The Dissertation Committee for Changhai Xu

certifies that this is the approved version of the following dissertation:

Steps Towards the Object Semantic Hierarchy

Committee:

Benjamin Kuipers, Supervisor

Kristen Grauman, Supervisor

Raymond Mooney

Peter Stone

Silvio Savarese

Steps Towards the Object Semantic Hierarchy

by

Changhai Xu, B.E.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2011

To everyone who has helped and supported me over the years

Acknowledgments

First and foremost, I would like to thank my advisor, Ben Kuipers, for his guidance in my research work. I greatly appreciate his stimulation, optimism, encouragement, and patience in my work, without which I would not have been able to make as much progress as I have achieved. His enthusiasm for research always brings brilliant new ideas to move our projects forward, and at the same time, he gives me a lot of freedom to explore the topics that I am most interested in. I am also very grateful to Ben for helping me improve my English speaking and writing skills. Working with Ben has been fun and I have learned a lot from him.

I would also like to thank my other committee members: Kristen Grauman, for agreeing to be my co-advisor, providing valuable comments for my papers, and teaching me computer vision techniques; Ray Mooney, for introducing me to basic machine learning concepts, and teaching me various machine learning techniques; Peter Stone, for teaching me a wide range of robotic techniques, and helping me quickly get into the implementation of a real robot system; Silvio Savarese, for providing valuable feedback for my papers, and encouraging me to communicate with his students.

Thanks are due to my labmates from the past and present: Aniket Murarka, Jonathan Mugan, Shilpa Gulati, Jeremy Stober, Lewis Fishgold, Joseph Modayil, Patrick Beeson, and Subramanian Ramamoorthy, for their helpful proofreading and discussions for my work. My thanks also go to Jingen Liu, Yong Jae Lee, Grace Tsai, Chia-Wei Luo, Do-ran Chakraborty, Yingze Bao, Yangming Li, and Bangxin Hu, for providing various help in

my work. I would also like to thank Lydia Griffith and Katherine Utz for their hard work to help students live a happy life in the department.

I would like to thank Shengyuan Yang, Jiawen Li, Liming Luo, Yiqing Wei, Huiya Liu, Yi Lin, Junwei Wei, Fan Zhang, Fang Tang, Peng Dong, Jing Zan, Wenyuan Li, Qintao Guo, Xiaohui Gao, Jun Wen, Lan Tang, Ruifang Ge, Wanwan Ren, Yuqiang Guan, Zeyun Yu, and Mansoor Jafry, who have provided me so much fun times at UT Austin. Thanks are also due to my friends Fei Wang, Xingtao Shen, Zheng Xie, Jiangong Li, Yongzhao Bian, and my cousin Yujiang Li, for their support in various ways. Specially, I want to thank Huaming Li for helpful discussions and constant encouragement and support.

Finally, thanks are due to my family for their understanding, support, and patience throughout these years, which have been a constant source of courage to keep me going.

CHANGHAI XU

The University of Texas at Austin
August 2011

Steps Towards the Object Semantic Hierarchy

Publication No. _____

Changhai Xu, Ph.D.
The University of Texas at Austin, 2011

Supervisors: Benjamin Kuipers, Kristen Grauman

An intelligent robot must be able to perceive and reason robustly about its world in terms of objects, among other foundational concepts. The robot can draw on rich data for object perception from continuous sensory input, in contrast to the usual formulation that focuses on objects in isolated still images. Additionally, the robot needs multiple object representations to deal with different tasks and/or different classes of objects. We propose the Object Semantic Hierarchy (OSH), which consists of multiple representations with different ontologies. The OSH factors the problems of object perception so that intermediate states of knowledge about an object have natural representations, with relatively easy transitions from less structured to more structured representations. Each layer in the hierarchy builds an explanation of the sensory input stream, in terms of a stochastic model consist-

ing of a deterministic model and an unexplained "noise" term. Each layer is constructed by identifying new invariants from the previous layer. In the final model, the scene is explained in terms of constant background and object models, and low-dimensional dynamic poses of the observer and objects.

