU}
f^{bias}
x_1 x_2 x_3 x_4 x_5
b_1 b_2 b_3 b_4
(a)
(b)
$p(x_1|b_1,x_2)$ $p(x_2|b_2,x_3)$ $p(x_3|b_3,x_4)$ $p(x_4)$
$p(b_1|x_2,b_2)$ $p(b_2|x_3,b_3)$ $p(b_3|x_4)$
(c)
$p(x_4|b_4,x_5)$ $p(x_5)$
$p(b_3|x_4,b_4)$ $p(b_4|x_5)$
(d)

Fig. 7.11 (a)-(b) Factor graphs and (c)-(d) Bayes nets for the pure-IMU case in two consecutive time instances. Adding IMU and bias factors involves re-eliminating only 2 past nodes. Modified parts are marked in red.

7.5.2 Equivalent IMU Factor

In this section we adopt a recently-developed technique [26] for IMU measurements pre-integration that allows to reduce the number of variables and factors in the optimization, resulting in significantly improved computational complexity.

The idea is to integrate consecutive IMU measurements between two time instances t_i and t_j into a single component, denoted by $\Delta x_{i\to j}$, comprising the accumulated change in position, velocity and orientation, represented respectively by $\Delta p_{i\to j}, \Delta v_{i\to j}$ and the rotation matrix R^i_j:

$$\Delta x_{i\to j} \doteq \left\{\Delta p_{i\to j}, \Delta v_{i\to j}, R^i_j\right\} = \eta\left(Z^{IMU}_{i\to j}, b_i\right),$$

where $Z^{IMU}_{i\to j}$ is the set of IMU measurements between the time instances t_i and t_j, that are corrected using the bias b_i, and η is a known non-linear function that describes

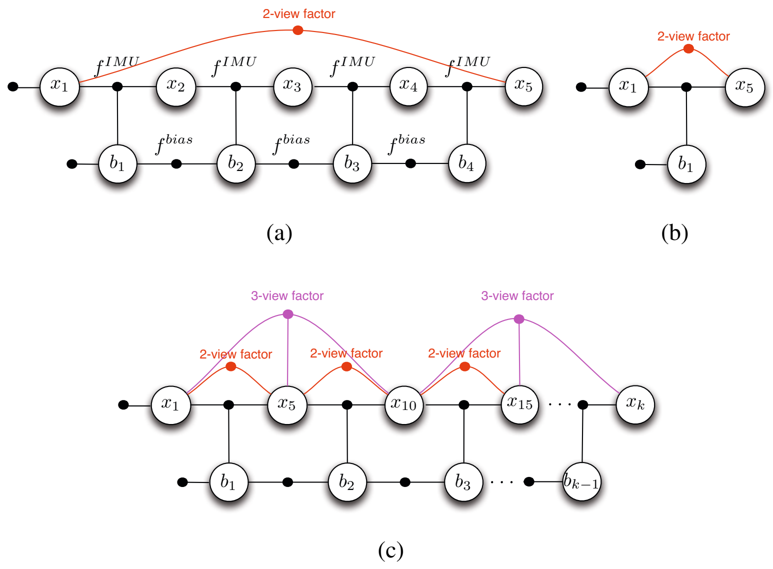

2-view factor
3-view factor
f^{IMU}
f^{bias}
x_1 x_2 x_3 x_4 x_5 x_{10} x_{15} x_k
b_1 b_2 b_3 b_4 b_{k-1}
(a)
(b)
(c)

Fig. 7.12 (a) Factor graph with a single two-view factor with IMU and bias factors; (b) The corresponding factor graph using an equivalent IMU factor; (c) Factor graph with many two- and three-view factors as well as equivalent IMU and bias factors

the IMU measurements pre-integration process. One can now use $\Delta x_{i \to j}$ to predict x_j based on the current estimate of x_i. Let h^{Equiv} represent this predicting function.

We can then define an *equivalent* IMU factor [16] f^{Equiv} as

$$f^{Equiv}(x_j, x_i, b_i) \doteq \exp\left(-\frac{1}{2}\left\|x_j - h^{Equiv}(x_i, b_i, \Delta x_{i \to j})\right\|_{\Sigma}^{2}\right), \tag{7.18}$$

which involves only the variables x_j, x_i and b_i for any reasonable[5] two time instances t_i and t_j. Figure 7.12b illustrates the conceptual difference between the conventional and equivalent IMU factors.

The approach for calculating $\Delta x_{i \to j}$ involves pre-integrating the IMU measurements while expressing them in the navigation frame. However, this will require re-calculating $\Delta x_{i \to j}$ from scratch each time the rotation estimate changes, i.e. each re-linearization of x_i. To resolve this, as proposed in [26], the different components in $\Delta x_{i \to j}$ are expressed in the body frame of the first time instant (i.e. t_i), which allows re-linearizing the factor (7.18) without recalculating $\Delta x_{i \to j}$. The reader is referred to [26] for further details.

[5] The original derivation in [26] neglects Earth curvature and Earth rotation, however it can be extended to the more general case which assumes the gravity vector and the rotation rate of the navigation frame with respect to an inertial frame are constant. The time instances t_i, t_j should be chosen such that these assumptions are satisfied.

The equivalent IMU factor allows to significantly reduce the number of variables and factors in the optimization, and enables high-rate performance while using efficient optimization techniques. This is illustrated in Figure 7.12c, that shows a factor graph with two- and three-view factors and the equivalent IMU factor bridging between navigation states (variables) from different time instances. Note that a conventional IMU factor would require adding consecutive navigation states to the graph.

Furthermore, since in typical navigation systems a navigation solution x_t is required in real time, i.e. each time an IMU measurement is obtained, one can predict x_t using the accumulated component $\Delta x_{i\to j}$ and the current estimates $\hat{x}_i, \hat{b}_i$ of x_i and b_i, in a parallel process and *without* incorporating x_t into the optimization, i.e. $h^{Equiv}\left(\hat{x}_i, \hat{b}_i, \Delta x_{i\to j}\right)$.

7.5.3 Evaluation in a Simulated Aerial Scenario

In this section we present an evaluation of the described extension of LBA to robotic navigation in a realistic aerial scenario covering an area of about 2×1.5 km as shown in Figure 7.13a. We also compare LBA to the BA approach, both using the equivalent IMU factors and incremental smoothing. A statistical study of the approach using a smaller scenario is reported in a conference version of this paper [12].

In the simulated scenario, the aerial vehicle gradually explores different areas and occasionally re-visits previously observed locations thereby providing loop closure measurements. The flight is at a constant height of 200 meter above mean ground level, with a 40 m/s velocity. The aerial vehicle travels a total distance of about 13 km in 700 seconds. A medium-grade IMU and a single downward-facing camera, operating at 100 Hz and 0.5 Hz, were used.

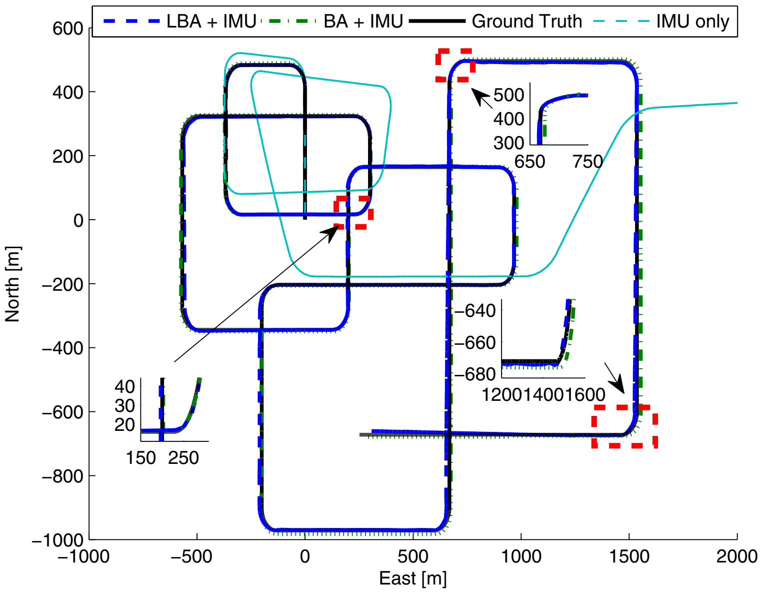
The 100 Hz ideal IMU measurements were corrupted with a constant bias and a zero-mean Gaussian noise in each axis. Bias terms were drawn from a zero-mean Gaussian distribution with a standard deviation of $\sigma = 10$ mg for the accelerometers and $\sigma = 10$ deg/hr for the gyroscopes. The noise terms were drawn from a zero-mean Gaussian distribution with $\sigma = 100\ \mu g/\sqrt{Hz}$ and $\sigma = 0.001\ deg/\sqrt{hr}$ for the accelerometers and gyroscopes. Visual observations of unknown 3D points were corrupted by a zero-mean Gaussian noise with $\sigma = 0.5$ pixels.

The estimated trajectory by LBA and BA, compared to ground truth and to pure IMU integration, is shown in Figure 7.13a, with position estimation errors given in Figure 7.13b. One can observe the fast drift of IMU-based dead reckoning, while both LBA and BA yield estimates close to ground truth with similar levels of accuracy. Note that only IMU and monocular cameras are used, *without* GPS or any additional sensors, producing position estimates with a typical estimation error of $5-10$ meters, with a highest estimation error of 20 meters.

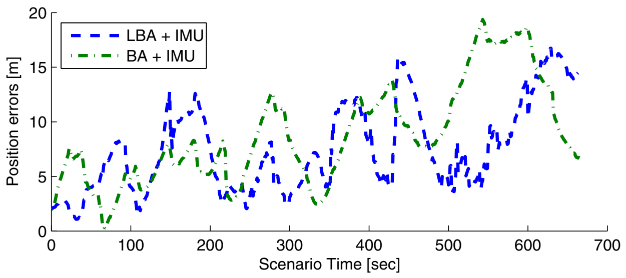
While a similar estimation accuracy was obtained both by LBA and BA, processing time is different. The latter depends on the number of feature observations per frame γ, which affects the number of observed landmarks. We therefore

Table 7.2 Average processing time per camera frame

#Features per frame	#Landmarks	#Observations	Ave. Time [sec]		
			BA	**LBA**	Ratio
200	9.5k	66k	0.59	**0.27**	2.19
500	23k	165k	1.95	**0.57**	3.42



(a)

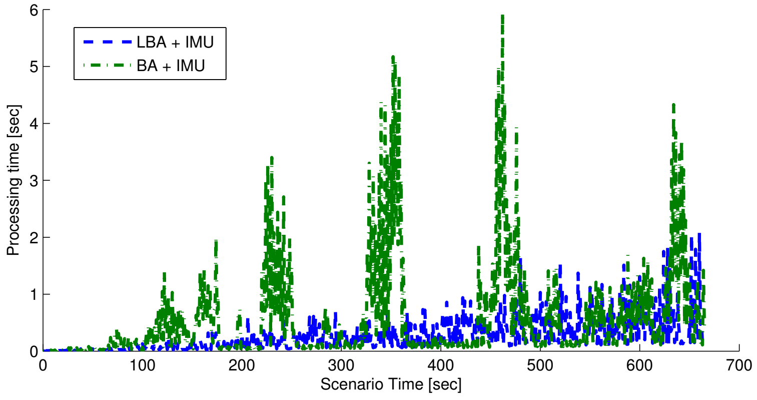


(b)

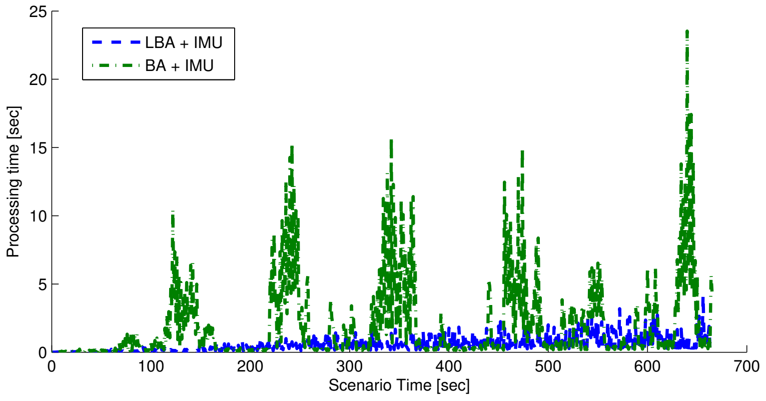
Fig. 7.13 (a) Top view of estimated trajectory. Inertial navigation quickly drifts while both LBA and BA result in bounded navigation errors over time; (b) Position estimation errors (norm).

discuss processing time for two different values of feature observations per frame, $\gamma = \{200, 500\}$, while performing exactly the same trajectory. In the former case, number of landmarks is 9.5k with total number of image observations of about 66k, while in the latter case, number of landmarks and total number of image observations are 23k and 165k, respectively.

Processing time for these two cases is shown in Figures 7.14a-7.14b and summarized in Table 7.2. As seen, while BA exhibits lower processing time now and then, in particular when far from loop closure frames, the overall processing time is much smaller in the LBA case. One can clearly observe the spikes in BA, that are the result of massive variable re-elimination and re-linearization triggered by loop



(a)



(b)

Fig. 7.14 (a) and (b) Processing timing comparison between the proposed method and bundle adjustment for 200 and 500 features per frame. Both methods use incremental smoothing and equivalent IMU factors.

closures and proceeds for many frames afterwards. Overall, the average processing time per frame in the shown scenario for $\gamma = 200$ features is 0.27 and 0.59 seconds for LBA and BA, respectively. Increasing number feature observations per frame to $\gamma = 500$, leads to further difference in average processing time, as shown in Figure 7.14b: 0.57 and 1.95 seconds for LBA and BA. Thus, LBA is about 2 times faster, on average, than BA for $\gamma = 200$ features, and almost 5 times faster for $\gamma = 500$ features.

7.6 Conclusions and Future Work

This paper focused on incremental light bundle adjustment (iLBA) [13], a structure-less bundle adjustment approach that reduces computational complexity by algebraically eliminating the 3D points using multiple view geometry constraints and utilizing an efficient incremental optimization - incremental smoothing. Our first contribution is a theoretical probabilistic analysis of iLBA, where we identified the root effects that may cause the underlying probability distribution of iLBA to be somewhat different from the probability distribution over camera poses that is calculated from full bundle adjustment. Using two real-imagery datasets we demonstrated that, in practice, these two probability distributions are very close in terms of the maximum a posteriori estimate and the estimated uncertainty.

The second contribution of this paper is an extension of iLBA to robotic navigation, where besides a camera sensor, additional sensors operating at different rates typically exist. In particular, we considered the problem of fusing information between high-rate inertial navigation sensors (IMU) and vision observations. Following the iLBA concept, our formulation avoids explicit estimation of camera-observed 3D points, and utilizes a recently developed technique for IMU pre-integration to significantly reduce the number of variables in the optimization. We demonstrated, based on a realistic synthetic aerial scenario, that iLBA for robotic navigation produces comparable state estimation accuracy to bundle adjustment formulation, where 3D points are explicitly inferred, while reducing average computational time by a factor of 2-3.5.

Future research will focus on developing approaches for optimally choosing past camera frames when adding new multi-view geometry constraints and on extensive experimental evaluation of the described application of iLBA to robotic navigation.

Appendix

This appendix presents further details regarding the metric used to compare estimated camera pose uncertainty in Section 7.4.2.

To compare two covariance matrices Σ_1 and Σ_2, we define a discrepancy measure of the square roots of the traces of each covariance matrix,

$$discrepancy(\Sigma_1, \Sigma_2) \overset{\Delta}{=} c\left(\sqrt{\mathrm{tr}(\Sigma_1)} - \sqrt{\mathrm{tr}(\Sigma_2)}\right), \tag{7.19}$$

where c is a scale factor that converts the unit-less 3D reconstructions into meters, which we determined by physically measuring the dataset collection area, or superimposing the trajectory onto a satellite image. We compute this separately for the blocks of the covariance matrices corresponding to rotation and translation. The units of the discrepancy are radians for rotation ($c = 1$) and meters for translation, with c properly determined to correct the reconstruction scale.

For example, to compare the Gaussian-approximated conditional density of LBA $p_{LBA}\left(x_i|x_j, Z\right)$ with covariance $\Sigma_{LBA}^{i|j}$ with that of BA $p\left(x_i|x_j, Z\right)$ with covariance $\Sigma_{BA}^{i|j}$, we compute $discrepancy\left(\Sigma_{LBA}^{i|j}, \Sigma_{BA}^{i|j}\right)$. Similarly for marginals $p_{LBA}\left(x_i|Z\right)$ and $p_{BA}\left(x_i|Z\right)$, we compute $discrepancy\left(\Sigma_{LBA}^{i}, \Sigma_{BA}^{i}\right)$. A positive discrepancy value means that the uncertainty estimate of LBA is conservative, whereas a negative discrepancy value means that the uncertainty estimate of LBA is overconfident.

References

1. Avidan, S., Shashua, A.: Threading fundamental matrices. IEEE Trans. Pattern Anal. Machine Intell. 23(1), 73–77 (2001)
2. Crandall, D., Owens, A., Snavely, N., Huttenlocher, D.: Discrete-continuous optimization for large-scale structure from motion. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 3001–3008 (2011)
3. Dellaert, F., Kaess, M.: Square Root SAM: Simultaneous localization and mapping via square root information smoothing. Intl. J. of Robotics Research 25(12), 1181–1203 (2006)
4. Eustice, R., Singh, H., Leonard, J.: Exactly sparse delayed-state filters for view-based SLAM. IEEE Trans. Robotics 22(6), 1100–1114 (2006)
5. Farrell, J.: Aided Navigation: GPS with High Rate Sensors. McGraw-Hill (2008)
6. Strasdat, H., Montiel, J.M.M., Davison, A.J.: Scale drift-aware large scale monocular SLAM. In: Robotics: Science and Systems (RSS), Zaragoza, Spain (2010)
7. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2000)
8. Ila, V., Porta, J.M., Andrade-Cetto, J.: Information-based compact Pose SLAM. IEEE Trans. Robotics 26(1) (2010), `http://dx.doi.org/10.1109/TRO.2009.2034435` (in press)
9. Indelman, V.: Navigation performance enhancement using online mosaicking. Ph.D. thesis, Technion - Israel Institute of Technology (2011)
10. Indelman, V.: Bundle adjustment without iterative structure estimation and its application to navigation. In: IEEE/ION Position Location and Navigation System (PLANS) Conference (2012)
11. Indelman, V., Gurfil, P., Rivlin, E., Rotstein, H.: Real-time vision-aided localization and navigation based on three-view geometry. IEEE Trans. Aerosp. Electron. Syst. 48(3), 2239–2259 (2012)
12. Indelman, V., Melim, A., Dellaert, F.: Incremental light bundle adjustment for robotics navigation. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS (2013)
13. Indelman, V., Roberts, R., Beall, C., Dellaert, F.: Incremental light bundle adjustment. In: British Machine Vision Conf., BMVC (2012)
14. Indelman, V., Roberts, R., Dellaert, F.: Probabilistic analysis of incremental light bundle adjustment. In: IEEE Workshop on Robot Vision, WoRV (2013)

15. Indelman, V., Wiliams, S., Kaess, M., Dellaert, F.: Factor graph based incremental smoothing in inertial navigation systems. In: Intl. Conf. on Information Fusion, FUSION (2012)
16. Indelman, V., Wiliams, S., Kaess, M., Dellaert, F.: Information fusion in navigation systems via factor graph based incremental smoothing. Robotics and Autonomous Systems 61(8), 721–738 (2013)
17. Kaess, M., Ila, V., Roberts, R., Dellaert, F.: The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In: Hsu, D., Isler, V., Latombe, J.-C., Lin, M.C. (eds.) Algorithmic Foundations of Robotics IX. STAR, vol. 68, pp. 157–173. Springer, Heidelberg (2010)
18. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., Dellaert, F.: iSAM2: Incremental smoothing and mapping using the Bayes tree. Intl. J. of Robotics Research 31, 217–236 (2012)
19. Kaess, M., Ranganathan, A., Dellaert, F.: iSAM: Incremental smoothing and mapping. IEEE Trans. Robotics 24(6), 1365–1378 (2008)
20. Kaess, M., Wiliams, S., Indelman, V., Roberts, R., Leonard, J., Dellaert, F.: Concurrent filtering and smoothing. In: Intl. Conf. on Information Fusion, FUSION (2012)
21. Konolige, K.: Sparse sparse bundle adjustment. In: British Machine Vision Conf., BMVC (2010)
22. Konolige, K., Agrawal, M.: FrameSLAM: from bundle adjustment to realtime visual mapping. IEEE Trans. Robotics 24(5), 1066–1077 (2008)
23. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Trans. Inform. Theory 47(2) (2001)
24. Lourakis, M.A., Argyros, A.: SBA: A Software Package for Generic Sparse Bundle Adjustment. ACM Trans. Math. Software 36(1), 1–30 (2009), doi:http://doi.acm.org/10.1145/1486525.1486527
25. Lu, F., Milios, E.: Globally consistent range scan alignment for environment mapping. Autonomous Robots, 333–349 (1997)
26. Lupton, T., Sukkarieh, S.: Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. IEEE Trans. Robotics 28(1), 61–76 (2012)
27. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.: An Invitation to 3-D Vision. Springer (2004)
28. Mourikis, A., Roumeliotis, S.: A multi-state constraint Kalman filter for vision-aided inertial navigation. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 3565–3572 (2007)
29. Mourikis, A., Roumeliotis, S.: A dual-layer estimator architecture for long-term localization. In: Proc. of the Workshop on Visual Localization for Mobile Platforms at CVPR, Anchorage, Alaska (2008)
30. Ni, K., Steedly, D., Dellaert, F.: Out-of-core bundle adjustment for large-scale 3D reconstruction. In: Intl. Conf. on Computer Vision (ICCV), Rio de Janeiro (2007)
31. Rodríguez, A.L., de Teruel, P.E.L., Ruiz, A.: Reduced epipolar cost for accelerated incremental sfm. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 3097–3104 (2011)
32. Snavely, N., Seitz, S., Szeliski, R.: Photo tourism: Exploring photo collections in 3D. In: SIGGRAPH, pp. 835–846 (2006)
33. Snavely, N., Seitz, S.M., Szeliski, R.: Skeletal graphs for efficient structure from motion. In: IEEE Conf. on Computer Vision and Pattern Recognition, CVPR (2008)
34. Steffen, R., Frahm, J.-M., Förstner, W.: Relative bundle adjustment based on trifocal constraints. In: Kutulakos, K.N. (ed.) ECCV 2010 Workshops, Part II. LNCS, vol. 6554, pp. 282–295. Springer, Heidelberg (2012)
35. Vidal, R., Ma, Y., Soatto, S., Sastry, S.: Two-View Multibody Structure from Motion. Intl. J. of Computer Vision 68(1), 7–25 (2006)

Chapter 8
Online Learning of Vision-Based Robot Control during Autonomous Operation

Kristoffer Öfjäll and Michael Felsberg

Abstract. Online learning of vision-based robot control requires appropriate activation strategies during operation. In this chapter we present such a learning approach with applications to two areas of vision-based robot control. In the first setting, self-evaluation is possible for the learning system and the system autonomously switches to learning mode for producing the necessary training data by exploration. The other application is in a setting where external information is required for determining the correctness of an action. Therefore, an operator provides training data when required, leading to an automatic mode switch to online learning from demonstration. In experiments for the first setting, the system is able to autonomously learn the inverse kinematics of a robotic arm. We propose improvements producing more informative training data compared to random exploration. This reduces training time and limits learning to regions where the learnt mapping is used. The learnt region is extended autonomously on demand. In experiments for the second setting, we present an autonomous driving system learning a mapping from visual input to control signals, which is trained by manually steering the robot. After the initial training period, the system seamlessly continues autonomously. Manual control can be taken back at any time for providing additional training.