The OSH contains two types of layers: the Object Layers and the Model Layers. The Object Layers describe how the static background and each foreground object are individuated, and the Model Layers describe how the model for the static background or each foreground object evolves from less structured to more structured representations. Each object or background model contains the following layers: (1) 2D object in 2D space (2D2D): a set of constant 2D object views, and the time-variant 2D object poses, (2) 2D object in 3D space (2D3D): a collection of constant 2D components, with their individual time-variant 3D poses, and (3) 3D object in 3D space (3D3D): the same collection of constant 2D components but with invariant relations among their 3D poses, and the time-variant 3D pose of the object as a whole.

In building 2D2D object models, a fundamental problem is to segment out foreground objects in the pixel-level sensory input from the background environment, where motion information is an important cue to perform the segmentation. Traditional approaches for moving object segmentation usually appeal to motion analysis on pure image information without exploiting the robot's motor signals. We observe, however, that the background motion (from the robot's egocentric view) has stronger correlation to the robot's motor signals than the motion of foreground objects. Based on this observation, we propose a novel approach to segmenting moving objects by learning homography and fundamental matrices from motor signals.

In building 2D3D and 3D3D object models, estimating camera motion parameters plays a key role. We propose a novel method for camera motion estimation that takes advantage of both planar features and point features and fuses constraints from both homography and essential matrices in a single probabilistic framework. Using planar features greatly

improves estimation accuracy over using point features only, and with the help of point features, the solution ambiguity from a planar feature is resolved. Compared to the two classic approaches that apply the constraint of either homography or essential matrix, the proposed method gives more accurate estimation results and avoids the drawbacks of the two approaches.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

An intelligent agent, embedded in the physical world, will receive a high-dimensional ongoing stream of sensory input. In order to understand and interact with the world, the agent must be capable of learning high-level concepts. Inspired by human developmental learning, we believe that foundational concepts such as *Space* and *Object* are essential for such a learning agent to abstract and control the complexity of its world. To bridge the gap between continuous interaction with the physical environment, and discrete symbolic descriptions that support effective planning, the agent will need multiple representations for these foundational concepts.

In these foundational concepts, there are often several quite different ways to represent entities of interest, drawing on different *ontologies*, that is, classes of logical concepts and relations. A *semantic hierarchy* is a collection of these different ontologies, arranged so that knowledge of the environment can be acquired in relatively small steps. The Spatial Semantic Hierarchy [Kuipers, 2000; Kuipers et al., 2004] is one such semantic hierarchy, organized to represent knowledge of large-scale and small-scale space.

In this work, we propose the *Object Semantic Hierarchy (OSH)* [Xu and Kuipers,

2010, 2009], which is a collection of representations for objects and their surrounding context, motivated by the work of the Spatial Semantic Hierarchy [Kuipers, 2000; Kuipers et al., 2004]. We assume that the observer is an agent, embedded in a continuous world, with high-bandwidth sensory input and the ability to move within its environment.

The OSH is a hierarchical representation of objects and the static background. Each layer in the hierarchy builds an explanation of the sensory input stream, in terms of a stochastic model consisting of a deterministic model and an unexplained "noise" term. Each layer is constructed by identifying new invariants within previous layer's noise term. In the final model, the scene is explained in terms of constant background and object models, and the low-dimensional time-variant pose trajectories of the observer and objects.

The multi-level representations in the OSH give the agent more robustness in coping with a complex environment, compared with any individual representation. The OSH enables the agent to choose different levels of models to work with, based on different goals. For example, while a 2D object model is usually sufficient for locating a box in input images, it's very likely that the agent needs a 3D object model in order to grasp the box properly. The OSH also provides the agent the ability to learn different descriptions of an object based on different situations. For example, when a car is far from the agent in its visual field, the agent can build a "blob" model to describe it, and when it comes close, the agent may be able to build a richer 3D model while still maintaining the blob model. Moreover, the agent's visual field may include multiple objects that are affected by the agent's own actions. Different objects in the scene will receive different amounts of attention, resulting in descriptions at different levels of detail. The multiplicity of representation in the OSH will help the agent accomplish this task.