8.1 Introduction

Perception-action learning for robotic systems is a challenging problem. Preferably, both learning and autonomous operation as well as switching of operational mode should be online such that new training data can be provided if and when previous training data is insufficient. Switching to training mode, obtaining training data and finally switching back to autonomous mode should be possible to perform without taking the system out of operation.

Kristoffer Öfjäll · Michael Felsberg
Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden
e-mail: {kristoffer.ofjall,michael.felsberg}@liu.se

Y. Sun et al. (eds.), *New Development in Robot Vision*,
Cognitive Systems Monographs 23, DOI: 10.1007/978-3-662-43859-6_8

There are two primary classes of problems, one class where sufficient information for self-evaluation is present in the learning scenario and one complementary class where it is not, that is, additional information is needed to determine the correctness of an action. In the first class, the system is able to observe and assess the success of its actions itself, without external intervention; in the second class, an external supervisor (typically human) provide the system with feedback on its performance. Online learning in the two classes are similar with respect to mode switching and request of new training data, the difference lies in what entity initiates the mode switch and from where training data is provided.

One typical problem from the first class is *reaching*, where the system has received a stimuli somewhere in the visual field and tries to position a manipulator at the position of the stimuli. An example from the second class is *autonomous driving*, where typical driving behaviors such as staying on the roads are not inherently available in the learning situation but have to be acquired from external sources. In the application presented here, not even the notion of *a road* is present in the system initially. The two learning concepts are related to two major modes of learning in biological systems: random exploration and mirroring. Problems in the first class are typically solved by random exploration such as small children randomly moving their arms, first for identification of the visual stimuli generated by arm movements and later for learning of the *inverse kinematics*: how desired positions in the visual field can be reached.

This chapter presents a perception-action learning system for robotics where the training data is either generated by modulated motor babbling (random exploration) or by demonstration. The proposed learning system is incremental and real-time capable during both learning and autonomous operation.

In experiments, the proposed method shows its capabilities of learning from exploration (inverse kinematics) and learning from demonstration (autonomous driving). For learning of the inverse kinematics of a robotic arm, training data may be self-generated by random exploration. However, as is shown in the experiments, an exploration scheme biased towards minimizing the pose error provides for faster convergence to previously unknown poses and produces training data more relevant for the problem at hand. For the autonomous driving experiment, the system is not given a-priori knowledge on the type of visual event it should expect or the driving rules it should follow, but rather learns them from a human. After manually piloting the robot 1.5 laps around a track, the controls are released and the robot successfully continues around the track. The system demonstrates ability to generalize to changes of the track. Correction of unwanted behavior is demonstrated in the experiments by reclaiming manual control during a fraction of a lap. Switching between autonomous driving and demonstration is performed without stopping the robot. A video demonstrating the applications in this chapter is available.[1]

[1] http://users.isy.liu.se/cvl/ofjall/onlinePerceptionActionLearningSmaller.mp4

8.2 Previous Work

The general Perception-Action learning field is far too wide for a comprehensive presentation in the current scope. Here we concentrate on the two application examples presented in the introduction and provide some references to learning approaches previously applied to problems regarding inverse kinematics and autonomous navigation.

With online learning methods, the robot model can adapt to changes while the system is in operation. This has shown to be useful, especially in cases where the controlled system tends to change properties within short time frames such as in [10]. Online learning enables immediate switching between learning mode and autonomous mode. All experience gathered during a training session is directly available for use when switching back to autonomous mode. In contrast, for offline learning systems, the system has to be taken out of service while processing training data.

First, the inverse kinematics problem is briefly presented (Sect. 8.2.1), which is used as an application for demonstration of learning from exploration (Sect. 8.2.2). Learning from demonstration is applied on a visual autonomous navigation (Sect. 8.2.3) task. Finally, two components used in the proposed method are presented, locally weighted projection regression (Sect. 8.2.4) and Levenberg-Marquardt minimization (Sect. 8.2.5).

8.2.1 Inverse Kinematics

The inverse kinematics of a robotic arm is a mapping from the desired *pose* (position and orientation) of the end effector to a set of angles for the joints of the robot (the *joint configuration*) which makes the end effector attain the desired pose. The classical approach uses handcrafted geometric models of the robotic arms [32]. The accuracy depends on the complexity of the model and any changes of the robot behavior due to wear are disregarded. Control models based on learning systems have the possibility to adapt to individual differences between robots of the same manufacturer and type as well as being able to learn different robot setups. By retraining the system, also changes in the robot can be handled. Different variations of neural networks have been popular approaches to this problem [21]. The results have been improved by using modular neural networks [17, 18], where several different neural networks are trained and the output from the locally most suitable network is used.

The neural network approaches mostly use offline training and require training points in the order of millions. Depending on the physical layout of the robotic arm, the mapping can be decomposed into one mapping from orientation of the end effector to a subset of the joints and another mapping from position to the remaining joints [1]. Any changes of the robot in these examples still require offline retraining.

8.2.2 Active Learning and Exploration

Active learning generally means that the learning system can affect the generation or selection of training data, the term is not clearly defined but [29] provides a survey. Active learning is mostly interesting in the case where training data is generated by the system itself, not by an external teacher. Here, this applies to the inverse kinematics learning application where the system has the possibility to actively control the robotic arm. This can be used to generate training data that maximizes information gain given the current state of the learnt model.

One fundamental requirement of online self-learning is the availability to the system of an estimate of its error immediately after an action. In the case of vision-based robotic manipulation, the actual pose achieved by the system can be inferred from the visual sensor data and compared with the desired pose to estimate the error in the last action of the system. This visual feedback is fundamental in visual servoing and in combination with online kinematics learning this enables continuous improvement of the kinematics model.

In cases where the current learnt model does not provide any clue regarding reducing the pose error, an exploration strategy is required for training data generation. Motor babbling is a popular approach where random motions around the current configuration are carried out. In some cases this would provide information on how to proceed towards the desired pose, but as described in [27], *falling down might not tell us much about the forces needed in walking*. That is, motor babbling may generate training data that does not affect the predictions of the learnt model.

Approaches which address this issue to some extent are proposed by [25, 3], where goals are used to direct the babbling and exploration of the available motion space. The work by [4] use a statistical foundation for presenting a more general and theoretical framework for this type of active learning.

When the space of possible movements is large, exploring the whole space can be very time consuming and depending on the application, only a small subset of these movements may be used. The Shifting Setpoint Algorithm [26] is a method where models are built along tubes in the motion space between desired points. As the name suggests, a setpoint is shifted towards the desired point and motor babbling is carried out around it. When the model is good enough locally (as determined by the algorithm), the setpoint is shifted again. This solves a similar problem but robot movement iterations and time are unnecessarily spent generating accurate models in between desired poses where a coarser model would suffice.

8.2.3 Visual Autonomous Navigation

One of the earliest successful learning approaches to vision based autonomous navigation systems was ALVINN (Autonomous Land Vehicle In a Neural Network) [19]. Like the work presented in this chapter, ALVINN learns how to control a vehicle by observing a human driving. The learning is based on a single hidden layer back-propagation network. A more recent work from LeCun et al. focuses on learning vision based obstacle avoidance for off-road robots [11]. The learning algorithm in [11] requires very large collections of data and is based on a large 6-layer convolutional network. The system learns features that predict traversable areas. The convolutional network approach has also been used to create a track-following robot [28]. All these examples use offline learning, where training data has to be processed for hours or days before the system can operate autonomously.

Here an autonomous driving challenge is used as an example application where correct behavior is not apparent from the problem itself. The problem of autonomous navigation has been approached in a number of different ways in the literature, which can be roughly divided between the classical control–based approaches [6, 35, 34] and the learning–based approaches [19, 20, 23] (we refer to [12] for an in–depth review). The approach used in this chapter belongs to the second category, where perception is visual and the correct action is demonstrated by manual control of the robot, *learning from demonstration*.

The visual perception consists of a generic, holistic representation of the whole visual field, using so-called Visual Gist [16]. Visual Gist provides a compact and generic approach to image description, and has been used successfully for scene identification [24], image search [7], indoor vs. outdoor detection [30, 31], road type detection [9] and driver action prediction [22].

8.2.4 Locally Weighted Projection Regression

Locally Weighted Projection Regression [36], LWPR, has successfully been applied to learning problems related to robotics applications [2, 26, 8, 15]. The general idea is to use the output from several local linear models, distributed in the input space, weighted together to form the output.

The output y_{dk} for each local model k for dimension d consists of r_k linear regressors

$$y_{dk} = \beta^0_{dk} + \sum_{i=1}^{r_k} \beta_{dki} \mathbf{u}^T_{dki} (\mathbf{x}_{dki} - \mathbf{x}^0_{dk}) \tag{8.1}$$

along different directions $\mathbf{u}_{dki}$ in the input space. Each projection direction and corresponding regression parameter β_{dki} and bias β^0_{dk} are adjusted online to fit the training examples. Variations in the input explained by each regression i is removed from the input $\mathbf{x}$ generating the input to the next regressor $\mathbf{x}_{dk(i+1)}$.

The total prediction $\hat{y}_d$ in one output dimension d

$$\hat{y}_d = \frac{\sum_{k=1}^K w_{dk} y_{dk}}{\sum_{k=1}^K w_{dk}} \tag{8.2}$$

depends on the distance from the center $\mathbf{c}_{dk}$ of each of the local models. Normally a Gaussian kernel is used, generating the weights

$$w_{dk} = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_{dk})^T \mathbf{D}_{dk} (\mathbf{x} - \mathbf{c}_{dk})\right) \tag{8.3}$$

where the metric $\mathbf{D}_{dk}$ is updated while the model centers $\mathbf{c}_{dk}$ remain constant. Even though the regression is performed in a low dimensional projected space, the local models still live in the full input space. An advantage of LWPR which is useful for self-learning, is the possibility to derive analytic Jacobians of the learnt model [13].

8.2.5 *Numerical Optimization*

For inverse kinematics, minimizing a suitable distance between the current and desired poses will generate a solution to the given problem. There are many different algorithms available from the field of optimization [33]. Especially numerical optimization methods geared towards reaching a minimum using as few iterations as possible will avoid unnecessary movements of the robot.

If the kinematics of the robotic arm is not explicitly known, a numerical method has to be used. This implies iterative or grid based methods, where the former is expected to require fewer movements of the arm. One popular numerical algorithm is the Levenberg-Marquardt method [14]. For fast convergence, the algorithm requires a good initial solution and the derivative of the distance function at the iteration points. One possibility is to use the current pose of the robot as initial solution and estimating the derivatives using finite differences. As will be shown (Sect. 8.3.1), there are more efficient (in terms of time to find a solution) ways of choosing the initial solution as well as obtaining derivatives.

In Gauss-Newton optimization, the update $\mathbf{q}_\Delta$ in each step is obtained by solving a linearized problem

$$\mathbf{J}^T\mathbf{J}\mathbf{q}_\Delta = -\mathbf{J}^T\mathbf{e} \tag{8.4}$$

where $\mathbf{J}$ is the Jacobian at the current point and $\mathbf{e}$ is the current residual vector. In LM, the joint space update $\mathbf{q}_\Delta$ in each step is obtained by solving

$$(\mathbf{J}^T\mathbf{J} + \lambda \operatorname{diag}(\mathbf{J}^T\mathbf{J}))\mathbf{q}_\Delta = \mathbf{J}^T(\mathbf{x}_d - \mathbf{x}(\mathbf{q})) \tag{8.5}$$

where $\mathbf{J}$ is the Jacobian at the current configuration $\mathbf{q}$, $\mathbf{x}(\mathbf{q})$ is the pose of the current configuration and $\mathbf{x}_d$ is the desired pose. The method is a weighted (by λ) combination of Gauss-Newton and weighted gradient descent.

8.3 Proposed Method

We provide an online learning system with applications to inverse kinematics (Sect. 8.3.1) and autonomous navigation (Sect. 8.3.2). These examples represent the self learning case and the case where external information is required. In the first case, the mode switch is initiated by a pose error larger than a fixed accuracy threshold and training data is provided by numerically minimizing the pose error. In the second case, a mode switch is initiated manually by activating manual control of the robot. While under manual control, the learning system is trained.

8.3.1 Learning Inverse Kinematics by Exploration

The inverse kinematics represent the system's knowledge of which control signals are required to reach any desired state. In the case of a robot reaching for an object, this means predicting joint angles for attaining a specified *desired pose* of the end effector.

If the system can obtain an estimate of the current pose of the end effector, it can also estimate performance (pose error) autonomously by comparing actual and desired poses—assuming a distance measure in pose space. This provides for self-learning by exploration.

We propose that by combining an online learning approach with a numerical optimization method, the desired pose is reached quicker on average. The current learnt inverse kinematics model can be used to provide an initial solution as well as estimated Jacobians as required by numerical optimization methods. In return, the iterations of the numerical method serves as excellent training data along the lines of active learning.

In this chapter, a combination of Locally weighted projection regression, LWPR, and Levenberg-Marquardt, LM, is evaluated. The proposed integration of both methods is compared with a naïve combination of the two algorithms. In the baseline implementation, only outputs from the two algorithms are used. For the integrated method, internal data is shared.

Initial Solution

In the LWPR algorithm, the weight of each local model (8.3), is compared to a threshold w_{cut} to determine which local models should be used when predicting the output of a given input. This can also be used to determine where reasonable predictions can be expected. We propose using this information for determining an initial solution. Given the weight of each local model (8.3), using the notation of (8.3), reasonable predictions can be expected within the set

$$X_p = \bigcap_d \left(\bigcup_k \{\mathbf{x} : (\mathbf{x} - \mathbf{c}_{dk})^T \mathbf{D}_{dk} (\mathbf{x} - \mathbf{c}_{dk}) \leq -\ln w_{cut}\} \right) \tag{8.6}$$

as at least one local model k in each output dimension d should provide a useful prediction. Given a desired pose $\mathbf{x}_d$ it is thus possible to find an initial pose $\mathbf{x}_t \in X_p$ such that no other pose $\mathbf{x} \in X_p$ is closer to the desired pose.

An approximation of the optimal initial pose $\mathbf{x}_t$ still within X_p can easily be found by starting in the desired pose $\mathbf{x}_d$ and using gradient ascent on

$$\min_d \left(\max_k \exp\left(-(\mathbf{x} - \mathbf{c}_{dk})^T \mathbf{D}_{dk} (\mathbf{x} - \mathbf{c}_{dk})\right) \right) \tag{8.7}$$

to improve the dimension with the worst response of the best fitting model. The ascent is continued until the value of the expression reaches w_{cut} where the learnt inverse kinematics model is expected to be accurate enough. The model can then be used to move the arm directly close to this pose. As the initial pose may be on the border of X_p (the set of poses where inverse kinematic predictions can be made) the accuracy of the inverse kinematics model can not be expected to be perfect.

If the robotic arm has previously moved close to the desired pose, the initial pose will be very close to the desired pose. In areas where the inverse kinematics model is accurate, the desired pose will be reached directly.

8.3.2 Learning Autonomous Driving from Demonstration

For the autonomous navigation application, a mobile robot is supposed to navigate along a track. However, the problem specification does not specify what kind of track or what kind of visual features that defines the track. This is to be learnt by the system along with appropriate steering actions for staying on the track. Since the track appearance or layout is not known beforehand, the learning system can not evaluate its own performance.

We propose an online learning system which is trained by demonstrating correct behavior (manually driving the robot) and which is supposed to be able to follow the track immediately and autonomously after the initial training sequence. Specifically, assuming consistency in track markings, we want to learn a mapping $f : \varPhi \rightarrow \varTheta_{\text{turn}}$ (where $\varPhi$ denotes the visual features extracted from images P and $\varTheta_{\text{turn}}$ the steering angles) that will keep the vehicle driving on the track indefinitely, for any track layout. Our aim is then to learn f from N example pairs $(\phi^i, \theta^i_{\text{turn}})^N_{i=1}$, sampled from a demonstration by a human driving around the track.

The chosen approach has the following characteristics: (i) the steering is estimated directly from the visual input, frame by frame and is therefore not affected by previous errors; (ii) the system does not have or build a model of the track, allowing for navigation on potentially infinite paths; (iii) the system's visual input encompasses the whole visual information, and it is the learning that specifies which aspects of the visual scene are relevant for steering control; (iv) it learns and runs real-time; and (v) any misbehavior or lack of training data can be corrected by additional manual control without stopping the system.

The system's visual perception consists of a generic, holistic representation of the whole visual field, using so-called visual Gist [16]. This encodes the whole visual field into one feature vector, using multiscale filtering to encode the visual scene. There exists different versions of the Gist features. In this work, we compute the image Gist by filtering a downscaled (128×128 pixels) version of the image P with Gabor filters $G_{\lambda,\theta}(P)$ tuned at different scales (λ_i) and orientations (θ_i), and averaging the filters' responses over Gaussian channels $C_{i,j}$ regularly spaced over the image. We use 4 scales, 8 orientations and 8×8 channels, resulting in a feature vector of dimension 2048.

In this case, LWPR is unable to handle an input space of dimension 2048 as every local model contains the center of the local model and a matrix representing the size of the local in the input space. The input space dimension is reduced to 256 by means of PCA where the principal components are obtained from the Gist feature vectors from manually controlling the robot one lap clockwise and one lap counter clockwise on the track in Fig. 8.7. Note that the same principal components calculated from the indoor track also is used for the outdoor forest road in Fig. 8.11.

8.4 Evaluation

The proposed systems, learning from exploration or from demonstration are evaluated. Evaluation is performed both in controlled environments and under realistic conditions.

8.4.1 Learning from Exploration

The explorative approach of obtaining training data is evaluated in a reaching scenario with a robotic manipulator. The evaluated system is sequentially presented with desired poses of the end effector and the task is to set the joint angles of the arm such that the desired pose is obtained. The evaluated system obtains measurements of the current pose of the end effector. For better accuracy in performance comparisons and as common in literature [5, 18, 13, 33], the presented methods are primarily evaluated on a simulated, planar robotic arm. The learning system has also been used to control a 7-DoF KUKA LWR, however it was not possible to obtain objective performance data from the latter experiment.

The simulated arm has three joints and the desired pose is the two dimensional position of the tip of the last link. The evaluated method controls the arm by specifying the three joint angles and the simulation returns the position of the final link with added Gaussian noise. The pose error $\mathbf{p}_e = \mathbf{p}_d - \mathbf{p}_c$ is used in the quadratic error function of the Levenberg-Marquardt algorithm (lsqnonlin implementation in Matlab). For LWPR, the implementation by Klanke, Vijayakumar and Schaal is used [36].

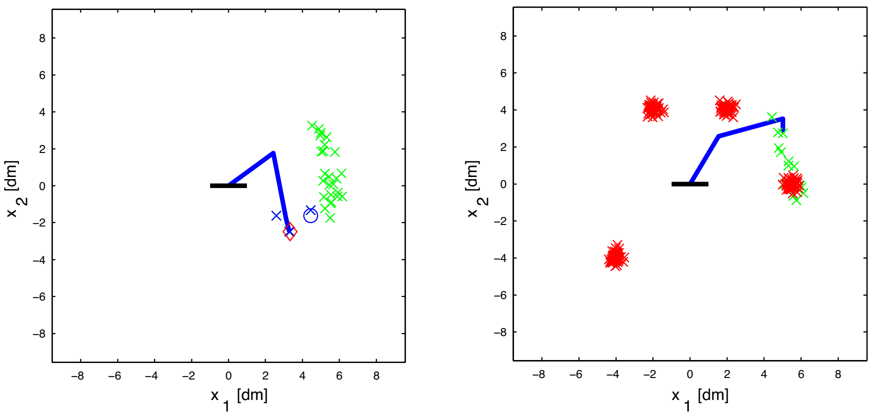


Fig. 8.1 (Left) The simulated robotic arm reaching for the red diamond marker. The blue circle indicates the initial position, the blue crosses are the iterations and the green crosses are the initial training data. (Right) The evaluation points marked with red crosses.

An initial training set with 15 points distributed in a small part of the reachable space of the robot is generated. Evaluation points are generated in four clusters with 50 points in each cluster. The training and evaluation points are shown in Fig. 8.1.

For evaluation, each LWPR model is trained using the training points. Each method is then used to move to the first evaluation point in the first cluster and the number of iterations required to reach the point is counted. This is repeated for the first point in the second cluster and so on until all evaluation points are reached.

When the first evaluation point in a cluster far from the training data is to be reached, the system is expected to require some iterations to move from the closest previously visited area to the desired position. When the arm is supposed to return to the second point within the same cluster, the number of iterations required is expected to be lower. After visiting a few points within the cluster, the system is expected to be able to move directly to the desired pose.

This behavior is expected both for systems using numerically estimated Jacobians (naïve) and for systems using Jacobians from the learnt model (integrated). For the Jacobians from the learnt model, the accuracy is expected to be low during the first runs but increasing as more points are visited within each cluster. For the numerically estimated Jacobians, the arm has to be moved for Jacobian estimation. Thus, if the system does not move the arm to the correct position in the first iteration, at least four additional movements of the arm are required.

In contrast to the learning approaches, using a numerical optimization method alone, always starting at the current arm pose, the number of iterations required to reach each point could be expected to stay constant as no information regarding the behavior of the robot is kept.

8.4.1.1 Results

The evaluation results are shown in Fig. 8.3. The graphs show the number of iterations required to reach each test point for the naïve combination and the proposed integrated method respectively. We expect real measurements to be noisy so independent zero mean Gaussian noise with standard deviation 0.03 was added to the position estimates. This corresponds to 0.2% of the diameter of the space reachable

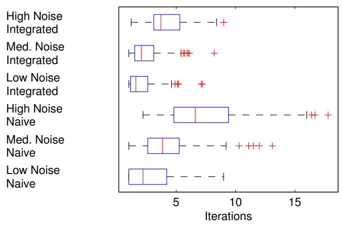


Fig. 8.2 Quartiles of the number of iterations required to reach each of the last 100 evaluation points for each evaluation session

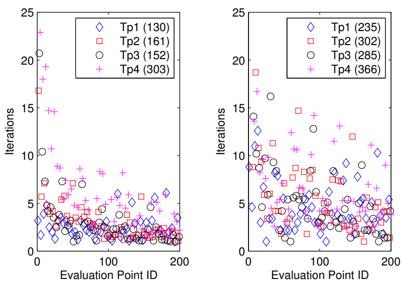


Fig. 8.3 Mean number of iterations required to reach each evaluation point. (Left) Integrated method; (Right) Naïve combination.

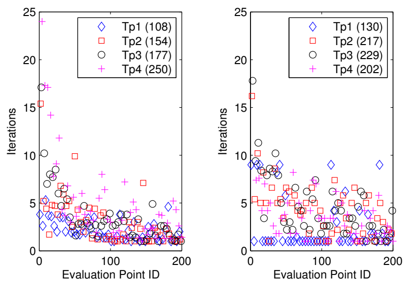


Fig. 8.4 Mean number of iterations required to reach each evaluation point with reduced measurement noise. (Left) Integrated method; (Right) Naïve combination.

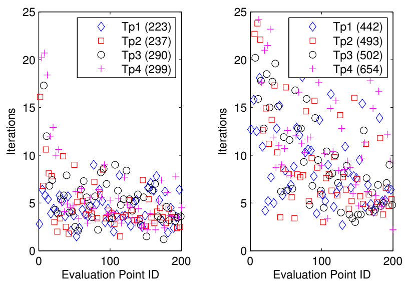


Fig. 8.5 Mean number of iterations required to reach each evaluation point with increased measurement noise. (Left) Integrated method; (Right) Naïve combination.

by the robot. The shown results are the average of ten runs. As expected, the number of iterations required to reach a point decays with increasing number of visited points in the same cluster.

If the Jacobians from the learnt model were perfect, the method using these should require about one fourth of the iterations required by the naïve method. Using Jacobians from the learnt model requires significantly fewer iterations than using numerically estimated Jacobians, but the theoretically possible reduction is not achieved. This is due to errors in the learnt model and to the implementation of the numerical solver not needing to estimate the Jacobian at every step.

Additionally, to assess the effect of the noise, simulations were run with reduced and increased noise variance. Reducing the standard deviation to one sixth of the original noise, the results in Fig. 8.4 are obtained. In Fig. 8.5 the standard deviation of the noise is doubled.

In the case of reduced noise, both methods perform better. The integrated method is better than using numerically estimated Jacobians. Increasing the noise, the numerical method performs significantly worse. Here, the averaging introduced by the learning method is a great advantage. In Fig. 8.2 evaluation results for the last 100 points in each session are presented more compactly. For each noise level, t-tests on this data indicate significant improvements ($p_{H_0} < 0.01$) where H_0: Equal mean error for naïve and integrated methods. One example of iterations for reaching a desired pose is shown in Fig. 8.1.

8.4.1.2 Learning Inverse Kinematics on the KUKA Robot

The proposed learning system is implemented on a 7 DoF KUKA LWR robotic arm. With this system, the 6 DoF pose of the cameras and tools can be controlled directly instead of controlling the joint angles of the robot. The online self learning algorithm enables increasing pose precision over time and self-recalibration after hardware changes or maintenance. The inverse kinematics solver shipped with the

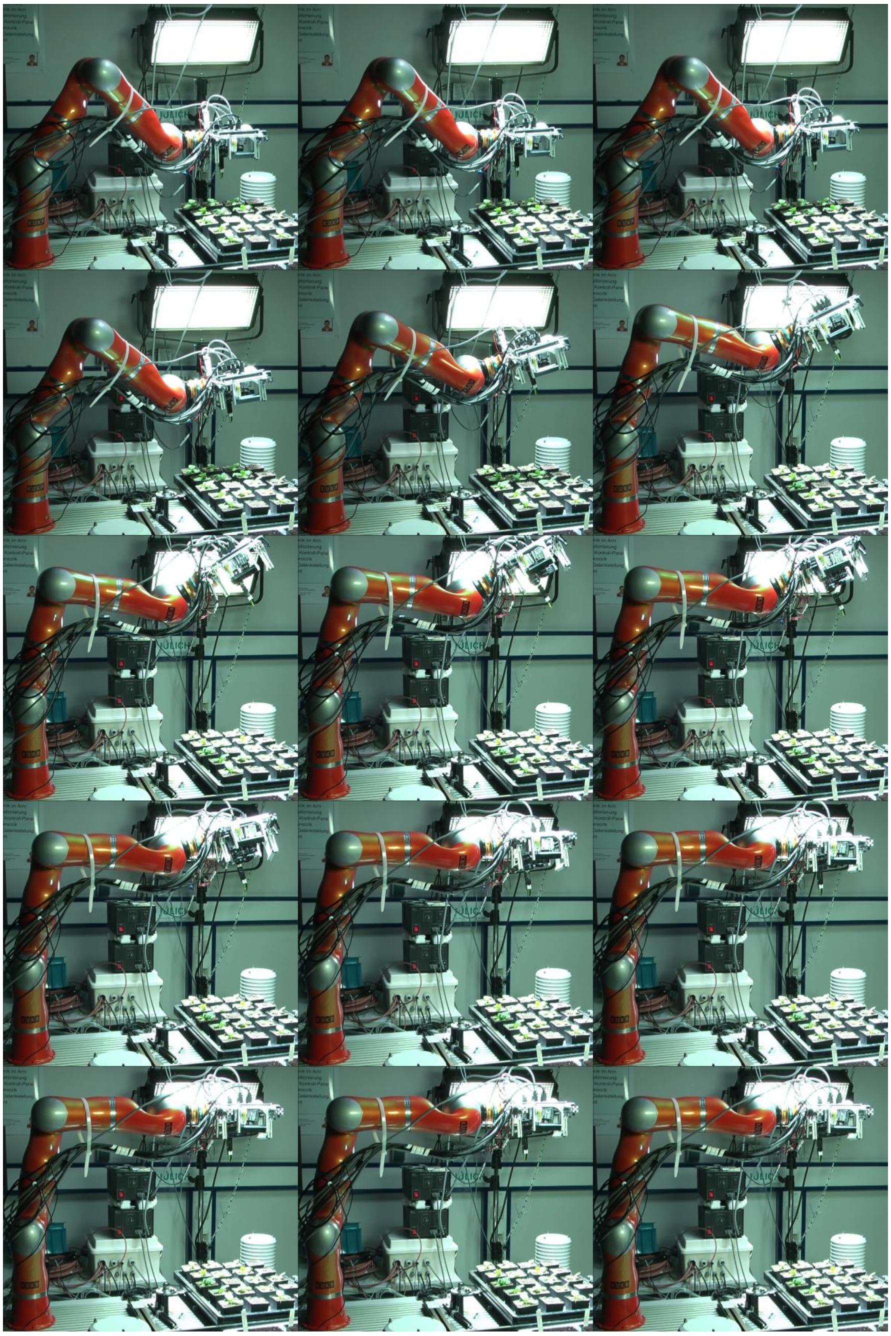


Fig. 8.6 Explorative learning of inverse kinematics for reaching on a 7 DoF robotic arm

arm is based on a geometric model and has issues with singularities. This poses a problem in the intended active vision applications where all desired poses are not known in advance and thus can not be checked for singularities before deploying the system.

In Fig. 8.6, a desired pose for a general view is selected. The desired pose is outside the region of known inverse kinematics solutions and the pose is reached by numerical minimization of the pose error. Intermediate poses provide training data. The figure shows time-equidistant frames from a video. Note that the desired pose is close to a singularity, as the desired pose is approached the rotational axes of the fifth and the seventh joints (counted from the base) are close to parallel. This configuration would not be well handled by the inverse kinematics solver shipped with the arm.

8.4.2 Learning from Demonstration

For evaluation of the learning from demonstration capabilities, a mobile robot is used based on a radio controlled car fitted with a laptop and a single grayscale camera. The laptop interfaces to the servo control on the car, so that controls generated by either the teacher (during training) or the software are actuated. The autonomous navigation module generates control signals from visual input from the camera. There is also a low-resolution color camera on the platform, however it is not used in the experiments.

This work uses teleoperation to gather the examples, i.e. the robot is operated by a teacher while recording both the control signals and the sensor readings. The teacher controls the car with a standard remote control transmitter. Training examples are processed online onboard the mobile robot. When the robot is able to generate control signal predictions from the visual input, the teacher is notified via an indication on the robot display and the remote controller can be released. The robot control can be manually overridden at any time for providing additional training examples. In the experiments, steering is predicted by the system while throttle is kept at a constant level.

The evaluation environment is a reconfigurable circuit made of tiles of carpet with sections of road markings. Different circuits can be made by placing tiles in the desired configuration, see fig. 8.9 for example circuit configurations. To evaluate the quality of a trajectory of the robot around the circuit, the robot is tracked. Very accurate tracking trajectories were obtained by tracking a red marker attached to the robot. Fig. 8.9e show results of this red marker tracking (the dashed lines). An homography is computed (by manually marking corresponding positions) from the position of the red marker to the ground plane, the result being the solid lines. Finally a second homography is computed to project both the images and the trajectories into a planar view, see fig. 8.9d for the projection of Fig. 8.9e. Since the red marker is not visible while behind the red beam, trajectories are linearly interpolated in that region.

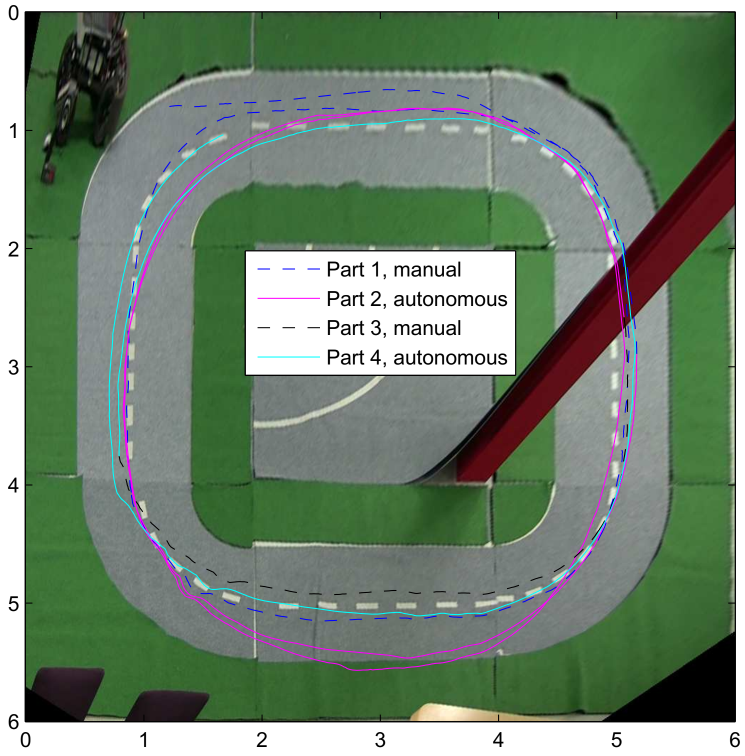


Fig. 8.7 Trace of first five clock-wise laps for a previously untrained system overlaid on an image of the track reprojected onto the ground coordinate system, with coordinates in meters

8.4.2.1 Autonomous Navigation Performance

Fig. 8.7 shows the trajectory for the first five laps of a previously untrained system. The robot is manually controlled through six corners (dashed blue line) whereafter the controls are released and the robot continues autonomously around the track (solid magenta line). As the trajectory in the lower part of the track was not considered accurate enough, manual control was used to override steering predictions through two corners (dashed black line). The new training data corrects the undesired behavior and during the following laps, the robot stays on the road (solid cyan line).

After the five first clockwise laps, the robot is manually turned around for counter clockwise laps. These are shown in Fig. 8.8. The robot is manually controlled through three corners whereafter the robot is able to generalize and make the forth turn autonomously. Steering control signals are plotted in Fig. 8.10. The switches to manual control for initial training, path correction and training for counter clockwise driving are clearly visible.

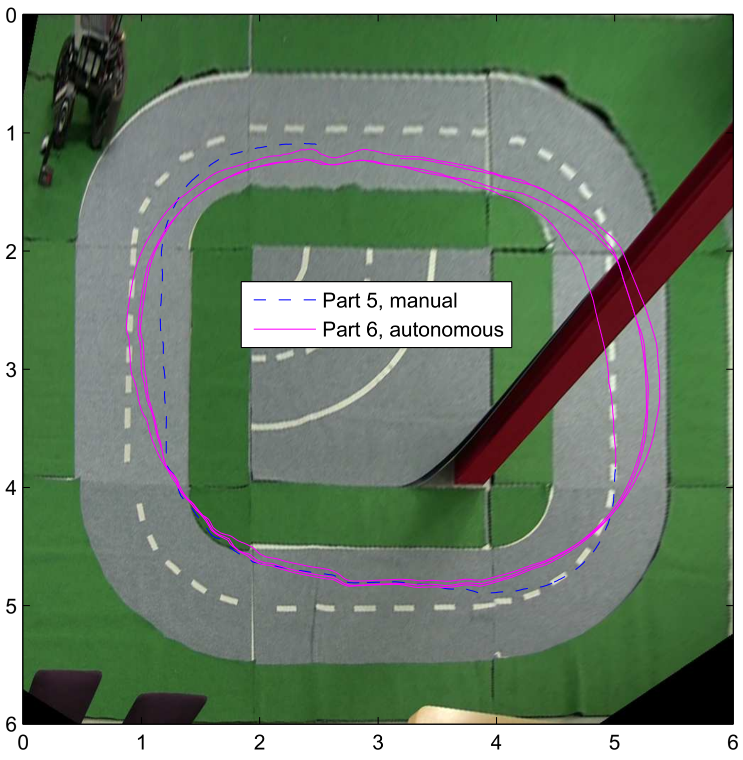
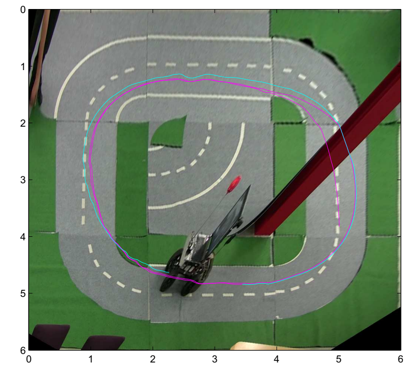
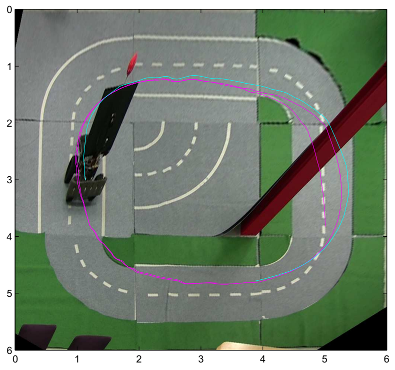


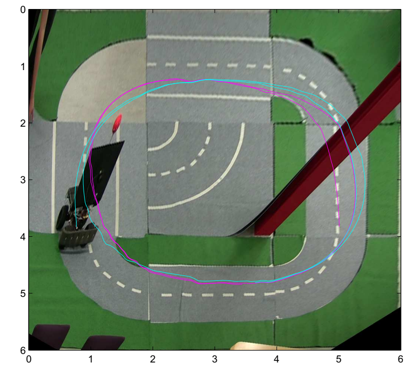
Fig. 8.8 Trace of the four laps following Fig. 8.7 where the robot runs counter clockwise

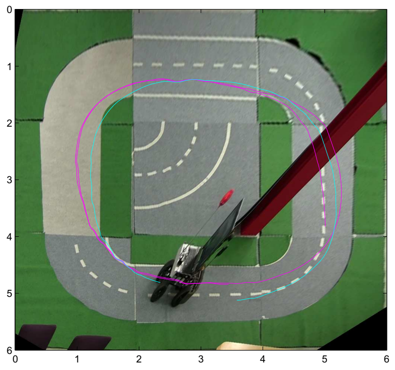
The robot continues to run autonomously while the track is changed, Fig 8.9. Road side markers are changed (Figs. 8.9a, 8.9b) and the robot generalizes, that is it successfully continues to navigate the track without additional training. Similarly, when restoring the initial road sides and changing the road itself, the robot is still able to follow the track with a tendency to cut the upper left corner compared to the autonomous trajectories from the unaltered track (Figs. 8.9c, 8.9d). This behavior of cutting corners was introduced in the training data for counter clockwise driving (dashed blue line in Fig. 8.8).

Additionally, the robot is evaluated in an outdoor environment, Fig. 8.11. In this environment, tracking is unavailable however the robot successfully follows the road. Note that no changes has been made to the robot for this evaluation, neither in software nor hardware. The robot is manually driven for approximately two minutes whereafter it drives autonomously.


(a) Part 7


(b) Part 8


(c) Part 9


(d) Part 10

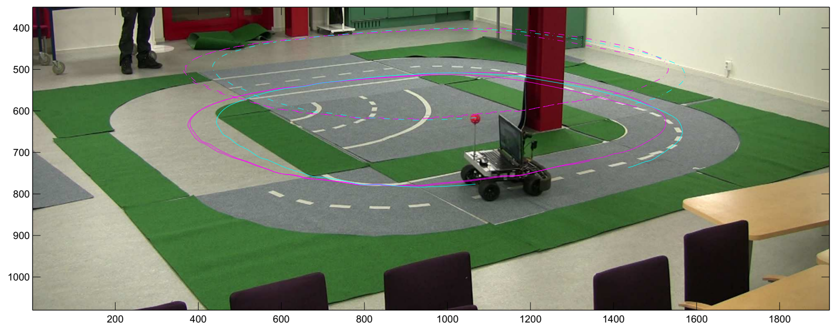

(e) Part 10, original view with red marker trajectory (dashed) and the corresponding ground projection (solid).

Fig. 8.9 Autonomous trajectories by the robot while reconfiguring the track (cyan), with last laps on unmodified track for comparison (magenta)

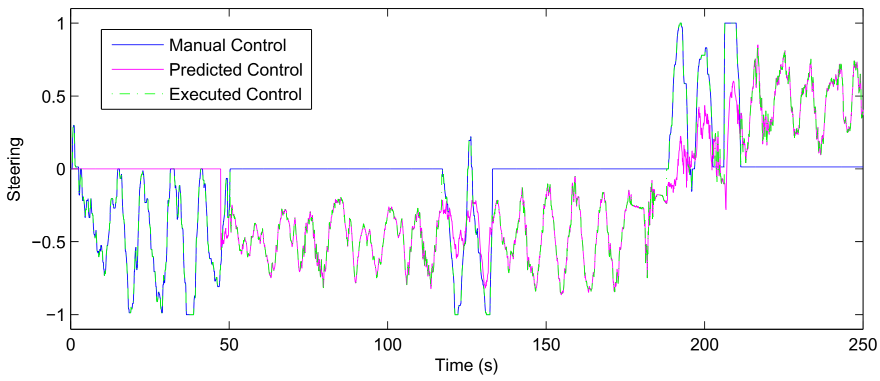


Fig. 8.10 Steering control during initial sequence, -1 is max right, 1 is max left

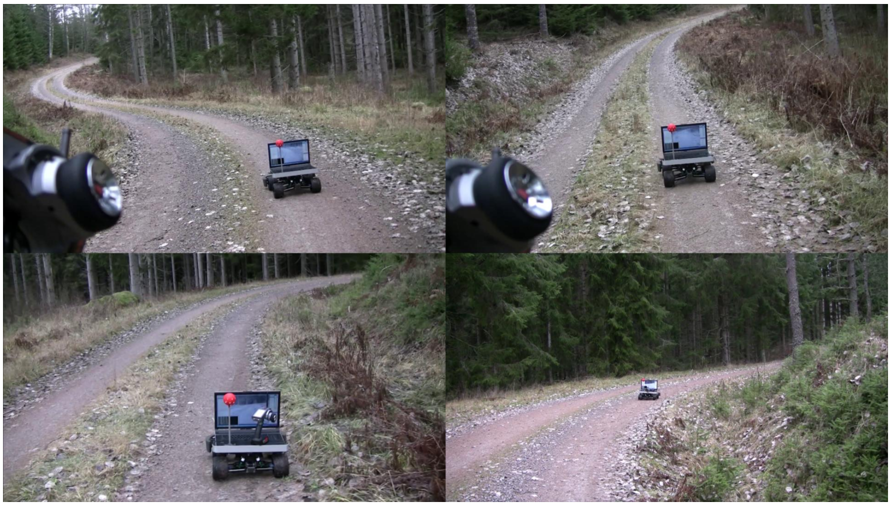


Fig. 8.11 Robot driving autonomously on a forest road

8.5 Conclusions

We have presented an online perception-action learning system capable of immediate and online switching between autonomous and learning mode. Two applications have been presented, one where the learning system is provided with enough information for self-evaluation and self-learning (inverse kinematics using explorative learning) and one where external information is required for determining the correct action (autonomous driving using learning from demonstration).

For learning of inverse kinematics, experiments have shown (i) that self-learning of inverse kinematics is possible using minimization of pose error as the explorative component, and (ii), that learning improves if utilizing information available from the already learnt model when performing exploration. The main experiments have been performed on a simulated robotic arm, however the proposed system has been able to perform successfully on a real 7 DoF robotic arm and has been implemented and used in the GARNICS demonstrator system.

The system has also shown its capability of learning a mapping from visual percepts to steering actions on a mobile robot platform. In this case, the correct action (depending on factors such as road type and desired driving style) is not available from the problem specification but is provided by means of demonstration. The same system has shown ability to learn to drive on different types of roads in different environments (indoor/outdoor).

For the indoor track, 1.5 laps of demonstration is sufficient for autonomous operation. After turning the robot around, additional demonstration during three quarters of one lap is enough for successful autonomous driving counter clockwise around the track. The robot generalizes to the last part of the track as well as to changes of the track. The outdoor experiment provides further examples of generalization capabilities, the robot is manually controlled for two minutes whereafter the robot immediately and autonomously continues along a previously unseen part of the road.

Acknowledgements. The authors would like to thank Hanno Scharr and Andreas Fischbach at ForschungszentrumJlich for access to, and assistance with the robotic arm. This work has been supported by EC's 7th Framework Programme, grant agreement 247947 (GARNICS), by SSF through a grant for the project CUAS, by VR through a grant for the project ETT, through the Strategic Areas for ICT research CADICS and ELLIIT.