1.2 Approach Overview

1.2.1 The OSH Framework

The OSH contains two types of layers: the object layers and the model layers. The object layers describe how the static background and each foreground object are individuated, and the model layers describe how the model for the static background and for each foreground object evolves from less structured to more structured representations.

In the object layers, initially everything in the sensory input is treated as noise; then the agent constructs a constant model of the static background world, where foreground objects are treated as remaining noise; then the foreground objects are progressively individuated from the background and their models are constructed while they are tracked over time.

In the model layers, each of the static background and the foreground objects is represented as

(a) **2D object in 2D space (2D2D)**: a set of constant 2D object views, and the time-variant 2D object poses;

(b) **2D object in 3D space (2D3D)**: a collection of constant 2D components, with their individual time-variant 3D poses;

(c) **3D object in 3D space (3D3D)**: the same collection of constant 2D components but with invariant relations among their 3D poses, and the time-variant 3D pose of the object as a whole.

The idea of the OSH is that early stages of analysis can robustly derive certain properties of the visual scene, that are then used as assumptions to make later processing layers simpler and more robust. When later layers cannot be constructed at a certain time due to limited resources or complicated situations, the earlier layers still allow objects to be tracked in the image, until they are more accessible to the more sophisticated kinds of analysis.

The lower layers in the OSH are easier to construct; the higher layers are more robust to noise, have stronger invariants, and factor the uncertainty in the system more effectively. In the end, the uncertainty in the sensor stream is factored into a collection of relatively simple models: the static background, the dynamic observer's pose, constant object models, dynamic object poses, and any remaining noise. The "blooming, buzzing confusion" of the initial pixel-level input is concisely explained in terms of a relatively small number of object-level concepts and relations.

The OSH approach to object representation exploits both machine vision methods such as feature extraction and object tracking, and machine learning methods such as clustering, regression, and Bayesian inference, to build its hierarchy of models.

1.2.2 Foreground Object Segmentation

To build the 2D2D model for an object, the robot needs to separate out the object in the pixel-level sensor stream from the static background environment, where motion information is an important cue to perform the separation. Based on the observation that the motion of the background (from the robot's egocentric view) has stronger correlation to the robot's motor signals than the motion of foreground objects, we propose a method to detect moving objects by clustering image features according to their motion consistency with motor signals [Xu et al., 2011].

To exploit motor signals is motivated by the human visual system. The human visual system does not rely only upon information from the retina to perceive object motion, because identical retinal stimulations can be evoked by the movement of objects as well as by self-evoked eye movements [Galletti and Fattori, 2003] or head/body movements. The signals for eye, head, and/or body movements for humans correspond to motor signals for a robot. The motor signals allow the robot to predict the motion patterns of background features. In contrast, the motion patterns of foreground features will be different from the predictions because they have independent motions from the robot. This difference provides

us a way to cluster image features based on their discrepancy with their predictions.

Motor signal change of a robot leads to visual change in its input images. As is well known, the visual change is constrained by a homography matrix in the case where the robot's camera has only rotation but no or small translation (for example, a pan tilt camera), and by a fundamental matrix when the camera has large translation (for example, a mobile robot) [Hartley and Zisserman, 2003]. In our system, homography and fundamental matrices are learned off-line as functions of the robot's motor signal changes. These homography and fundamental matrices are then used on-line to predict the feature locations based on motor signal changes. The errors between the predicted feature locations and their actual tracked locations are calculated. The features are clustered into background/foreground using Expectation-Maximization on these errors. Labeled features are then used for pixel-level image segmentation with Active Contours [Isard and Blake, 1998; Kass et al., 1988] and Graph-based Transduction techniques [Joachims, 2003; Shi and Malik, 2000].

Unlike pixel-level background subtraction methods, the proposed approach does not require a large number of frames for background model construction, and does not suffer from accumulated image registration error for dynamic cameras. In contrast to existing sparse feature based foreground/background separation methods, our approach clusters features in only one dimensional space instead of a higher dimensional space, and there is no need to search for parameters in an affine or homography transformation space or motion trajectory space.