References

1. Ruiz de Angulo, V., Torras, C.: Learning inverse kinematics via cross-point function decomposition. In: Dorronsoro, J.R. (ed.) ICANN 2002. LNCS, vol. 2415, pp. 856–861. Springer, Heidelberg (2002), `http://dl.acm.org/citation.cfm?id=646259.684479`
2. Atkeson, C., Moore, A., Schaal, S.: Locally weighted learning for control. Artificial Intelligence Review 11(1-5), 75–113 (1997), doi:10.1023/A:1006511328852
3. Baranes, A., Oudeyer, P.Y.: Intrinsically motivated goal exploration for active motor learning in robots: A case study. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1766–1773 (2010), doi:10.1109/IROS.2010.5651385
4. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. CoRRcs.AI/9603104 (1996)
5. de la Cruz, J.S., Kulić, D., Owen, W.: Online incremental learning of inverse dynamics incorporating prior knowledge. In: Kamel, M., Karray, F., Gueaieb, W., Khamis, A. (eds.) AIS 2011. LNCS (LNAI), vol. 6752, pp. 167–176. Springer, Heidelberg (2011), `http://dl.acm.org/citation.cfm?id=2026956.2026975`
6. Dickmanns, E., Graefe, V.: Dynamic monocular vision. Machine Vision and Applications 1, 223–240 (1988)
7. Douze, M., Jégou, H., Sandhwalia, H., Amsaleg, L., Schmid, C.: Evaluation of gist descriptors for web-scale image search. In: CIVR 2009: Proceedings of the ACM International Conference on Image and Video Retrieval (2009)

8. D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 298–303 (2001), doi:10.1109/IROS.2001.973374
9. Kastner, R., Schneider, F., Michalke, T., Fritsch, J., Goerick, C.: Image–based classification of driving scenes by a hierarchical principal component classification (HPCC). In: IEEE Intelligent Vehicles Symposium, pp. 341–346 (2009)
10. Larsson, F., Jonsson, E., Felsberg, M.: Simultaneously learning to recognize and control a low-cost robotic arm. Image Vision Computing 27(11), 1729–1739 (2009), `http://dx.doi.org/10.1016/j.imavis.2009.04.003`, doi:10.1016/j.imavis.2009.04.003
11. Lecun, Y., Muller, U., Ben, J., Cosatto, E., Flepp, B.: Off-Road Obstacle Avoidance through End-to-End Learning. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems 18, pp. 739–746. MIT Press, Cambridge (2006)
12. Markelic, I.: Teaching a robot to drive: A skill-learning inspired approach. Ph.D. thesis, University of Göttingen (2010)
13. Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal feedback control with learned internal dynamics models. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 65–84. Springer, Heidelberg (2010)
14. Moré, J.J.: The Levenberg-Marquardt algorithm: Implementation and theory. In: Watson, G.A. (ed.) Numerical Analysis, pp. 105–116. Springer, Berlin (1977)
15. Öfjäll, K.: Leap, a platform for evaluation of control algorithms. Master's thesis, Linköping University (2010)
16. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. International Journal of Computer Vision 42(3), 145–175 (2001)
17. Oyama, E., Agah, A., MacDorman, K.F., Maeda, T., Tachi, S.: A modular neural network architecture for inverse kinematics model learning. Neuro-computing 38, 797–805 (2001)
18. Oyama, E., Maeda, T., Gan, J., Rosales, E., MacDorman, K., Tachi, S., Agah, A.: Inverse kinematics learning for robotic arms with fewer degrees of freedom by modular neural network systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2005, pp. 1791–1798 (2005), doi:10.1109/IROS.2005.1545084
19. Pomerleau, D.: Alvinn: An autonomous land vehicle in a neural network. In: Proc. of NIPS (1989)
20. Pomerleau, D.: Efficient training of artificial neural networks for autonomous navigation. Neural Computation 3(1), 88–97 (1991)
21. Pourboghrat, F., Shiao, J.C.: Neural networks for learning inverse kinematics of redundant manipulators. In: Seattle International Joint Conference on Neural Networks, IJCNN 1991, vol. 2, p. 1004 (1991), doi:10.1109/IJCNN.1991.155683
22. Pugeault, N., Bowden, R.: Learning pre-attentive driving behaviour from holistic visual features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 154–167. Springer, Heidelberg (2010)
23. Pugeault, N., Bowden, R.: Driving me around the bend: Learning to drive from visual gist. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 1022–1029 (2011), doi:10.1109/ICCVW.2011.6130363
24. Renninger, L., Malik, J.: When is scene identification just texture recognition? Vision Research 44, 2301–2311 (2004)
25. Saegusa, R., Metta, G., Sandini, G., Sakka, S.: Active motor babbling for sensorimotor learning. In: IEEE International Conference on Robotics and Biomimetics, ROBIO 2008, pp. 794–799 (2009), doi:10.1109/ROBIO.2009.4913101

26. Schaal, S., Atkeson, C.: Robot juggling: implementation of memory-based learning. IEEE Control Systems 14(1), 57–71 (1994), doi:10.1109/37.257895
27. Schaal, S., Atkeson, C.: Learning control in robotics. IEEE Robotics Automation Magazine 17(2), 20–29 (2010), doi:10.1109/MRA.2010.936957
28. Schmiterlöw, M.: Autonomous path following using convolutional networks. Master's thesis, Linköping University (2012)
29. Settles, B.: Active learning literature survey. Tech. rep. (2009)
30. Siagian, C., Itti, L.: Rapid biologically-inspired scene classification using features shared with visual attention. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(2), 300–312 (2007)
31. Siagian, C., Itti, L.: Biologically inspired mobile robot vision localization. IEEE Transactions on Robotics 25(4), 861–873 (2009)
32. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics, Modelling, Planning and Control (2009) ISBN 978-1-84628-642-1
33. Sugihara, T.: Solvability-unconcerned inverse kinematics based on levenberg-marquardt method with robust damping. In: 9th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2009, pp. 555–560 (2009), doi:10.1109/ICHR.2009.5379515
34. Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P.: Stanley: The robot that won the DARPA Grand Challenge. Journal of Robotic Systems 23(9), 661–692 (2006)
35. Turk, M., Morgenthaler, D., Gremban, K., Marra, M.: VITS—a vision system for autonomous land vehicle navigation. IEEE Trans. in Pattern Analysis and Machine Intelligence 10(3), 342–361 (1988)
36. Vijayakumar, S., D'souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Comput. 17, 2602–2634 (2005), http://dl.acm.org/citation.cfm?id=1119418.1119423, doi:10.1162/089976605774320557

Chapter 9
3D Space Automated Aligning Task Performed by a Microassembly System Based on Multi-channel Microscope Vision Systems

Zhengtao Zhang, De Xu, and Juan Zhang

Abstract. In this chapter, a microassembly system based on 3-channle microvision system is proposed which achieves the automatic alignment of the mm sized complex micro parts. The design of the system is described firstly. Then two different alignment strategies are presented including position based and image Jacobian based methods. Also, a coarse-to-fine alignment strategy in combination with active zooming algorithm is proposed. In the coarse alignment stage, alignment process is conducted with maximum field of view (FOV). In the fine alignment stage, the microscope is of maximum magnification to ensure the highest assembly accuracy. For the image based control, the image jacobian based on several microscope vision systems controlling the micro part movement is derivation based on microscope vision model. It is proved that the image jacobian is a constant when controlling position movement. As active movement of the micro part, the image jacobian is online self-calibration. The PD controller is adopted to control the micro part movement quickly and effectively. The experiment verifies the effectiveness of the proposed algorithms.

9.1 Introduction

Microassembly techniques can be widely used in biology, semiconductor industry and so on. Microscope vision is the important sensing method in the microassembly. Compared to traditional vision, it has several characteristics, such as short depth of view and limited field of view (FOV), especially for high-resolution microscopes used in microassembly. Surely, there are still many problems remain unresolved, such as the contradiction between accuracy of measurement and the FOV, system calibration, control method in high precision and the online assembly parameter detection in 3D space.

Zhengtao Zhang · De Xu · Juan Zhang
Institute of Automation, Chinese Academy of Sciences,
95 Zhongguancun East Road, Beijing, China
e-mail: {zhengtao.zhang,de.xu,juan.zhang}@ia.ac.cn

Fortunately, more and more researchers focus on the key problems aforementioned [1, 2, 3, 4, 5, 6]. Aimed at contradiction between accuracy of measurement and the FOV, reference [7] used two cameras with different magnifications. But this method makes the continual visual tracking be more difficult, increases system complexity and limits workspace apparently. An active zooming strategy was proposed in [8, 9], which is based on artificial potential field (APF) and applied to the assembly task on plane. Visual servoing methods were proposed in [10] focusing on the assembly on plane.

In this chapter, we design a microvision assembly system which can realize the two mm size micro parts alignment with 3 microscopies [11]. Coarse-to-fine assembly strategy is explored to solve the contradiction between FOV and the depth of field for microscope. Two different alignment strategies are realized and compared which are image Jacobian based and position based alignment control method [12].

The reminder of this chapter is organized as follows. Section 2 introduces the hardware of the proposed microassembly system. The feature selection and relative pose calculation given in Section 3. In Section 4 the coarse-to-fine assembly strategy is presented in detail. The visual servo control based on image Jacobian is also discussed in Section 5. Experiments are given to verify the effectiveness of the algorithms in Section 6. Finally, the chapter is concluded in Section 7.

9.2 System Structure

The task for the microassembly system is to assemble two microparts together. The structure design of the system is shown in Fig. 1, which contains three microvision systems. The degree of freedom (DOF) for each subsystem is shown in table 1.

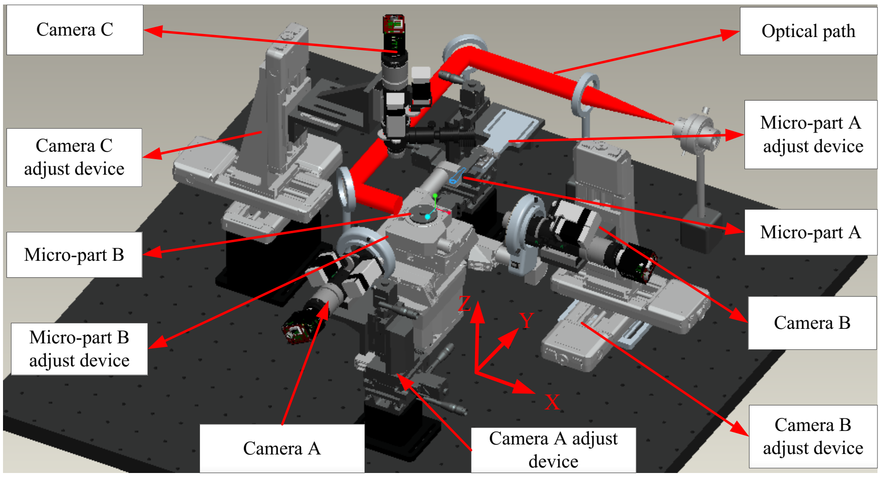


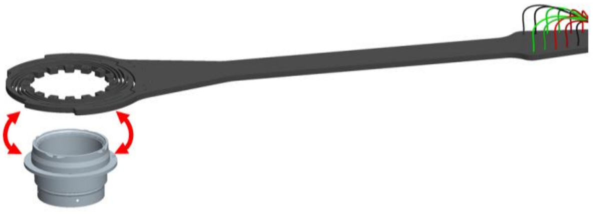
Fig. 9.1 Structure design of microassembly system

Table 9.1 DOF of subsystems

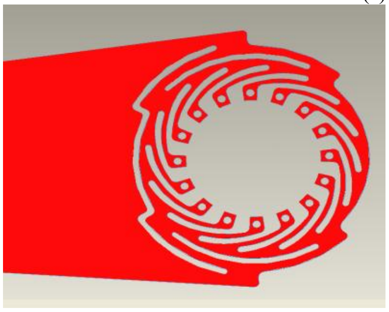
Adjustment device	Total DOFs	Electrical DOFs
Camera C	T_x, T_y, T_z, θ_x, θ_y	T_x, T_y, T_z
Camera A	T_x, T_y, T_z, θ_x, θ_z	T_x, T_y, T_z
Camera B	T_x, T_y, T_z, θ_y, θ_z	T_x, T_y, T_z
Micro-part A	T_x, T_y, T_z, θ_x, θ_y, θ_z	T_x, T_y, T_z
Micro-part B	T_z, θ_x, θ_y, θ_z	T_z, θ_x, θ_y,

The T_x, T_y, T_z are the translational DOF along the X, Y and Z axis, respectively. The θ_x, θ_y, θ_z denote the rotation DOF around the X, Y and Z axis, respectively.

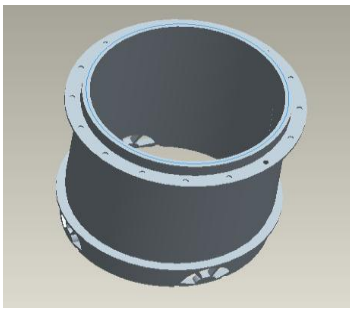
The microassembly task is shown in Fig. 2(a). The shape of micro-part B is shown in Fig. 2(b). Its outside diameter is about 55mm, height is 56mm and wall thickness is 0.2mm. The shape of micro-part A is also shown in Fig. 2(c). Both microparts have 16 small holes which are used to align each other.



(a)



(b)



(c)

Fig. 9.2 Two micro-parts Assembly, (a) assembly schematic diagram, (b) structure of micro-part A, (c) structure of micro-part B

9.3 Features Selection and Relative Pose Calculation

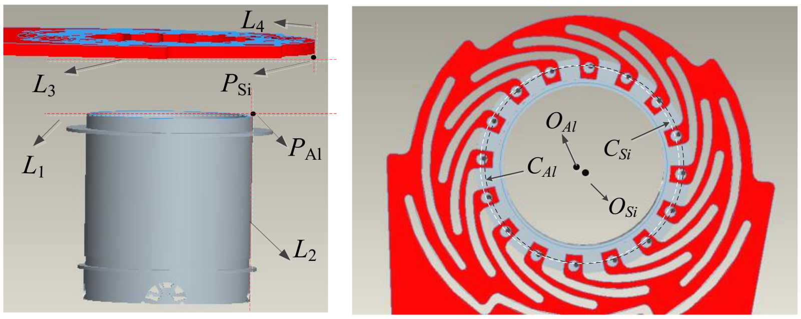
Before the alignment, some image features need to be extracted by image process algorithm as shown in Fig. 3. The feature lines are extracted by RANSAC algorithm [13]. A subpixel algorithm introduced in [14] is employed. The lines L_1 and $L_2$are the top and right edges of micro-part B in image plane captured by camera B. The lines L_3 and $L_4$are the bottom and right edges of micro-part A in image plane. P_{Al} is the point of intersection between L_1 and L_2. P_{Si} is the point of intersection between L_3 and L_4. As it is shown in Fig. 3(b), 16 circles which are served as flags to align in the top surface of micro-part A and micro-part B will be extracted. The sixteen centers of 16 circles in micro-part A and micro-part B construct another two big circles, C_{Si} in micro-part A and C_{Al} in micro-part B, respectively. The two centers O_{Si} and O_{Al} of two big circles are used to compute the distance error in X-Y plane for two microparts.

The main parameters to define the relative pose between micro-part A and micro-part B are δ_{Tx}, $\delta_{Ty,}\delta_{Tz}$and $\delta_{\theta x}$, $\delta_{\theta y}$, $\delta_{\theta z}$. δ_{Tx}, $\delta_{Ty,}\delta_{Tz}$ are calculated by equation (1).

$$\begin{cases} \delta_{Tx} = (U_{Osi} - U_{OAl}) \cdot \varepsilon \\ \delta_{Ty} = (V_{Osi} - V_{OAl}) \cdot \varepsilon \\ \delta_{Tz} = (V_{PSi} - V_{PAl}) \cdot \varepsilon \end{cases} \tag{9.1}$$

Where U_{Osi} and U_{OAl} are the X-axis coordinates of O_{Si} and O_{Al} measured from Camera C. V_{Osi} and V$_{OAl}$ are the Y-axis coordinates of O_{Si} and O_{Al}.V_{PSi} and V_{PAl} are the Y-axis coordinates of P_{Si} and P_{Al} measured from Camera A.. ε is pixel equivalence and its unit is μm/pixel.

$$\begin{cases} L_2: \quad y_2 = k_2 \cdot x_2 + b_2 \\ L_4: \quad y_4 = k_4 \cdot x_4 + b_4 \end{cases} \tag{9.2}$$



(a) (b)

Fig. 9.3 The features used to compute the relative pose, (a) horizontal view from camera B, (b) vertical view from camera C

$$\delta_{\theta x} = \arctan \frac{|k_2 - k_4|}{1 + k_2 \cdot k_4} \tag{9.3}$$

The expression of L_2 and L_4 are shown in equation (2). k_2, k_4 are the slope and b_2, b_4 are the intercept of the lines L_2 and L_4. $\delta_{\theta x}$ calculated by equation (3) denotes the angle between L_2 and L_4.$\delta_{\theta y}$ is computed in the same way from the view of camera A.

$$\delta_{\theta z} = \frac{1}{n_{si}} \sum_{i=1}^{n_{si}} \Delta\theta_i \tag{9.4}$$

$\delta_{\theta z}$ is calculated by equation (4). n_{si} is the number of the holes and equals 16. Connections from centers of the 16 holes in micro-part A to the center of C_{Si} construct 16 line segments. Also, Connections from centers of the 16 holes in micro-part B to the center of C_{Al} construct another 16 line segments. $\Delta\theta_i$ is the angle between two adjacent line segments in micro-part A and micro-part B.

9.4 Coarse-to-Fine Alignment Strategy with Active Zooming Algorithm

9.4.1 Coarse Alignment

The assembly schematic diagram is shown in Fig. 2 (a) and the alignment order is executed sequentially in the guide of camera A, B and camera C. In the coarse alignment phase, the micro-part A is in the view of three cameras and the micro-part B is further away.

Because the adjustment process in guide of camera A is similar to the one in camera B as shown in Fig. 4, we take the alignment process in camera B for example. When micro-part A is focused in camera B, its bottom and right edges are extracted and fitted as shown in Fig. 3(a). Then, the micro-part B moves into the views of camera B as shown in Fig. 5(a). The right side edge of micro-part B is extracted and fitted by RANSAC algorithm.δ_{Tz} and $\delta_{\theta x}$ is calculated by equation (1) and (3). After the adjustment of $\delta_{\theta x}$ by micro-part B adjust device, the relative pose of two microparts is shown in Fig.5 (b). At last, by translational adjustment, the two parts become closer as shown in Fig.5 (c).

The alignment in camera C is shown in Fig. 6. Firstly, the micro-part A moves out of the view, leaving Al alone as shown in Fig. 6(a). Then, the camera C is focused on the top side of the micro-part B and 16 small circles and the center of circle O_{Al} are extracted. Afterwards, the micro-part A moves into the view of camera C as shown in Fig 6(b). Another 16 circles are extracted with the center of circle O_{Si}. δ_{Tx}, δ_{Ty} are calculated according to equation (1) and $\delta_{\theta z}$ is computed with equation (4). Finally, the pose of micro-part B is adjusted with δ_{Tx}, δ_{Ty}, $\delta_{\theta z}$.

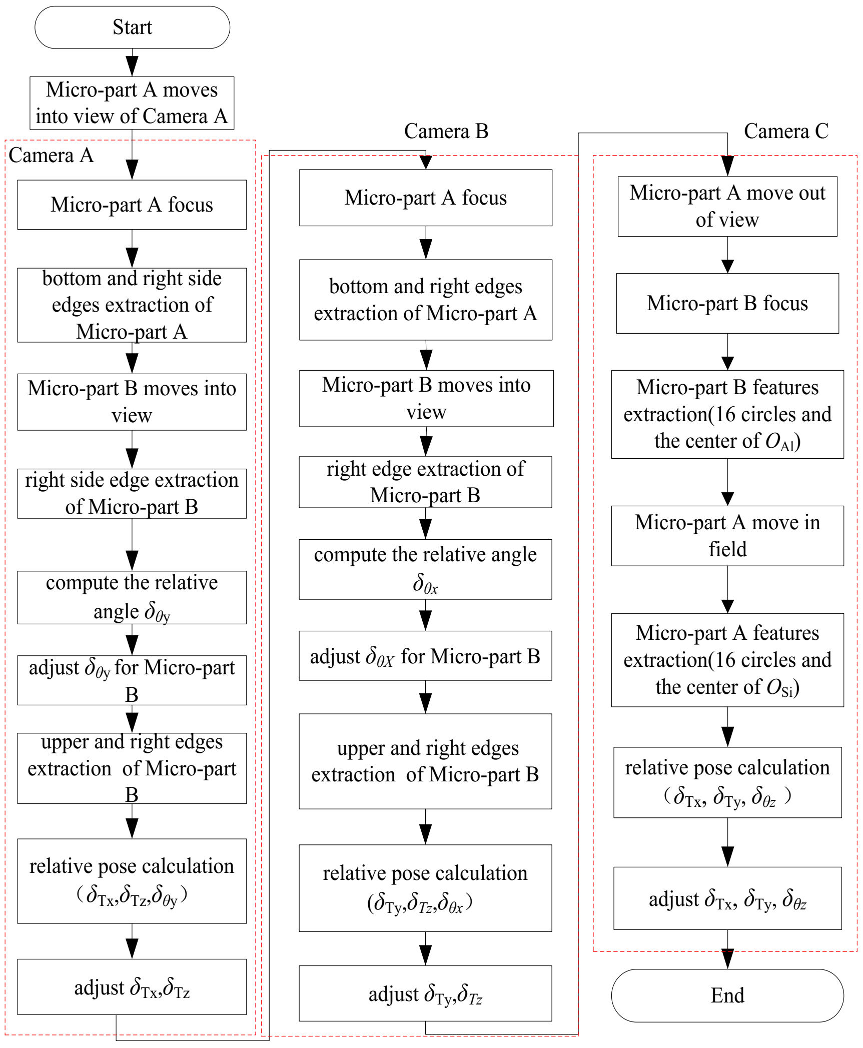

Start
Micro-part A moves into view of Camera A
Camera A
Micro-part A focus
bottom and right side edges extraction of Micro-part A
Micro-part B moves into view
right side edge extraction of Micro-part B
compute the relative angle $\delta_{\theta y}$
adjust $\delta_{\theta y}$ for Micro-part B
upper and right edges extraction of Micro-part B
relative pose calculation (δ_{Tx},δ_{Tz},$\delta_{\theta y}$)
adjust δ_{Tx},δ_{Tz}
Camera B
Micro-part A focus
bottom and right edges extraction of Micro-part A
Micro-part B moves into view
right edge extraction of Micro-part B
compute the relative angle $\delta_{\theta x}$
adjust $\delta_{\theta X}$ for Micro-part B
upper and right edges extraction of Micro-part B
relative pose calculation (δ_{Ty},δ_{Tz},$\delta_{\theta x}$)
adjust δ_{Ty},δ_{Tz}
Camera C
Micro-part A move out of view
Micro-part B focus
Micro-part B features extraction(16 circles and the center of O_{Al})
Micro-part A move in field
Micro-part A features extraction(16 circles and the center of O_{Si})
relative pose calculation (δ_{Tx}, δ_{Ty}, $\delta_{\theta z}$)
adjust δ_{Tx}, δ_{Ty}, $\delta_{\theta z}$
End

Fig. 9.4 Alignment flowchart

9.4.2 Active Zooming

The proportional relationship between image plane and Cartesian coordinate is represented by μm/pixel called pixel equivalent. The magnification factor has an effect on the pixel equivalent which is calibrated and shown in Fig.7. The horizontal axis of Fig.7 is denoted by pulse because of the zooming driver of microscopy is stepping motor. It can be seen from Fig.7 that the pixel equivalent decreases from 4.9 to 3.13 for camera A. Therefore, with greater magnification factor has higher resolution.