1.2.3 Recovery of Camera Motion and Object Structure

The recovery of camera motion and object structure is critical for building the 2D3D and 3D3D object models in the OSH. Two classic approaches have been widely used for camera motion estimation from two views of the same 3D scene: the homography matrix based approach and the essential matrix based approach.

The homography matrix based approach [Sturm, 2000; Zhang, 2000; Ma, 2004;

Hartley and Zisserman, 2003; Cobzas et al., 2009; Molton et al., 2004] works for a planar environment. Two views of a planar surface are related by a homography matrix. The camera motion parameters as well as the plane normal can be obtained from decomposing the homography matrix [Hartley and Zisserman, 2003; Ma, 2004]. This approach may give two physically possible solutions, and in practice it can be very difficult to select the correct solution. The essential matrix based approach [Pollefeys et al., 2004; Nistér, 2004; Zhang, 1998; Longuet-Higgins, 1981; Hartley and Zisserman, 2003; Ma, 2004] works for a more general environment. For two sets of calibrated corresponding points in two images where not all points lie on the same planar surface, they can be related by an essential matrix (in the uncalibrated case, it is called the fundamental matrix). By decomposing the essential matrix, we can get the camera motion parameters [Hartley and Zisserman, 2003; Ma, 2004]. This approach usually needs a large number of point features and RANSAC fitting to get robust parameter estimation. Based on our observation, both of these approaches are sensitive to noise, especially when the camera motion is small.

We propose a novel method that takes advantage of both planar features and point features and fuses constraints from both the homography matrix and the essential matrix in a single probabilistic framework. A single planar region feature is tracked over time, and its normal is estimated based on its tracked location and the information of a set of tracked point features. In the probabilistic normal estimation framework, the constraints from the homography matrix and the essential matrix are formulated together in the likelihood function to improve estimation accuracy. Then the camera motion is obtained based on the estimated normal of the planar region feature. Compared to the two classic approaches based on either the homography matrix or the essential matrix, our method gives more accurate estimation results and avoids the drawbacks of the HMB and EMB approaches.

Other existing approaches for camera motion estimation and object structure recovery include Bundle Adjustment (BA) [Snavely et al., 2008; Klein and Murray, 2007; Sibley et al., 2009; Triggs et al., 2000; Engels et al., 2006] and Visual SLAM [Davison et al.,

2007; Newcombe and Davison, 2010; Eade and Drummond, 2006; Nister et al., 2006]. The dimension of the parameters to be estimated in these approaches is linear in the number of image features. When the number of image features is large, the size of the search space for the parameters will be extremely high. In contrast, our method maintains a probability distribution over only two dimensional normal parameters for a single tracked planar region feature, and the other motion and structure parameters are calculated based on the two estimated normal parameters. Low dimensional parameter estimation allows robust and fast convergence to the solution.

After the camera pose is estimated, we are able to get the 3D models for a tracked object. The 3D object structure is recovered from the camera motion as a collection of triangles in 3D space, where the triangles are obtained from tracked point features. The normal of each triangle is modeled as a Gaussian distribution, and based on the triangles' local geometric continuity, the final model is constructed as a compact set of surfaces. The number of surfaces and their boundaries are automatically identified by maximum a posteriori estimation

1.3 Scope of this Thesis

The OSH is a very general and large framework for object representation and various approaches can be used in building its hierarchy. This thesis targets at:

(a) formalizing the framework of the OSH (2D and 3D models for foreground objects and the static background), and

(b) providing solutions to two key problems: 2D object segmentation and 3D pose estimation, where 2D object segmentation is a fundamental step for separation of a foreground object and the static background in the image stream, and 3D pose estimation is a basic step for 3D model construction.