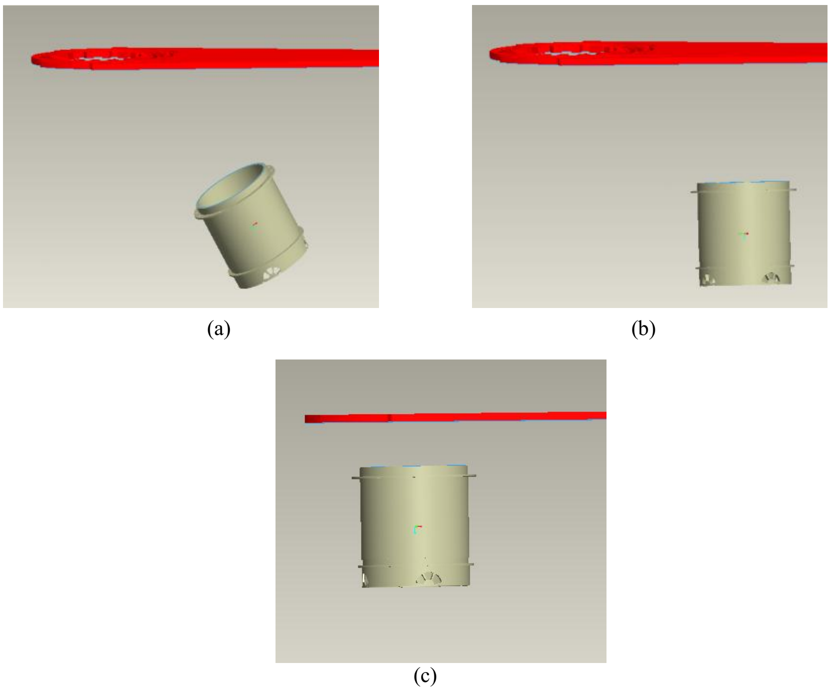


Fig. 9.5 Schematic diagram of coarse alignment in camera B, (a) $\delta\theta$x calculation (b) $\delta\theta$x adjustment (c) translational adjustment

When the two microparts become closer, the active zooming strategy is employed to improve the accuracy of measurement. The schematic diagram is shown in Fig.8 and the right edge of the micro-part A is chosen as the features to be traced. Firstly, the feature area S_1 is saved as template. The search area is S_e for next frame depending on the zooming step. The area S_2 is found based on the template S_1 in the process of zooming.

The normalized correlation matching method (NCMM) is employed. The searching algorithm is shown in equation (5).

$$R(x,y) = \frac{\sum_{x\prime,y\prime}[T(x\prime,y\prime)\cdot I(x+x\prime,y+y\prime)]}{\sqrt{\sum_{x\prime,y\prime}T(x\prime,y\prime)^2\cdot\sum_{x\prime,y\prime}I(x+x\prime,y+y\prime)^2}} \tag{9.5}$$

where (x,y) means the image coordinate in the sampled frame, I(x,y) means the grey value in (x,y). (x’,y’) means the coordinate in template area. T(x’,y’) means the grey value in (x’,y’). R(x,y) means the match result.

If equation (6) is 0 which means the feature point F shown in Fig.8 reaches the safe margin, the zooming process stops.

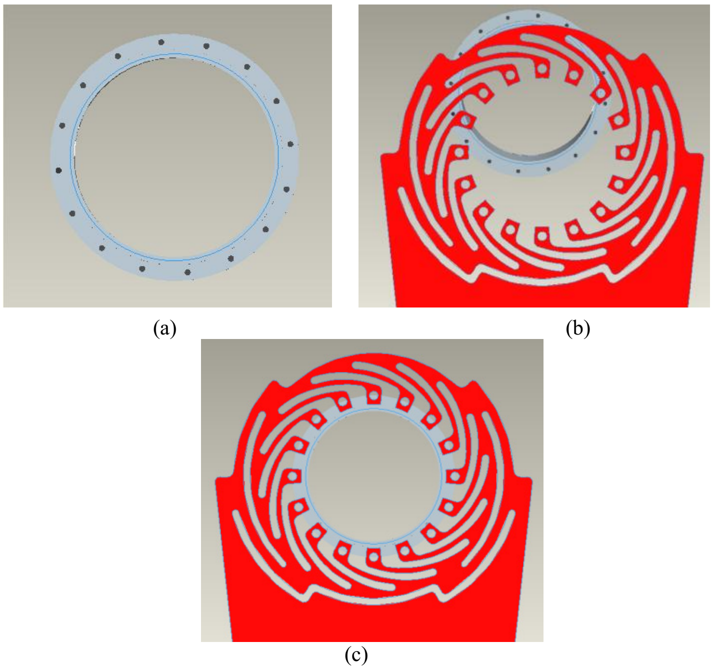


Fig. 9.6 Schematic diagram of coarse alignment in camera C, (a) micro-part B features extraction (b) micro-part A features extraction (c) adjustment with δ_{Tx}, δ_{Ty}, $\delta_{\theta z}$

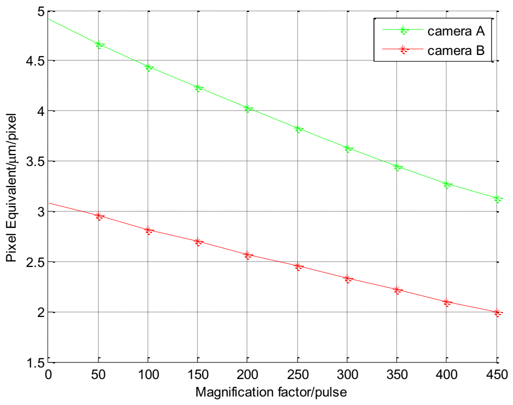


Fig. 9.7 Relationship between magnification factor and pixel equivalent

$$\begin{cases} 0 & if\ x_f - \zeta \le 0, x_f + \zeta \ge U, y_f - \zeta \le 0, y_f + \zeta \ge V \\ 1 & other \end{cases} \qquad (9.6)$$

where (x_f, y_f) is the coordinate of the feature point, ζ is the distance between safe margin and the frame margin. U is the width of the frame and V is the height of the frame.

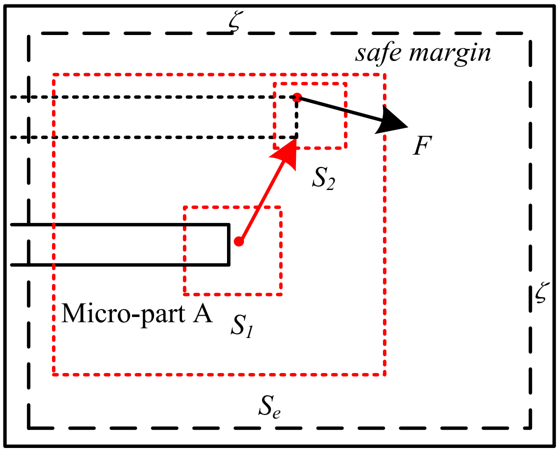


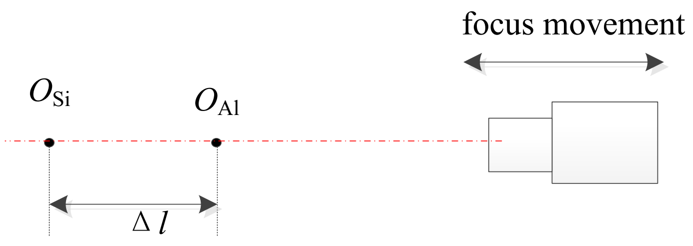
Fig. 9.8 Zooming strategy

9.4.3 Fine Alignment

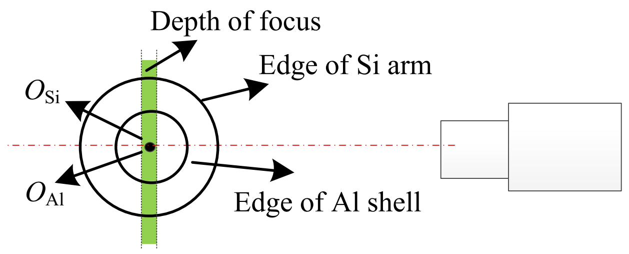
In the fine alignment stage, the flowchart is nearly the same as it is in the coarse stage but with higher magnification and without complex zooming action. There are two important factors influence the accuracy of measurement for the assembly system except the magnification factor shown in Fig.7. At coarse alignment stage, microparts are far away from each other. The camera has to do focus movement as shown in Fig. 9(a). The movement Δl of motion mechanism will bring more error into the measurement.

The other factor can be seen from Fig. 10 which shows the principle of zoom for microscopy. The adjusting part inside the microscopy is some optical lenses. The motion of the lenses will cause the change of optical path and optical axis. Then, the change of the imaging decreases the accuracy of the measurement.

However, after the alignment of coarse phase, the center O_{si} and O_{Al} are nearly coincides as shown in Fig. 9(b). The features can be extracted without repeated focusing. Then the two important factors can be evitable. So, higher magnification without complex focus movement and zooming adjustment increase the accuracy of the measurement.



(a)



(b)

Fig. 9.9 Focus in coarse and fine stage, (a) The movement during focusing, (b) focusing without movement in fine stage

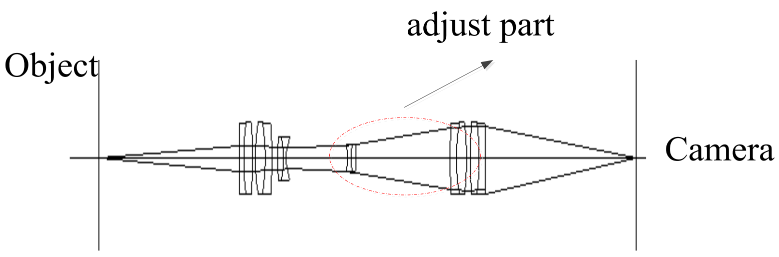


Fig. 9.10 Principle of zoom for microscopy

9.5 Vision Servo Control Based on Jacobian

During coarse alignment stage, the position-based vision servo method is employed. However, image-based vision servo is more suitable for the micro-assembly system. This section the image Jacobian which describes the relationship between changes of image features and the motion of micro-part in Cartesian space is deduced. Then the online calibration method is described and the controller is designed finally.

9.5.1 Image Jacobin Matrix Derivation

Micro-part A has 3 translational DOF and it is measured by 3 microscopes. So, its Jacobian matrix denotes the relationship between translational speed of micro-part A and its image features changing. The coordinate of micro-part A in world coordinate is defined as (7). Image feature space of Micro-part A is defined as (8). $[u_i(t), v_i(t)]$ is the image features change in i-th channel microscope and i =1,2 3. Their relationship is shown in (9).$J_G(p)$ is the Jacobian matrix of the motion control for the micro-part A.

$$p(t)=[x_b(t), y_b(t), z_b(t)]^T \tag{8.7}$$

$$s(t)=[u_1(t), v_1(t), u_2(t), v_2(t), u_3(t), v_3(t)]^T \tag{8.8}$$

$$\dot{s} = J_G(p)\dot{p} \tag{9.9}$$

Micro-vision system is comprised of microscope lens and CCD which is different from normal camera imaging principle. The projection imaging relationship is shown in (10). Because of the limited depth of field, if the object keep clear on the image plane after the automatic focusing, z_{ic} can be regard as constant. The transformation relation for the micro-part A from word coordinate to i-th camera coordinate is shown in (11). $[x_{ic}, y_{ic}, z_{ic}]$ is the coordinate of micro-part A in i-th camera. δ_i is the proportionality coefficient.

$$\begin{cases} u_i = \delta_i \frac{x_{ic}}{z_{ic}} \\ v_i = \delta_i \frac{y_{ic}}{z_{ic}} \end{cases} \tag{9.10}$$

$$\begin{bmatrix} x_{ic} \\ y_{ic} \\ z_{ic} \end{bmatrix} = R_i \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} + T_i \tag{9.11}$$

where,

$$R_i = \begin{bmatrix} r_{i11} & r_{i12} & r_{i13} \\ r_{i21} & r_{i22} & r_{i23} \\ r_{i31} & r_{i32} & r_{i33} \end{bmatrix}, \quad T_i = \begin{bmatrix} T_{ix} \\ T_{iy} \\ T_{iz} \end{bmatrix}$$

Set $\lambda_i=\delta_i/z_{ic}$, (12) can be deduced by (10) and (11). Then, the Jacobian matrix used to control the motion of micro-part A based on single camera is shown in (13).

$$\begin{bmatrix} \dot{u}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} \lambda_i \dot{x}_{ic} \\ \lambda_i \dot{y}_{ic} \end{bmatrix} = \begin{bmatrix} \lambda_i r_{i11} & \lambda_i r_{i12} & \lambda_i r_{i13} \\ \lambda_i r_{i21} & \lambda_i r_{i22} & \lambda_i r_{i23} \end{bmatrix} \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix} \tag{9.12}$$

$$J_i = \begin{bmatrix} \lambda_i r_{i11} & \lambda_i r_{i12} & \lambda_i r_{i13} \\ \lambda_i r_{i21} & \lambda_i r_{i22} & \lambda_i r_{i23} \end{bmatrix} \tag{9.13}$$

If the rotation relationship between world coordinate and camera coordinate remains unchanged, R_i is constant. Also, if the magnification of the microscope keeps unchanged, λ_i is constant. It can be seen from (13) that J_i is constant under these two conditions. The Jacobian matrix based on three cameras is shown in (14). J_1, J_2, $J_3$are according Jacobian matrixes for camera A, B and C.

$$J_G = [J_1, J_2, J_3]^T \tag{9.14}$$

Similarly, the Jacobian matrix for micro-part B can be deduced which is shown in (15) when it rotates in the world coordinate. $\dot{\partial}$, $\dot{\beta}$ are the differential of α, β which are shown in Fig.11. $\dot{\theta}_x$, $\dot{\theta}_y$ are the differential of θ_x and θ_y. θ_x and θ_y are the angles of micro-part B around X and Y axis in world coordinate.

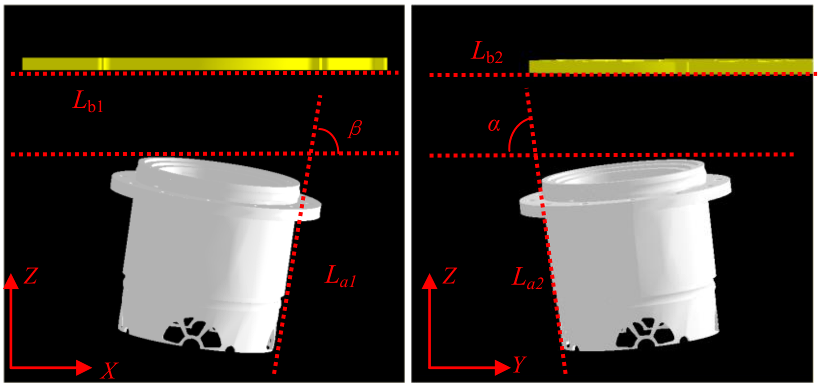
$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} J_{T11} & J_{T12} \\ J_{T21} & J_{T22} \end{bmatrix} \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \end{bmatrix} \tag{9.15}$$

9.5.2 Feature Select for the Image Jacobian Control

The image features used to adjust the pose of micro-part B are shown in Fig.11. The edge lines L_{a1} and L_{a2} are chosen as features. α, β are the angle of the two lines. The desired attitude of micro-part B is computed according to the lines L_{b1} and L_{b2}. which is shown in (16). k_{b1}, k_{b2} are the slope of L_{b1} and L_{b2}. θ_{e1},θ_{e2} are the image angle errors in microscope A and B in image plane.

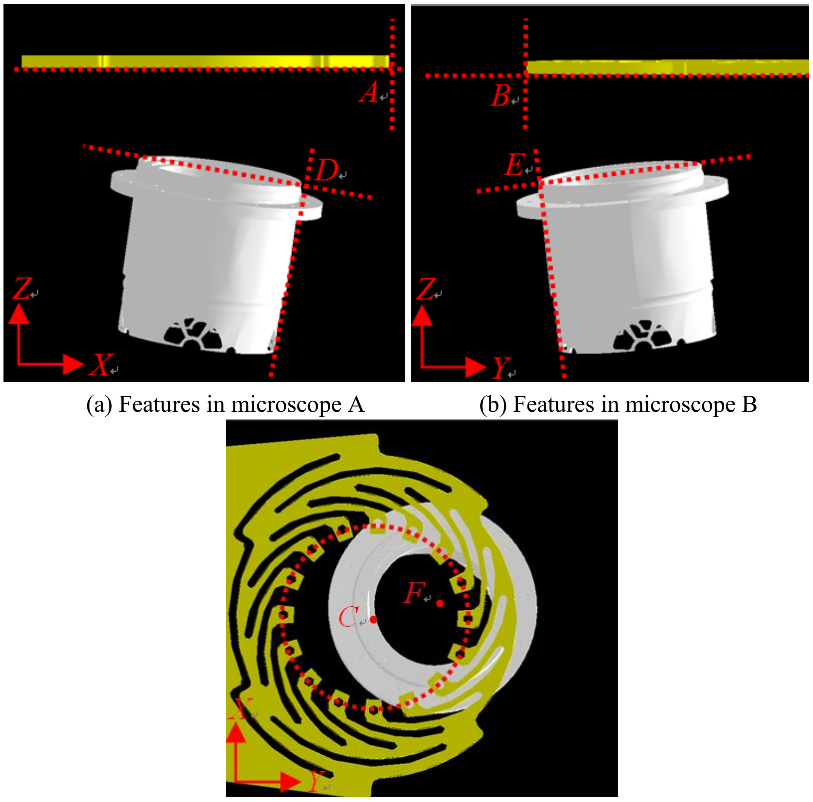
$$\begin{cases} \theta_{e1} = \arctan(k_{\mathrm{b}1}) + \frac{\pi}{2} - \beta \\ \theta_{e2} = \arctan(k_{b2}) + \frac{\pi}{2} - \alpha \end{cases} \tag{9.16}$$

Image features related to micro-part A position adjustment are shown in Fig.12. Image corners A and B in microscope A and B in image plane are chosen as image features. Feature point C is the center of the micro-part A in image plane. The



(a) Features in microscope A (b) Features in microscope B

Fig. 9.11 Image features related to micro-part B pose adjustment



(a) Features in microscope A
(b) Features in microscope B
(c) Features in microscope C

Fig. 9.12 Image features related to micro-part A position adjustment

desired positions in image plane can selected according to the feature points D, E and F. D, E is feature points and F is the center of micro-part B in image plane observed from microscope C.

9.5.3 Online Self-calibration for Jacobian

By the linear independent movement of motion axis, the Jacobian matrix can be calibrated autonomously. Let the micro-part B rotate around X and Y axis separately and linear independently with $\mathrm{d}\theta_{x1}$, $\mathrm{d}\theta_{y2}$. The image Jacobian of micro-part B J_T is shown in (17). $(\Delta\alpha_1, \Delta\beta_1)$, $(\Delta\alpha_2, \Delta\beta_2)$ are the image feature errors when micro-part B rotates around X and Y axis.

$$J_T = \begin{bmatrix} \Delta\alpha_1 & \Delta\alpha_2 \\ \Delta\beta_1 & \Delta\beta_2 \end{bmatrix} \begin{bmatrix} d\theta_{x1} & 0 \\ 0 & d\theta_{y2} \end{bmatrix}^{-1} \tag{9.17}$$

The computation of image Jacobian for micro-part A is shown in (18). dT_{x1}, dT_{y2}, dT_{z3} are 3 linear independence translational motion. $(\Delta u_{ij}, \Delta v_{ij})$ is the image feature errors in the *j*-th microscope in the *i*-th movement (*i,j*=1,2,3).

$$J_G = \begin{bmatrix} \Delta u_{11} & \Delta u_{21} & \Delta u_{31} \\ \Delta v_{11} & \Delta v_{21} & \Delta v_{31} \\ \Delta u_{12} & \Delta u_{22} & \Delta u_{32} \\ \Delta v_{12} & \Delta v_{22} & \Delta v_{32} \\ \Delta u_{13} & \Delta u_{23} & \Delta u_{33} \\ \Delta u_{13} & \Delta u_{23} & \Delta u_{33} \end{bmatrix} \begin{bmatrix} dT_{x1} & 0 & 0 \\ 0 & dT_{y2} & 0 \\ 0 & 0 & dT_{z3} \end{bmatrix}^{-1} \tag{9.18}$$

9.5.4 Controller Design for Image Servo Based on Jacobian

Take the motion control of micro-part A for example, its control block diagram is shown in Fig.13. By the image processing, the image features can be easily obtained. Compared with the expected image position, the image errors can be acquired. The errors can be transferred to from image plane to the 3D Cartesian space. By the PD controller, the motion of micro-part A can be controlled. The adjustment of the micro-part B is similar.

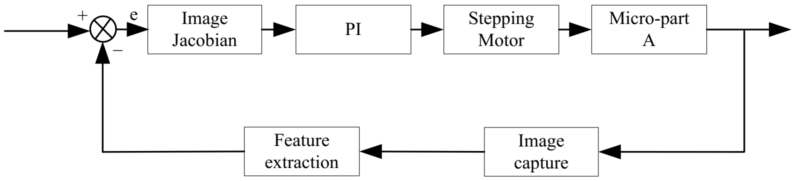


Fig. 9.13 The block diagram of micro-part A movement control

The control law based on the feedback error is shown in (19).

$$\Delta u(k) = \begin{cases} K_P((J^TJ)^{-1}J^Te(k)) + K_D((J^TJ)^{-1}J^T\Delta e(k)) & |e(k)| \geq e_T \\ 0 & |e(k)| < e_T \end{cases} \tag{9.19}$$

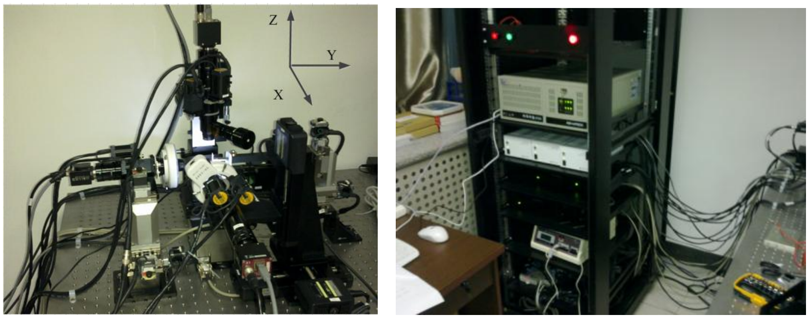
Where,

$\Delta u(k)$ is current motion control value;
K_P, K_D are the proportional and differential coefficient;
*J*I is the image Jacobian matrix calibrated by the online method;
$e(k)$ is the current image error;
$\Delta e(k)$ is the variation of the image error;
e_TI is the given error range.

9.6 Experiments and Results

9.6.1 Hardware Setup

The experimental system is shown in Fig. 14(a) which contains 25 degree of freedoms with 3-channels microvision systems. The control cabinet is shown in Fig. 14(b). The image size of the camera is 2440×2050 pixel with 15 Frames/s. The magnification of microscope is 0.71∼4.5X according field of view is 15.6∼2.44mm. The depth of field is 0.43∼0.04mm. The motion resolution of each adjust device is 1μm.



(a) (b)

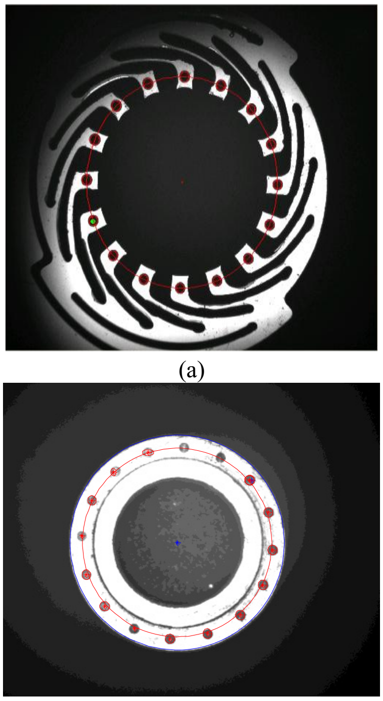
Fig. 9.14 Experimental device, (a) assembly platform, (b) control cabinet

9.6.2 Error Analysis for the Position-Based Method

The angle and position error analysis is conducted. The precision of micro-part B and micro-part A adjustment device is higher than the vision detection system. The translational accuracy is 1μm and angular accuracy is 0.02 degree. On this condition, the movement of the motion is considered as true value. The measurement of the vision system is regarded as measurement value.

The micro-part B adjustment device moves 1000μm along the X axis. Then, sixteen holes of micro-part A and micro-part B with the center of C_{Si} are extracted as shown in Fig. 15. δ_{Tx} is computed according to equation (1). The measurement errors of 20 times for δ_{Tx} are shown in Fig. 16(a). The error data is shown in Table 2. It can be seen that the maximum error is 4.9μm, average value is 1.7μm and variance is 1.6163.

In the same way, the measurement errors for $\delta_{\theta z}$ are shown in Fig. 16(b). The error data is shown in Table 3. It can be seen that the maximum error is 0.2^o, average error is 0.1^o and variance is 0.0043.



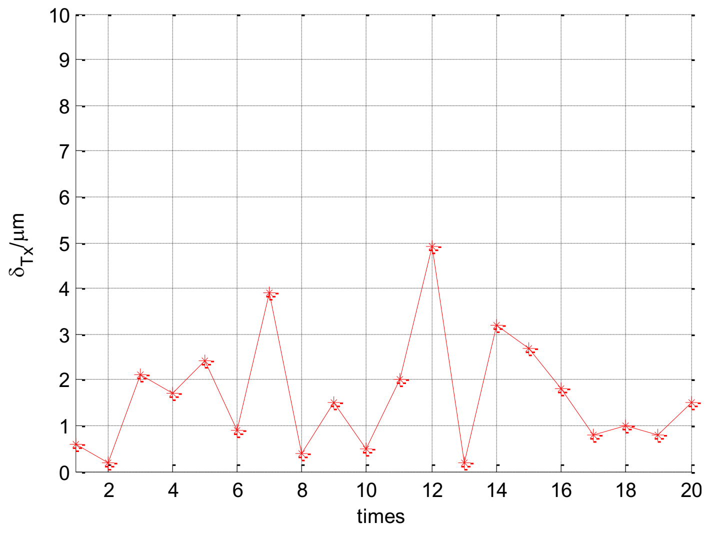
(a)

(b)

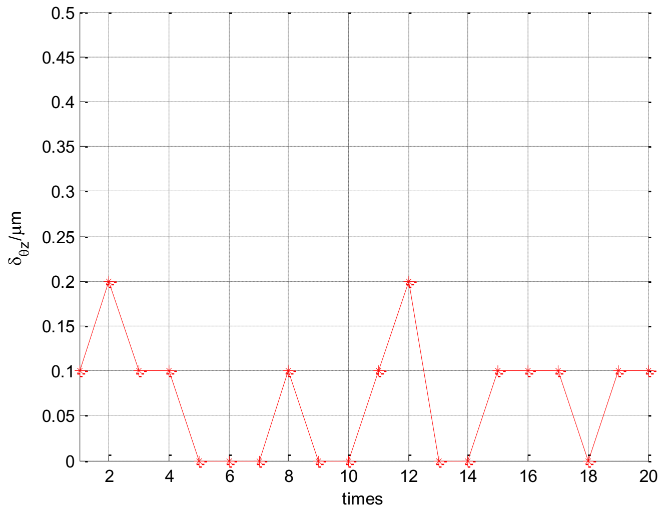
Fig. 9.15 Feature extraction (a) Sixteen holes of micro-part A and center of C_{Si}, (b) Sixteen holes of micro-part B and center of C_{Al}

Table 9.2 Error date for δ_{Tx}

times	Error (μm)	times	Error (μm)	times	Error (μm)
1	-0.6	8	0.4	15	-2.7
2	-0.2	9	1.5	16	-1.8
3	2.1	10	0.5	17	-0.8
4	-1.7	11	-2.0	18	-1.0
5	2.4	12	-4.9	19	-0.8
6	0.9	13	-0.2	20	-1.5
7	3.9	14	-3.2		



(a)



(b)

Fig. 9.16 Error of δ_{Tx} and $\delta_{\theta z}$, (a) errof of δ_{Tx}, (b) error of $\delta_{\theta z}$

Table 9.3 Error data for $\delta_{\theta z}$

times	Error (μm)	times	Error (μm)	times	Error (μm)
1	-0.1	8	0.1	15	-0.1
2	0.2	9	0.0	16	-0.1
3	0.1	10	0.0	17	0.1
4	-0.1	11	0.1	18	-0.0
5	0.0	12	-0.2	19	0.1
6	0.0	13	-0.0	20	-0.1
7	-0.0	14	-0.0		

The feature extraction is shown in Fig.17 which is used to compute the relative pose of two microparts as shown in Fig. 3(a). The red line denotes the estimated object area and the green line denotes the feature lines.

The measurement error of $\delta_{\theta x}$ is shown in Fig.18(a) and error data is shown in Table 4. It can be seen from that the maximum error is 0.4^o, average error is 0.2^o and variance is 0.0087. The measurement error of δ_{Tz} without zooming is shown in Fig.18(b). The error data is shown in Table 5. It can be seen that the maximum error is 7.0μm, average value is 4.9μm and variance is 0.9734. In order to improve the accuracy of the measurement in Z axis, the camera zoomed with the strategy described in Part 4.B. The measurement error of δ_{Tz} after active zooming is shown in Fig.18(c) and error data is shown in Table 6. It can be seen that the maximum error is 3.9μm, average value is 2.9μm and variance is 0.2617. The accuracy of measurement is improved obviously.

Table 9.4 Error data for $\delta_{\theta x}$

times	Error (degree)	times	Error (degree)	times	Error (degree)
1	0.2	8	0.4	15	0.1
2	0.3	9	0.3	16	0.2
3	0.2	10	0.3	17	0.3
4	0.3	11	0.1	18	0.1
5	0.1	12	0.4	19	0.3
6	0.3	13	0.2	20	0.2
7	0.2	14	0.2		

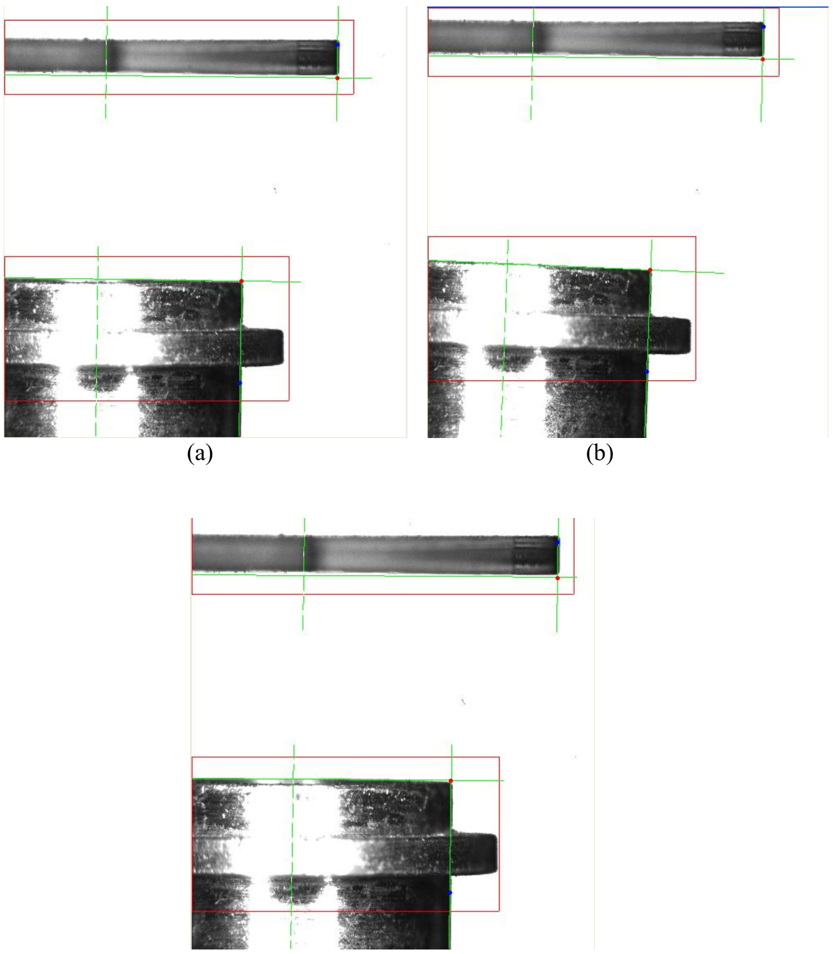
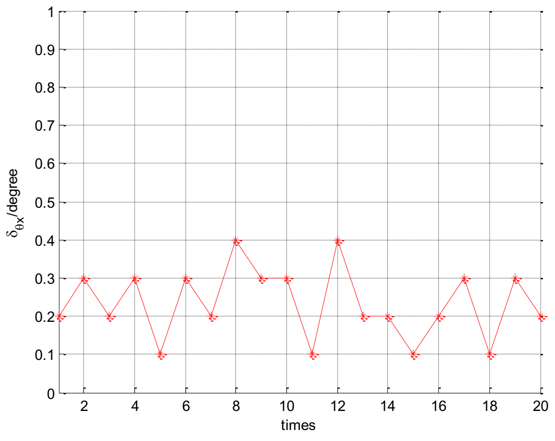


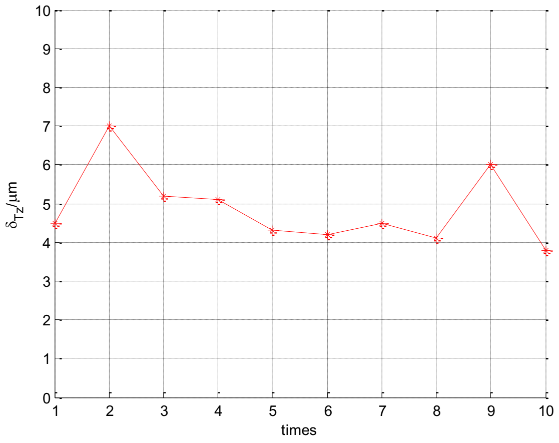
Fig. 9.17 Image process and feature extraction in camera B, (a) calculation of δ_{Tz} without zooming, (b) calculation of $\delta_{\theta x}$, (c) calculation of δ_{Tz} after active zooming

Table 9.5 Error data for δ_{Tz} without zoom

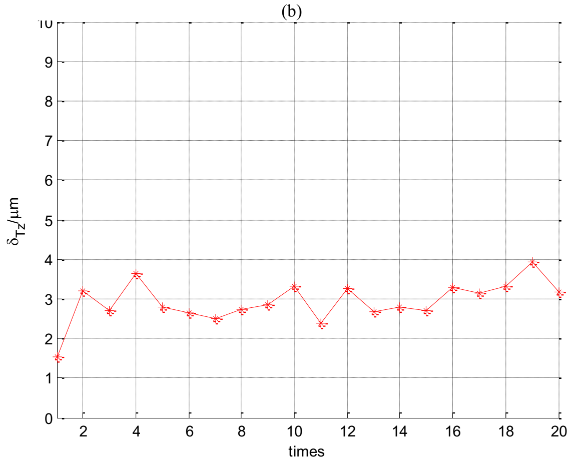
times	1	2	3	4	5	6	7	8	9	10
Error (μm)	4.5	7.0	5.2	4.3	4.3	4.2	4.5	4.1	6.0	3.8



(a)



(b)



(c)

Fig. 9.18 Error of $\delta_{\theta x}$ and δ_{Tz}, (a) Error of $\delta_{\theta x}$, (b) error of δ_{Tz} without zooming, (c) error of δ_{Tz} after zooming

Table 9.6 Error data for δ_{T_Z} after zooming

times	Error (μm)	times	Error (μm)	times	Error (μm)
1	-1.5	8	-2.7	15	-2.7
2	-3.2	9	-2.9	16	-3.3
3	-2.7	10	-3.3	17	-3.2
4	-3.7	11	-2.4	18	-3.3
5	-2.8	12	-3.3	19	-3.9
6	-2.7	13	-2.7	20	-3.2
7	-2.5	14	-2.8		

9.6.3 Image Servo Control Based on Jacobian Matrix

During the attitude adjust for the micro-part A, the calibration result of Jacobian matrix and B are shown in (20), (21). and B. The parameters for PD controller is set K_p=0.5, K_d=0.1and e_T is 0.2 degree. The experiment result is shown in Fig.19. During the position adjust for micro-part B, the calibration result of Jacobian matrix is shown in (21). The parameters for PD controller is set K_p=0.7, K_d=0.1 and e_T is 5μm. The experiment result is shown in Fig. 20. It can be seen from Fig. 19 and 20 that, the pose errors converge to the given error range rapidly. Fig. 21 shows the images of the two micro-parts before and after alignment. Experiments shows that the method proposed in this chapter can adjust the pose in 3D space very fast and effectively.

$$J_G = \begin{bmatrix} -0.0010 & -0.2070 & 0 \\ -0.0070 & -0.0010 & -0.1990 \\ 0.3210 & 0 & 0 \\ 0.0040 & 0.0030 & -0.3270 \\ 0.2550 & 0.0140 & -0.0110 \\ -0.0040 & -0.1950 & -0.0010 \end{bmatrix} \tag{9.20}$$

$$J_T = \begin{bmatrix} 1.0 & 0.0 \\ -0.2 & 1.2 \end{bmatrix} \tag{9.21}$$

Compared the two alignment methods, the position based algorithm is more acceptable. But the control time is much longer than the Jacobian method, because of the coupling of motion axes. The image Jacobian matrix based alignment method is more fast and effectively with the same precision.

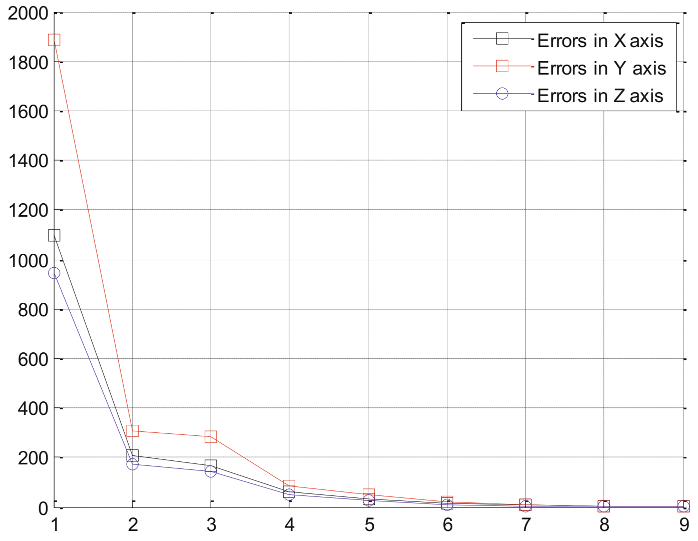


Fig. 9.19 Error in the micro-part B movement control process

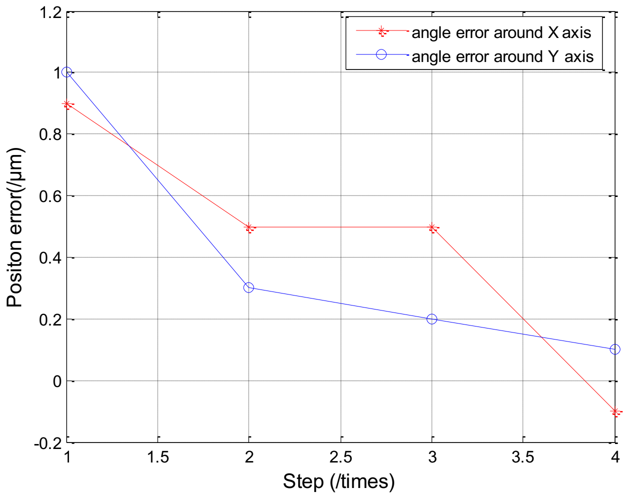
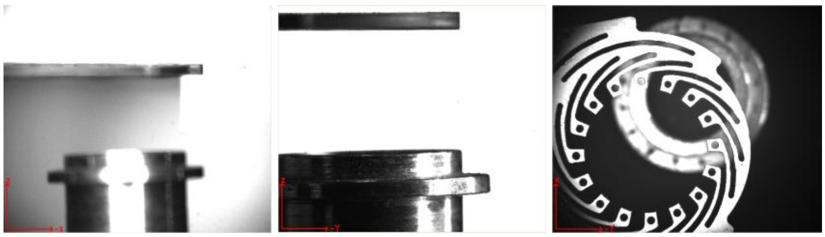


Fig. 9.20 Error in the micro-part A movement control process


(a)

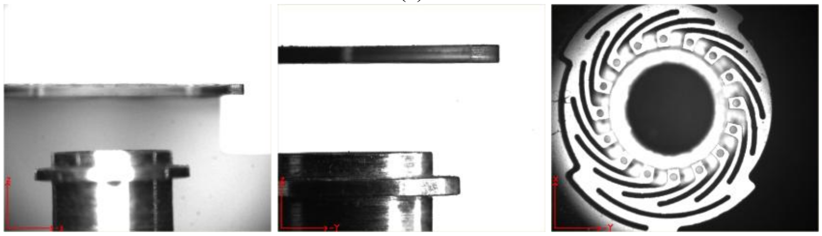

(b)

Fig. 9.21 Images before and after alignment process, (a) Before alignment, (b) After alignment

9.7 Conclusion

In this paper, a microvision assembly system which can realize the two mm size micro parts alignment with 3 microscopies is introduced. A kind of coarse-to-fine assembly strategy is proposed in the guide of 3 cameras. Position based alignment strategy is presented. It can realize the alignment task in the coarse alignment stage when the two microparts is far away from each other at first. Then the system can achieve better accuracy at final alignment stage in combination with active zooming. Also, the image Jacobian based 3D space control method is proposed including the Jacobian matrix deduce, self-calibration method and the controller design. With the features selected, the microvision assembly system can measure the assembling parameters online precisely. The experiment results show that the measurement error for position is less than 5μm and the error for angle is less than 0.5^o based on this system.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under grant 61105036.

References

1. Zhang, Z., Zhang, J., Li, H., Liu, W.: Structure Design and Application of Micro-gripper in Cryogenic Target Assembly. In: Proc. IEEE International Conference on Computer Science and Automation Engineering, Zhangjiajie (2012)

2. Zhang, J., Zhang, Z., Xu, D., Zhang, W.: Aligning Micro-gripper to Ring Object in High Precision with Microscope Vision. In: Proc. IEEE International Conference on Computer Science and Automation Engineering, Zhangjiajie (2011)
3. Yang, G., Gaines, J.A., Nelson, B.J.: Optomechatronic design of microassembly systems for manufacturing hybrid microsystems. IEEE T. Ind. Electron. 52(4), 1013–1023 (2005)
4. Feddema, J.T., Simon, R.W.: CAD-driven microassembly and visual Servoing. In: 1998 Ieee International Conference on Robotics and Automation, vol. 2, pp. 1212–1219 (1998)
5. Murthy, R., Das, A., Popa, D.: Multiscale Robotics Framework for MEMS Assembly. In: Proc. 9th International Conference on Control, Automation, Robotics and Vision, ICARCV 2006 (2006)
6. Zhou, Y., Nelson, B.J., Vikramaditya, B.: Fusing force and vision feedback for micromanipulation. In: 1998 IEEE International Conference on Robotics and Automation, vol. 2, pp. 1220–1225 (1998)
7. Ralis, S.J., Vikramaditya, B., Nelson, B.J.: Micropositioning of a weakly calibrated microassembly system using coarse-to-fine visual servoing strategies. Ieee T. Electron Pack 23(2), 123–131 (2000)
8. Tao, X.D., Janabi-Sharifi, F., Cho, H.: An Active Zooming Strategy for Variable Field of View and Depth of Field in Vision-Based Microassembly. Ieee T. Autom. Sci. Eng. 6(3), 504–513 (2009)
9. Tao, X.D., Janabi-Sharifi, F., Cho, H.: An active zooming method for and depth of field in simultaneous control of field of view vision-based microassembly. In: Proc. Spie, vol. 6719 (2007)
10. Ferreira, A., Cassier, C., Hirai, S.: Automatic microassembly system assisted by vision servoing and virtual reality. Ieee-Asme T. Mech. 9(2), 321–333 (2004)
11. Zhang, Z., Zhang, J., Xu, D.: Design of Microassembly System and Research on Coarse-to-Fine Alignment Strategy in combination with Active Zooming
12. Juan, Z., De, X., Zhengtao, Z.: liyan, L.: An Automatic Alignment Strategy of Micro Parts Based on Microscope Vision Systems. ROBOT (2013)
13. Fischler, M.A., Bolles, R.C.: Random Sample Consensus - a Paradigm for Model-Fitting with Applications to Image-Analysis and Automated Cartography. Commun. Acm 24(6), 381–395 (1981)
14. Lyvers, E.P., Mitchell, O.R., Akey, M.L., Reeves, A.P.: Subpixel Measurements Using a Moment-Based Edge Operator. Ieee T. Pattern Anal. 11(12), 1293–1309 (1989)

Chapter 10
Intensity-Difference Based Monocular Visual Odometry for Planetary Rovers

Geovanni Martinez

Abstract. A monocular visual odometry algorithm is presented that is able to estimate the rover's 3D motion by maximizing the conditional probability of the intensity differences between two consecutive images, which were captured by a monocular video camera before and after the rover's motion. The camera is supposed to be rigidly attached to the rover. The intensity differences are measured at observation points only that are points with high linear intensity gradients. It represents an alternative to traditionally stereo visual odometry algorithms, where the rover's 3D motion is estimated by maximizing the conditional probability of the 3D correspondences between two sets of 3D feature point positions, which were obtained from two consecutive stereo image pairs that were captured by a stereo video camera before and after the rover's motion. Experimental results with synthetic and real image sequences revealed highly accurate and reliable estimates, respectively. Additionally, it seems to be an excellent candidate for mobile robot missions where space, weight and power supply are really very limited.

10.1 Introduction

Over the past two decades, robotic rovers have been extensively used for planetary surface exploration and have demonstrated that unmanned missions are very practical and productive, as well as much cheaper and less risky than manned ones. Up to date, the most successful planetary rovers have been the following six-wheel rocker-bogie rovers developed at NASA Jet Propulsion Laboratory: the Mars Pathfinder mission rover Sojourner [1], the Mars Exploration Rovers (MER) Spirit and Opportunity [2] and the Mars Science Laboratory (MSL) rover Curiosity [3]. In order to increase the maximum exploration range from few tens of kilometers to hundreds of

Geovanni Martinez
Image Processing and Computer Vision Research Laboratory (IPCV-LAB),
Escuela de Ingeniería Eléctrica, Universidad de Costa Rica, 2060 San José, Costa Rica
e-mail: gmartin@eie.ucr.ac.cr

Kilometers, a Mars Airplane is also being investigated [4]. As opposite to the Mars Airplane, flapping insect robots (entomopters) are also being studied due to their potential to fly slow as well as safety land and take off on rocky planets with low atmospheric pressure like Mars [5].

Since communication between Earth and Mars rovers only occurs once or twice a day, and there is a significant delay from the time a command is sent to when the rover receives it, they must be able to autonomously navigate to science targets and to place instruments precisely against these targets, where any navigation error could cause the loss of the entire day of scientific activity.

For reliable and precise autonomous navigation, the rovers need to be able to determine its position and orientation at any time instant. Usually the current rover's position and orientation are estimated by integrating the rover's motion (rover's change of position and orientation) from the time the motion began to the current time, assuming that the initial rover's position and orientation are known or previously estimated. In the MER rovers Spirit and Opportunity the rover's change of orientation (rover's rotation) is estimated from measurements of three-axis angular rate sensors (gyros) provided by an Inertial Measurement Unit (IMU) onboard the rover [6]. The rover's change of position (rover's translation) is estimated from encoder readings of how much the wheels turned (wheel odometry). The initial rover's orientation is estimated from measurements of three-axis accelerometers provided by the IMU, as well as a sun position vector provided by a sun sensor which is also onboard the rover [7]. The initial position is reset by command at the beginning of the rover's motion.

One limitation of the rocker-bogie mobile rovers as observed on Mars is the excessive wheel slippage on steep slopes and soft soils, which causes large errors particularly on the estimated rover's position from wheel odometry. These position errors can even bring the rover to get stuck, digging itself deeper and deeper into a sand hole, from where it is difficult to get the rover back up on solid ground without incurring some significant damage in the process.

To correct any position error due to wheel slippage, the rover's 3D motion is also estimated by using a stereo visual odometry algorithm. It estimates the rover's 3D motion by maximizing the conditional probability of the 3D correspondences between two sets of 3D feature point positions, which were previously obtained from two consecutive stereo image pairs that were captured by a stereo video camera before and after the rover's motion, respectively. This feature based stereo visual odometry algorithm was first proposed by Moravec in [8] and then improved in [9, 10]. Afterwards, it evolved to become more robust [11] until it was finally implemented in real time to be used in the Mars Exploration Rover Mission [12]. Similar feature based stereo visual odometry algorithms have been also presented in [13, 14, 15]. In [14] and [16] feature based visual odometry algorithms are also described using a monocular standard video camera and an omnidirectional video camera, respectively. In [17, 18] feature based monocular visual odometry is extended to a Simultaneous Localization and Mapping (SLAM).

In the stereo visual odometry algorithm used in the Mars Exploration Rover Mission [12], feature points are first selected on the left frame (image) of the first stereo pair. Next, the feature points are also found in the corresponding right frame by correlation based template matching. Then, the 3D positions of the feature points before rover's motion are estimated from the 2D positions of the feature points in both frames of the first stereo pair by stereo triangulation. Next, the feature points are moved using an initial motion estimate provided by the onboard wheel odometry, then they are projected into the left frame of the second stereo pair and their positions refined by correlation based template matching. Then, the feature points are also found in the corresponding right frame by correlation based template matching. Next, the 3D positions of the feature points after rover's motion are estimated from the 2D positions of the feature points in both frames of the second stereo pair by stereo triangulation. Next, the 3D correspondences (3D offsets) between the two sets of 3D feature point positions before and after the rover's motion are established. Finally, those 3D motion parameters which maximize the conditional probability of the established 3D correspondences are considered to be the stereo visual odometry 3D motion estimates. The conditional probability is computed by modeling the 3D position error at each feature point with Gaussian distributions and using a linearized 3D feature point position transformation, which transforms the 3D position of a feature point before motion into its 3D position after motion given the rover's 3D motion parameters.

After evaluating the performance of the above stereo visual odometry algorithm in both MER rovers Spirit and Opporttuniy on Mars, it was further improved in [19] resulting in a more robust and at least four time more computationally efficient algorithm, which can also operate with no initial motion estimate from wheel odometry. This last updated version of the stereo visual odometry algorithm was planed to be used in the MSL rover Curiosity.

In this paper, as an alternative to the traditional feature based stereo visual odometry, an intensity-difference based monocular visual odometry algorithm is described, which will be able to estimate the rover's 3D motion evaluating the intensity differences at different observation points between two intensity frames captured by a monocular video camera before and after the robot's motion, where an observation point is an image point with high linear intensity gradient. This avoids establishing feature points correspondences for rover's 3D motion estimation and the problems associate with it. The rover's 3D motion will be estimated by maximizing the conditional probability of the frame to frame intensity differences at the observation points. The conditional probability is computed by expanding the intensity signal by a Taylor series and neglecting the nonlinear terms, resulting the well known optical flow constraint [20, 21], as well as using a linearized 3D observation point position transformation, which transforms the 3D position of an observation point before motion into its 3D position after motion given the rover's 3D motion parameters. Perspective projection of the observation points into the image plane and zero-mean Gaussian stochastic intensity errors at the observation points are also assumed. Similar approaches have been already implemented successfully in applications such as video compression [22] and teleoperation of space robots [23].

Our approach differs from traditional optical flow approaches such as described in [24, 25] because we do not follow the typical two-stage algorithm, where the optical flow vector field is first estimated and then the rover's 3D motion is estimated from the previously estimated optical flow vector field, instead we follow a similar approach such as described in [26] by developing an one-stage estimation algorithm, which is able to directly deliver the rover's 3D motion parameters without estimating the optical flow vector field, avoiding in this way solving the optical flow as an intermediate step and the associate problems with the flow estimation.

This contribution is organized as follows. In section 10.2, the proposed monocular visual odometry algorithm is presented. In section 10.3, the experimental results are presented. In section 10.4, a brief summary and the conclusions are given. Finally, in section 10.5, a short description of the future work is given.

10.2 Monocular Visual Odometry Algorithm

In this section, an algorithm to estimate the 3D motion of a rover between two arbitrary time instants t_{k-1} and t_k is presented. This is done by maximizing the conditional probability of the intensity differences between two consecutive frames I_{k-1} and I_k, which were captured at the same time instants by a monocular camera rigidly attached to the rover. The intensity differences are measured at selected points called observation points. The conditional probability is a function of the rover's 3D motion, the frame to frame intensity differences and the covariance matrix of the intensity errors at the observation points. To compute this conditional probability, a mathematical relationship between the rover's 3D motion and the frame to frame intensity differences at the observation points is used. This relationship is based on a number of assumptions about the world and how it is projected into the image plane of the camera. This assumptions are reviewed in subsection 10.2.1. Subsection 10.2.2 explains what is an observation point and how the intensity differences are measured at the observation points. In subsection 10.2.3, the conditional probability of the intensity differences at the observation points is computed. In subsection

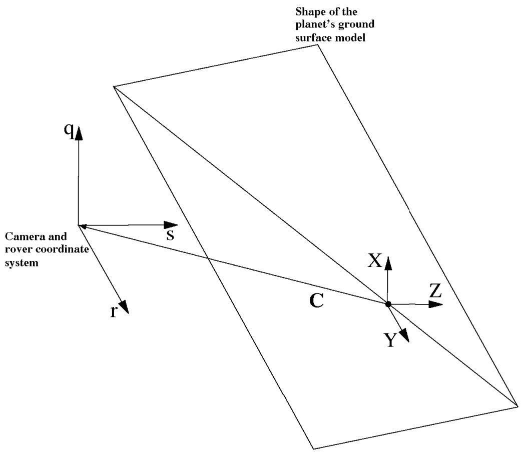

Shape of the planet's ground surface model
q
s
r
Camera and rover coordinate system
C
X
Y
Z

Fig. 10.1 Shape of the planet's ground surface model. Currently, it is described by a planar and rigid mesh of two triangles with coordinate system (X,Y,Z). The camera coordinate system (q,r,s) is supposed to coincide with the robot coordinate system.

10.2.4, the method to maximize the conditional probability for getting the rover's 3D motion estimates is described. Finally, subsection 10.2.5 explains how the algorithm is initialized at the beginning of the image sequence.

10.2.1 Planet's Ground Surface Model

For rover's 3D motion estimation from time t_{k-1} to time t_k, it is assumed that the monocular camera is looking down toward the planet's ground surface, that the planet's ground surface is visible by the camera and that it covers the whole area of the captured intensity images. Furthermore, it is assumed that at time t_{k-1} a model of a visible portion of the planet's ground surface is available or has been previously computed as described in subsection 10.2.5, whose shape with coordinate system (X,Y,Z), texture and relative pose to the camera coordinate system (q,r,s) correspond with those of the real portion of the planet's ground surface at time t_{k-1}. For simplicity, from now on, this model will be referred as planet's ground surface model. The model's shape is represented by a rigid mesh of triangles. Currently, the mesh consists of only two triangles forming a rectangle (see Fig. 10.1). The model's texture is described by the intensity and chrominance values being reflecting from it. The model's pose is given by six parameters: the three components of a 3D position vector and three rotation angles. In addition, the camera coordinate system (q,r,s) is supposed to coincide with the rover coordinate system and an image is supposed to be generated by perspective projection of the planet's ground surface into the image plane of the camera. Moreover, it is assumed that there are no moving objects on the planet's ground surface and that the illumination is diffuse as well as spatial and time invariant. Thus, the frame to frame intensity differences are due to the frame to frame rover's 3D motion only.

The rover's 3D motion from time t_{k-1} to time t_k is described by a rotation followed by a translation of its own coordinate system (q,r,s) respect to the ground surface coordinate system (X,Y,Z). The translation is described by the 3 components of the 3D translation vector $\Delta\mathbf{T} = (\Delta T_X, \Delta T_Y, \Delta T_Z)^\top$. The rotation is described by 3 rotation angles: $\Delta\omega_X, \Delta\omega_Y, \Delta\omega_Z$. These six motion parameters are represented by the vector $B = (\Delta T_X, \Delta T_Y, \Delta T_Z, \Delta\omega_X, \Delta\omega_Y, \Delta\omega_Z)^\top$ and estimated by maximizing the conditional probability of the frame to frame intensity differences measured at the observation points.

10.2.2 Observation Points

For estimating the rover's 3D motion from time t_{k-1} to time t_k, it is also assumed that at time t_{k-1} a set of observation points is available. An observation point lies on one of the triangles of the mesh of the planet's ground surface model at barycentric coordinates $\mathbf{A}_v$ and carries the intensity value I, as well as the linear intensity gradients $g = (g_x, g_y)^\top$ at position A_v. The observation points are created and initialized at the beginning of the image sequence according to the method that is described in subsection 10.2.5. Let $A = (A_q, A_r, A_s)^\top$ be the corresponding position of an observation point with respect to the rover coordinate system at time t_{k-1} and let $a = (a_x, a_y)^\top$ be the position of its perspective projection into the camera plane (see Fig. 10.2). Assuming that the shape, position and orientation of the planet's ground surface model correspond with those of the real portion of the planet's ground surface at time t_{k-1}, the frame to frame intensity difference fd at the observation point a is approximated as follows:

$$fd(\mathbf{a}) = I_k(\mathbf{a}) - I_{k-1}(\mathbf{a}) \approx I_k(\mathbf{a}) - I \tag{10.1}$$

where $I_{k-1}(a)$ and $I_k(a)$ represent the intensity value of the images I_{k-1} and I_k at the position a, respectively. Since in general a lies outside of the image raster, the intensity value $I_k(a)$ is computed by bilinear interpolation of the intensity values of the nearest four pixels of the intensity image I_k. Then, the frame to frame intensity difference at N observation points is represented as follows:

$$FD = (fd(a^{(N-1)}), fd(a^{(N-2)}), \dots, fd(a^{(0)}))^\top \tag{10.2}$$

Moreover, the mean squared frame to frame intensity difference at the observation points is given by:

$$msd = \frac{1}{N} \sum_{n=0}^{N-1} fd(a^{(n)})^2 \tag{10.3}$$

10.2.3 Conditional Probability of the Intensity Differences

Let's consider an arbitrary observation point at barycentric coordinates A_v on the surface of one of the triangles of the planet's ground surface model. Let I and $g = (g_x, g_y)^\top$ be its intensity value and linear intensity gradients, respectively. Due

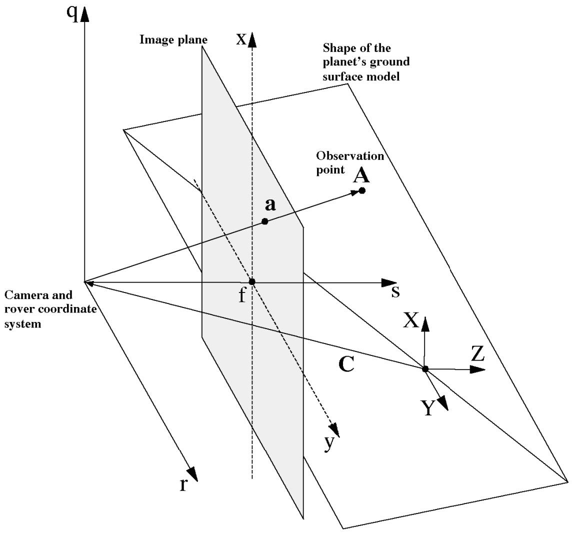

q
Image plane
X
Shape of the planet's ground surface model
Observation point
A
a
f
s
Camera and rover coordinate system
X
C
Z
Y
y
r

Fig. 10.2 Observation point A on the planet's ground surface model with respect to the camera coordinate system and its perspective projection a into the image plane

to the robot's motion from time t_{k-1} to time t_k it moves from A to A' with respect to the camera coordinate system (see Fig. 10.3). The corresponding perspective projections into the image plane are a and a', respectively. Expanding the intensity signal I_{k-1} at image position a by a Taylor series and neglecting the nonlinear terms as proposed in [21], the following relationship between the unknown position a' and the frame to frame intensity difference is obtained:

$$fd(a) = I_k(a) - I_{k-1}(a) \approx -g^\top \left(a' - a\right) \tag{10.4}$$

which is known in the professional literature as Horn and Schunck optical flow constraint equation [20]. Expressing a with their corresponding coordinates at the camera coordinate system using a perspective camera model with known focal distance f results:

$$a = \begin{bmatrix} \frac{f\,A_q}{A_s} \\ \frac{f\,A_r}{A_s} \end{bmatrix} \tag{10.5}$$

Approximating the Eq. 10.5 at position A by using a Taylor series and neglecting the nonlinear terms the following transformation for the unknown position a' can be obtained:

$$a' \approx a + \begin{bmatrix} \frac{f}{A_s} & 0 & \frac{f\,A_q}{{A_s}^2} \\ 0 & \frac{f}{A_s} & \frac{f\,A_r}{{A_s}^2} \end{bmatrix} (A' - A) \tag{10.6}$$

where the known position $A = (A_q, A_r, A_s)^\top$ is related with the unknown position $A' = (A'_q, A'_r, A'_s)^\top$ according to:

$$A' = \Delta\mathbf{R}\,(A + C) - C - \Delta\mathbf{T} \tag{10.7}$$

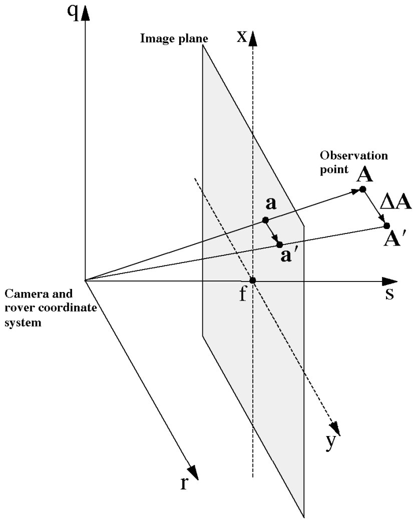


Fig. 10.3 Translation ΔA of an observation point due to the rover's 3D motion with respect to the camera coordinate system and its perspective projection into the camera plane

In the previous equation $C = (C_X, C_Y, C_Z)^\top$ represents the origin of the robot coordinate system respect to the ground surface coordinate system (see Fig. 10.1) and $\Delta\mathbf{R}$ a rotation matrix computed with the rotation angles $-\Delta\omega_X$, $-\Delta\omega_Y$, $-\Delta\omega_Z$. Assuming that the rotation angles $-\Delta\omega_X$, $-\Delta\omega_Y$, $-\Delta\omega_Z$ are small, thus $\cos(-\Delta\omega) \approx 1$ and $\sin(-\Delta\omega) \approx -\Delta\omega$, the Eq. 10.7 can be transformed into:

$$A' \approx \begin{bmatrix} A_q{}' \\ A_r{}' \\ A_s{}' \end{bmatrix} = \begin{bmatrix} 1 & \Delta\omega_Z & -\Delta\omega_Y \\ -\Delta\omega_Z & 1 & \Delta\omega_X \\ \Delta\omega_Y & -\Delta\omega_X & 1 \end{bmatrix} \begin{bmatrix} A_q + C_X \\ A_r + C_Y \\ A_s + C_Z \end{bmatrix} - \begin{bmatrix} C_X - \Delta T_X \\ C_Y - \Delta T_Y \\ C_Z - \Delta T_Z \end{bmatrix} \tag{10.8}$$

Substituing Eq. 10.8 in Eq. 10.6, we obtain:

$$a' \approx \begin{bmatrix} \frac{f\,A_q}{A_s} \\ \frac{f\,A_r}{A_s} \end{bmatrix} + \begin{bmatrix} \frac{f}{A_s} & 0 & \frac{f\,A_q}{A_s{}^2} \\ 0 & \frac{f}{A_s} & \frac{f\,A_r}{A_s{}^2} \end{bmatrix} \begin{bmatrix} \Delta\omega_Z(A_r + C_X) - \Delta\omega_Y(A_s + C_Z) - \Delta T_X \\ -\Delta\omega_Z(A_q + C_X) + \Delta\omega_X(A_s + C_Z) - \Delta T_Y \\ \Delta\omega_Y(A_q + C_X) - \Delta\omega_X(A_r + C_Y) - \Delta T_Z \end{bmatrix} \tag{10.9}$$

Finally, substituting Eqs. 10.9 and 10.5 in Eq. 10.4, the following linear equation that relates the unknown motion parameters and the frame to frame intensity difference measured at the observation point position **a** is obtained:

$$\begin{aligned} fd(a) = {} & \frac{f\,g_x}{A_s}\,\Delta T_X + \frac{f\,g_y}{A_s}\,\Delta T_Y - \frac{f\,(A_q g_x + A_r g_y)}{A_s^2}\,\Delta T_Z + \\ & -\frac{f\,[A_q g_x(A_r + C_Y) + A_r g_y(A_r + C_Y) + A_s g_y(A_s + C_Z)]}{A_z^2}\,\Delta\omega_X + \\ & +\frac{f\,[A_r g_y(A_q + C_X) + A_q g_x(A_q + C_X) + A_s g_x(A_s + C_Z)]}{A_z^2}\,\Delta\omega_Y + \\ & -\frac{f\,[g_x(A_r + C_Y) - g_y(A_q + C_X)]}{A_s}\,\Delta\omega_Z \end{aligned} \tag{10.10}$$

Eq. 10.10 can also be written in vector form as:

$$fd(a) = o^\top\,B + \Delta I \tag{10.11}$$

where

$$o = \begin{bmatrix} \frac{f\,g_x}{A_s} \\ \frac{f\,g_y}{A_s} \\ -\frac{f\,(A_q g_x + A_r g_y)}{A_s^2} \\ -\frac{f\,[A_q g_x(A_r + C_Y) + A_r g_y(A_r + C_Y) + A_s g_y(A_s + C_Z)]}{A_s^2} \\ \frac{f\,[A_r g_y(A_q + C_X) + A_q g_x(A_q + C_X) + A_s g_x(A_s + C_Z)]}{A_s^2} \\ -\frac{f\,[g_x(A_r + C_Y) - g_y(A_q + C_X)]}{A_s} \end{bmatrix} \tag{10.12}$$

and the term ΔI represents the frame to frame intensity error caused by the camera noise, the local or global temporal illumination changes, the shape error of the planet's ground surface model and the accumulated position error due to the motion estimation errors occurred by the motion analysis of previous frames.

Evaluating Eq. 10.11 at $N > 6$ observation points the following system of linear equations is obtained:

$$FD = O\,B + V \tag{10.13}$$

where O is the observation matrix:

$$O = \left[o^{(N-1)^{\top}}, o^{(N-2)^{\top}}, \ldots, o^{(0)^{\top}} \right]^{\top} \tag{10.14}$$

and V is the vector with the N intensity errors:

$$V = \left[\Delta I^{(N-1)}, \Delta I^{(N-2)}, \ldots, \Delta I^{(0)} \right]^{\top} \tag{10.15}$$

The latter can be computed solving for V in Eq. 10.13:

$$V = FD - O\,B \tag{10.16}$$

Modeling the intensity error $\Delta I^{(n)}$ with image coordinates $a^{(n)}$ by a stationary zero-mean Gaussian stochastic process, the joint probability density of the intensity errors at N observation points with image coordinates $a^{(n)}$, $n = 0,..,N-1$, can be computed as:

$$p(V) = \frac{1}{\sqrt{(2\pi)^N |U|}}\, e^{-\frac{1}{2}\left(V^{\top}\, U^{-1}\, V\right)} \tag{10.17}$$

where $|U|$ is the determinant of the covariance matrix U of the intensity errors at the N observation points. Assuming that the variance of the intensity error $\Delta I^{(n)}$ with image coordinates $a^{(n)}$ is 1 and that the intensity errors are statistically independent, this covariance matrix becomes the identity matrix:

$$U = E[V\,V^{\top}] = \begin{bmatrix} 1\,0\,0 \cdots 0 \\ 0\,1\,0 \ldots 0 \\ 0\,0\,1 \cdots 0 \\ \vdots\;\vdots\;\vdots\;\vdots\;\vdots \\ 0\,0\,0\;\;0\;\;1 \end{bmatrix} \tag{10.18}$$

Substituting Eq. 10.16 in Eq. 10.17 the conditional probability of the frame to frame intensity differences at the N observation points can be written as follows:

$$p(FD|B) = \frac{1}{\sqrt{(2\pi)^N |U|}} e^{-\frac{1}{2}\left((FD - O\,B)^{\top} U^{-1} (FD - O\,B)\right)} \tag{10.19}$$

10.2.4 Maximizing the Conditional Probability

The 3D motion parameters $B = (\Delta T_X, \Delta T_Y, \Delta T_Z, \Delta \omega_X, \Delta \omega_Y, \Delta \omega_Z)^\top$, which describe the rover's 3D motion from time t_{k-1} to time t_k, are estimated by maximizing the Eq. 10.19. To this end, the derivative of the natural logarithm of the Eq. 10.19 is first computed and then set to 0 as follows:

$$\frac{\partial \ln p(FD|B)}{\partial B} = \frac{\partial((FD - O\,B)^\top U^{-1}(FD - O\,B))}{\partial B} = 0 \tag{10.20}$$

Finally, the Maximum-Likelihood motion estimates are obtained by solving for B in the above equation:

$$\widehat{B} = \left(O^\top U^{-1} O\right)^{-1} O^\top U^{-1} FD \tag{10.21}$$

Due to the truncation errors caused by neglecting the nonlinear terms in all approximations done to obtain Eq. 10.10, the algorithm needs to be applied iteratively to improve its reliability and accuracy [22, 23]. After each iteration i, the resulting estimates ${}^i\widehat{B}$ are used to compensate the motion of the planet's ground surface model, as well as to update the motion estimates $\widehat{B}$ found by previous iterations. Due to the motion compensation, the frame to frame intensity differences at the observation points decreases. The iteration ends when after two consecutive iterations the mean square frame to frame intensity difference at the observation does not decrease significantly. In each iteration i the following steps are carried out:

1. Evaluate Eq. 10.10 at each observation point
2. Compute the intensity differences ${}^iFD$ and observation matrix ${}^iO$ according to Eq. 10.2 and Eq. 10.14, respectively
3. Obtain the motion estimates ${}^i\widehat{B}$ using Eq. 10.21
4. Compensate the motion of the mesh of triangles by moving its vertices according to Eq. 10.7 with the estimates ${}^i\widehat{B}$
5. Compute the mean squared intensity difference ${}^imsd$ using Eq. 10.3
6. Update the rotation matrix: $\widehat{\Delta\mathbf{R}} \leftarrow {}^i\widehat{\Delta\mathbf{R}}\,\widehat{\Delta\mathbf{R}}$
7. Update the translation vector: $\widehat{\Delta T} \leftarrow \widehat{\Delta T} + {}^i\widehat{\Delta T}$
8. If $|{}^imsd - {}^{i-1}msd| \geq \delta_2$ goto step 1

10.2.5 Planet's Ground Surface Model Initialization

As explained in subsections 10.2.1 and 10.2.2, to estimate the rover's 3D motion from time t_{k-1} to time t_k, the shape, the pose and the observation points of a visible portion of the planet's ground surface need to be known at time t_{k-1}. Here, they are computed by compensating the motion of a planet's ground surface model with the accumulated motion estimates from time t_0 to time t_{k-1}. The planet's ground surface model is created and initialized at the beginning of the image sequence. The motion compensation is gradually performed in the fourth step of each iteration of the motion estimation algorithm described in the previous subsection.

The initialization of the shape, the pose and the texture of the model along with the observation points is carried out only once at the beginning of the image sequence, one after the other, as follows:

1. *Shape initialization*: The model's shape is currently initialized as a flat and rigid mesh of two triangles forming a rectangle. Thus, at present, it is only possible to model the shape of a flat portion of the visible planet‘s ground surface. Here, the image area of that portion is at least 20% of the total image area.
2. *Pose initialization*: Currently, the orientation of the model's shape is initialized manually so that it fairly corresponds to that of the surface portion being modeled. The position is also set manually so that the perspective projection of the mesh into the image plane covers the corresponding image region of the surface portion being modeled. The focal lens f is assume to be 1. In this uncalibrated case, the robot translation can only be estimated up to scale factor.
3. *Texture initialization*: The model's texture is initialized by projecting the intensity and chrominance values of the first image of the sequence onto its mesh of triangles by using texture mapping.
4. *Observation points initialization*: The observation points are created and initialized as follows. First, the gradient images G_{0x} and G_{0y} are computed by convolving the first intensity image I_0 with the Sobel operator. Then, the 3D vertex positions of all visible triangles of the ground surface model are perspectively projected into the camera plane. In order to reduce the influence of the camera noise and to increase the accuracy of the estimation, an arbitrary image point $\mathbf{a}$ inside the image region of a projected triangle will be selected as an observation point only if the linear intensity gradient at position $\mathbf{a}$ satisfies $|\mathbf{G}_0(\mathbf{a})| > \delta_1$. Next, the 3D position vector $\mathbf{A}$ with respect to the camera coordinate system of each selected observation point is computed as the intersection of the $\mathbf{a}$'s line of sight and the plane containing the corresponding triangle's vertex 3D positions. Then, the corresponding barycentric coordinates $\mathbf{A}_v$ with respect to the vertex 3D positions are also computed. Finally, each selected observation point is rigidly attached to the triangle's surface. Its position, intensity value I and linear intensity gradient $\mathbf{g} = (g_x, g_y)^\top$ are set to $\mathbf{A}_v$, $I_0(\mathbf{a})$ and $(G_{0x}(\mathbf{a}), G_{0y}(\mathbf{a}))^\top$, respectively. Since it is assumed that the illumination is diffuse as well as spatial and time invariant, the intensity value I and the linear intensity gradient $\mathbf{g} = (g_x, g_y)^\top$ of at observation point at position $\mathbf{A}_v$ remain constant during the image sequence.

10.3 Experimental Results

In order to evaluate the estimation accuracy of the proposed monocular visual odometry algorithm, we applied it to each image pair of a set of 1000 synthetically generated image pairs with the following image dimensions: 688 pixel $\times$544 pixel. Each image pair consists of two images captured by a rover at time t_{k-1} and time t_k, where the camera is rigidly attached to the rover and its coordinate system coincides with the rover's coordinate system. For each image pair, the shape, the texture and the observation points of the ground surface, as well as its position and orientation at time t_{k-1} with respect to the camera, including the rotation and translation parameters which describes the rovert's 3D motion from time t_{k-1} to time t_k are exactly known. The rover's rotation and the rover's translation from time t_{k-1} to time t_k are different in all image pairs and their parameters are uniformly distributed in interval (-10.0 to +10.0) m for the translation and in interval (-5.0 to +5.0)° for the rotation. The experiment was performed on a iMac with an Intel Core i5 at 3.1 GHz and 12.0 GB RAM.

At a camera noise variance of 10, the Table 10.1 depicts the absolute estimation error of the translation parameter along the X axis $|E_{\Delta T_X}|$, along the Y axis $|E_{\Delta T_Y}|$ and along the Z axis$|E_{\Delta T_Z}|$. This gives an average of 0.00432 m and 0.00111 m for the mean and the standard deviation of the absolute estimation error of the translation parameters, respectively. Taking into account that the rover's translation along each axis was uniformly distributed in interval (-10.0 to +10.0) m, an average of the absolute estimation error of the translation parameters of approximately 0.005 m is an excellent indicative of the high accuracy achieved by the proposed algorithm in the estimation of the rover's translation parameters.

At a camera noise of 10, Table 10.2 depicts the absolute estimation error of the rotation parameter around the X axis $|E_{\Delta \omega_X}|$, around the Y axis $|E_{\Delta \omega_Y}|$ and around the Z axis $|E_{\Delta \omega_Z}|$. This gives an average of 0.00337° and 0.00198° for the mean and the standard deviation of the absolute estimation error of the rotation parameters, respectively. Taking into account that the rover's rotation around each axis was uniformly distributed in interval (-5.0 to +5.0)°, an average of the absolute estimation error of the rotation parameters of approximately 0.004° is also an excellent indica-

Table 10.1 Absolute estimation error of the translation parameter along the X axis $|E_{\Delta T_X}|$, along the Y axis $|E_{\Delta T_Y}|$ and along the Z axis $|E_{\Delta T_Z}|$ at a camera noise variance of 10

Absolute estimation error	mean (m)	standard deviation (m)
$\lvert E_{\Delta T_X}\rvert$	0.00203	0.00124
$\lvert E_{\Delta T_Y}\rvert$	0.00065	0.00033
$\lvert E_{\Delta T_Z}\rvert$	0.01027	0.00177
Average	0.00432	0.00111

Table 10.2 Absolute estimation error of the rotation parameter around the X axis $|E_{\Delta \omega_X}|$, around the Y axis $|E_{\Delta \omega_X}|$ and around the Z axis $|E_{\Delta \omega_X}|$ at a camera noise variance of 10

Absolute estimation error	mean (°)	standard deviation (°)
$\lvert E_{\Delta \omega_X}\rvert$	0.00537	0.00316
$\lvert E_{\Delta \omega_Y}\rvert$	0.00324	0.00234
$\lvert E_{\Delta \omega_Z}\rvert$	0.00150	0.00042
Average	0.00337	0.00198

tive of the high accuracy achieved by the proposed algorithm in the estimation of the rover's rotation parameters.

In a second experiment, we applied the proposed visual odometry algorithm to 6 different real image sequences with image size of 256 pixel $\times$ 256 pixel. They were taken by the left navigation camera of the MER rover Opportunity at different dark chocolate brown flat desolate martian landscapes. All original images are courtesy of NASA/JPL-Caltech. Because in this experiment the shape, the texture and the observation points of the planet's ground surface model as well as its relative pose with respect to the camera are not known at the beginning of the image sequence, they are first initialized as described in subsection 10.2.5. The experimental results revealed an average processing time of 0.1 sec/frame, as well as that the tracking was never lost. We have also generated an image sequence where a virtual planar object (triangle) has been integrated into the original image sequence and animated in a way that it looks like it is rigidly attached to the ground surface using the

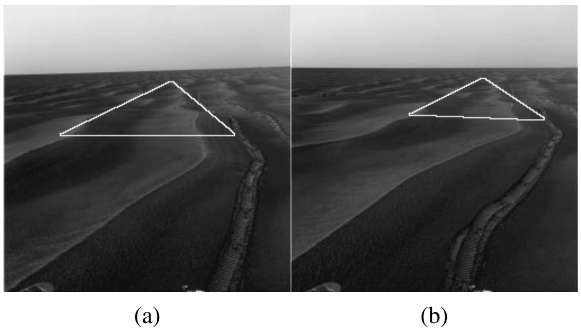


Fig. 10.4 aaPerspective projection of a triangle into (a) the first and (b) the last image of the test sequence #2 captured by the rover Opportunity on Mars. Since the triangle is motion compensated with the negative of the accumulated rover's 3D motion estimates, it seems to be glued to the ground surface. All original images are courtesy NASA/JPL-Caltech.

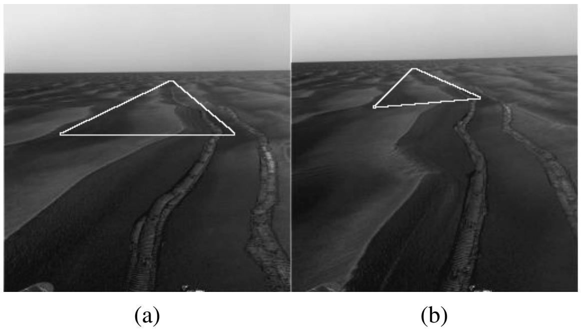


Fig. 10.5 Perspective projection of a triangle into (a) the first and (b) the last image of the test sequence #3 captured by the rover Opportunity on Mars. Since the triangle is motion compensated with the negative of the accumulated rover's 3D motion estimates, it seems to be glued to the ground surface. All original images are courtesy NASA/JPL-Caltech.

estimated frame to frame 3D motion parameters of the rover. Actually this is achieved by compensating the motion of the triangle with the negative of the accumulated rover's 3D motion estimates and then projecting the triangle into the image plane for each image of the sequence. A subjective analysis of the results revealed that the triangle really seems to be glued to the ground surface, which is also an indicative of the excellent reliability of the proposed algorithm. Figs. 10.4.(a)-10.7.(a) and Figs. 10.4.(b)-10.7.(b) depict the animated triangle projected into the first image and into the last image of the test image sequence #2, #3, #4, and #6, respectively. We have also achieved similar results by applying the proposed visual odometry algorithm to both the 400 frames of a real aerial image sequence and the 400 frames of a real infrared aerial image sequence, the two with image size of 688 pixel $\times$ 544 pixel.

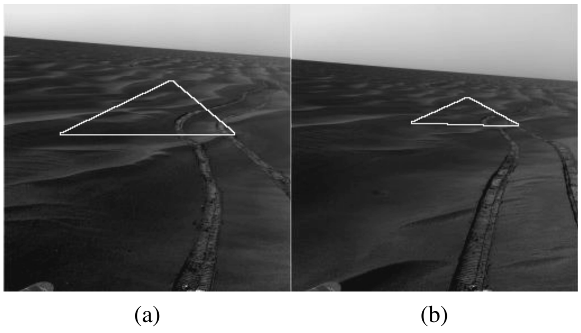


Fig. 10.6 Perspective projection of a triangle into (a) the first and (b) the last image of the test sequence #4 captured by the rover Opportunity on Mars. Since the triangle is motion compensated with the negative of the accumulated rover's 3D motion estimates, it seems to be glued to the ground surface. All original images are courtesy NASA/JPL-Caltech.

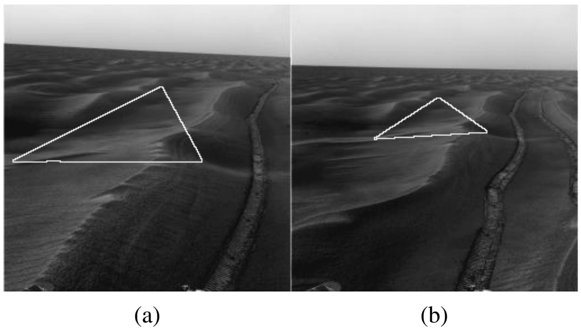


Fig. 10.7 Perspective projection of a triangle into (a) the first and (b) the last image of the test sequence #6 captured by the rover Opportunity on Mars. Since the triangle is motion compensated with the negative of the accumulated rover's 3D motion estimates, it seems to be glued to the ground surface. All original images are courtesy NASA/JPL-Caltech.

10.4 Summary and Conclusions

In this contribution, we have showed that the rover's 3D motion can be also precisely estimated from measurements of frame to frame intensity differences provided by a monocular video camera. Although a validation of its performance in a real rover test bed is still missing, the experimental results so far revealed that it is promising candidate to be used to improve long-range autonomous navigation of rovers on a planetary surface. This because it could help when wheel odometry and traditionally stereo visual odometry have failed, as well as it could be used to validate the stereo visual odometry estimate or to generate a better estimate by statistically combining the wheel odometry estimate, the stereo visual odometry estimate and the estimate of the proposed algorithm using sensor fusion techniques. It is also an excellent candidate for lighter rover systems or entomopters where space, weight and power supply are really very limited. We even believe that the proposed algorithm could have a similar error growth than that achieved with the stereo visual odometry algorithm used by the MER rovers and the MSL rover and it could be more effective when the distance to the scene is much larger than the stereo baseline. Additionally, it has the advantage of being able to operate just with a single monocular video camera, which consumes less energy, weight less and needs less space than a stereo video camera. We are also convinced that the proposed algorithm could be computationally more efficient than the stereo visual odometry because it does not depend at all on any correlation based template matching for operation.

Our intention is not to replace the stereo visual odometry but to show that monocular visual odometry based on frame to frame intensity differences is another reliable and precise way for odometry estimation that can be merged with other sensors to improve the long rage autonomous navigation of the current and future rovers, airplanes and flapping insect robots for planetary exploration.

10.5 Future Work

In the future work, the proposed algorithm will be implemented and tested in a real rover platform Clearpath Robotics Husky A200 (see Fig.10.8), as well as a set of experiments will be performed in a simulated Martian landscape to assess its reliability, robustness, error growth, power consumption and overall size. The obtained results will be compared with those achieved by the stereo visual odometry and existing monocular SLAM algorithms.

Currently, we are working on the development of the rover's real time image acquisition system consisting of three IEEE-1394 cameras installed onto the rover's mast, each looking down toward the ground surface in three different directions: to the front, rear and left side parts of the rover, respectively (see Figs. 10.8 and 10.9). During the experiments only one of those three cameras will be used for monocular visual odometry. The system is being developed under Ubuntu 12.04.2 LTS, ROS Fuerte and the programing language C. The image acquisition system will also correct, in real time, the radial and tangential distortions due to the camera lens.

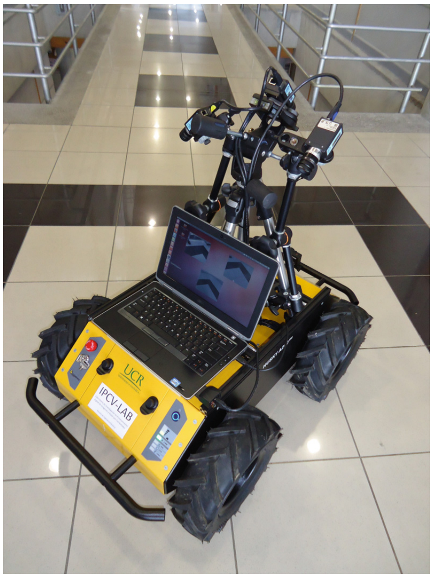


Fig. 10.8 Real rover test bed Clearpath Robotics Husky A200, which is currently used to validate the proposed monocular visual odometry algorithm. The rover's real time image acquisition system consists of three IEEE-1394 cameras installed onto the rover's mast, each looking down toward the ground surface in three different directions: to the front, rear and left side parts of the rover, respectively. During the experiments only one of those three cameras will be used for monocular visual odometry.

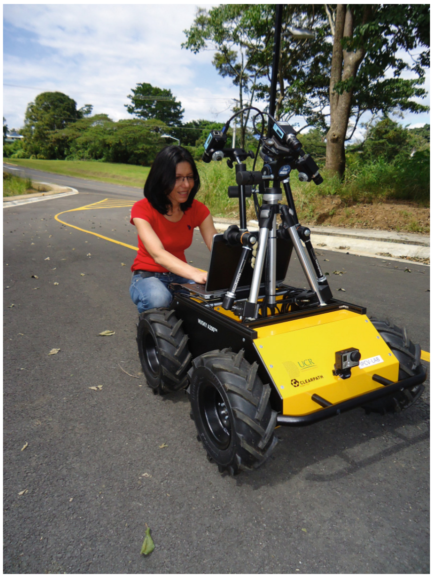


Fig. 10.9 First tests of the real rover test bed Husky A200 and its image acquisition system on campus of the University of Costa Rica

References

1. Shirley, D., Matijevic, J.: Mars Pathfinder Microrover. Autonomous Robots 2, 283–289 (1995)
2. NASA, Mars Exploration Rover Launches. In: Press Kits. Jet Propulsion Laboratory, California Institute of Technology (2003)
3. NASA, Mars Science Laboratory Launch. In: Press Kits. Jet Propulsion Laboratory, California Institute of Technology (2011)
4. Braum, R.: Design of the ARES Mars Airplane and Mission Architecture. Journal of Spacecraft and Rockets 43(5), 1026–1034 (2006)
5. Michelson, R.C.: Slow flight in the lower Mars Atmosphere in support of NASA science missions. In: Proceedings of International Unmanned Vehicles Workshop (2010)
6. Ali, K., Vanelli, C., Biesiadecki, J., et al.: Attitude and Position Estimation on the Mars Exploration Rovers. In: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (2005)
7. Eisenman, A., Liebe, C., Perez, R.: Sun sensing on the Mars exploration rovers. In: Proceedings of IEEE Aerospace Conference (2002)
8. Moravec, H.: Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. PhD Thesis. Stanford University (1980)
9. Matthies, L., Shafer, S.: Error Modeling in Stereo Navigation. IEEE Journal of Robotics and Automation Ra-3(3), 239–248 (1987)
10. Matthies, L.: Dynamic stereo vision. PhD Thesis. Computer Science, Carnegie Mellon (1989)
11. Olson, C., Matthies, L., Schoppers, M., et al.: Robust stereo ego-motion for long distance navigation. In: Proceedings of IEEE Conference on Computer Vision Pattern Recognition (2000)
12. Maimone, M., Cheng, Y., Matthies, L.: Two years of visual odometry on the Mars Exploration Rovers. Journal of Field Robotics 24(3), 169–186 (2007)
13. Lacroix, S., Mallet, A., Chatila, R., et al.: Rover self localization in planetary-like environments. In: Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space (1999)
14. Nister, D., Naroditsky, O., Bergen, J.: Visual Odometry. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2004)
15. Comport, A., Malis, E., Rives, P.: Accurate Quadrifocal Tracking for Robust 3D Visual Odometry. In: Proceedings of IEEE International Conference on Robotics and Automation (2007)
16. Corke, P., Strelow, D., Singh, S.: Omnidirectional Visual Odometry for a Planetary Rover. In: Proceedings of IEEE International Conference on Robots and Systems (2005)
17. Strasdat, H., Montiel, J., Davison, A.: Real time monocular SLAM: Why filter? In: Proceedings of IEEE International Conference on Robotics and Automation (2010)
18. Strasdat, H., Montiel, J., Davison, A.: Scale Drift-Aware Large Scale Monocular SLAM. In: Proceedings of Robotics: Science and Systems (2010)
19. Johnson, A., Goldberg, S., Cheng, Y., et al.: Robust and Efficient Stereo Feature Tracking for Visual Odometry. In: Proceedings of IEEE International Conference on Robotics and Automation (2008)
20. Horn, B., Schunck, B.: Determining Optical Flow. Artificial Intelligence 17, 185–203 (1981)
21. Lucas, B., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence (1981)

22. Martinez, G.: Analyse-Synthese-Codierung basierend auf dem Modell bewegter dreidimensionaler, gegliederter Objekte. PhD Thesis. Leibniz Universitaet Hannover, Germany (1998)
23. Martinez, G., Kakadiaris, I., Magruder, D.: Teleoperating ROBONAUT: A case study. In: Proceedings of British Machine Vision Conference (2002)
24. Adiv, G.: Determining Three-Dimensional Motion and Structure from Optical Flow Generated by Several Moving Objects. IEEE Transactions on Pattern Analysis and Machine Intelligence 7(4), 384–401 (1985)
25. Heeger, D., Jepson, A.D.: Subspace methods for recovering rigid motion I: Algorithm and Implementation. International Journal of Computer Vision 7(2), 95–117 (1992)
26. Horn, B., Weldon, E.J.: Direct methods for recovering motion. International Journal of Computer Vision 2, 51–76 (1988)

Author Index