1.4 Contributions

The primary contributions of the work are:

(i) We propose the Object Semantic Hierarchy (OSH) to represent objects and its surrounding background. The OSH includes both view-based 2D holistic models and part-based 3D structural models. The models in the OSH evolve from one layer to the next in a progressive manner, and in the end the input sensor stream is explained in terms of constant background and object models, and low-dimensional dynamic poses of the observer and objects.

(ii) We propose and develop a novel 2D object segmentation method (MSMS) to separate out moving objects from the static background. As far as we know, this is the first work to use motor signals for object segmentation, in contrast to traditional approaches which appeal to information from images/videos only.

(iii) We evaluate the MSMS method for various camera settings: static camera, pan-tilt camera, and free-moving camera. Robust segmentation results are achieved for datasets where objects have large translation, rotation, scaling, and illumination changes.

(iv) We propose and develop a novel method (LSMGS) for recovery of camera motion and object structure which takes advantage of both planar features and point features and fuses constraints from both homography and essential matrices.

(v) We evaluate the LSMGS method on both planar and non-planar objects/scenes. The method gives more accurate estimation results, compared to the approaches that use only one type of feature/constraint, and avoids their major drawbacks.

1.5 Thesis Organization

The remaining chapters are organized as follows.

Chapter 2 reviews related work, including various object representations such as 2D, $2\frac{1}{2}$D, 3D, and hierarchical models.

In Chapter 3, we present the framework of the OSH. Details of the Object Layers and Model Layers in the OSH are discussed.

Feature tracking is an important building block for the OSH. Chapter 4 describes various tracking methods for point features and planar region features. These tracking methods are used in the OSH for 2D object segmentation, 3D pose estimation, and 3D structure recovery.

To build the object models in the OSH, a fundamental step is to separate the object of interest from its background. In Chapter 5 we present a novel method for moving object segmentation based on both image information and motor signals. The method is evaluated against the pixel-level background subtraction and RANSAC-based homography/fundamental matrix fitting approaches which are widely used in existing works.

After an object is separated from the background, we aim at building its 3D model based on tracked features. Chapter 6 describes a new method for camera pose estimation by fusing two types of features (points and planar regions) and two types of constraints (homography and essential matrices). We evaluate the method on various datasets and compare it against the homography/essential matrix decomposition and Bundle Adjustment approaches.

With estimated camera motion parameters, preliminary results for 3D object/scene structure recovery is discussed in Chapter 7. The constructed model contains a small set of surfaces and the boundary of each surface is automatically identified. Chapter 8 summarizes the contributions of the thesis and discusses future directions.

Chapter 2

Literature Review

In order for the agent to build object models in the OSH from the high bandwidth sensor data (or the firehose of experience [Kuipers, 2008]), we need to answer the following questions: how do we represent objects and its background, how do we separate objects from its background, how do we track an object, how do we estimate the object pose and the pose of the observer, and how do we recover the 3D object structure? This chapter gives a brief introduction to related works in the topics of object representation, feature tracking, object segmentation, and 3D pose estimation and structure recovery.

2.1 Object Representation

To understand and interact with the external world in terms of objects, the agent must have an internal representation of the objects. Various models have been developed in the computer vision community, and we review 2D, $2\frac{1}{2}$D, 3D, and hierarchical object models in this section.

2.1.1 2D Object Models

Individual objects or object categories can be represented by a bag of features which is a set of visual words without consideration of their spatial arrangements. Viola and Jones [2001] used rectangular Haar-like features and selected a small set of critical features using AdaBoost [Freund and Schapire, 1995] for object detection. Grauman and Darrel [2006] learned feature masks for object categories by embedding sets of unordered image features into a space where they cluster according to their partial-match correspondences using the pyramid match kernel [Grauman and Darrell, 2005].

An object category can also be represented by a set of parts and their geometric relations. Burl and Perona [1996] developed a face model consisting of a set of facial features and the spatial arrangement among them. Weber *et al.* [2000] focused on learning object models that are represented as flexible constellations of rigid parts, where the variability within an object category is modeled by a joint probability density function on the configuration of parts and the output of part detectors. Fergus *et al.* [2003] extended the work of Weber et al. [2000] to learn shape, appearance, occlusion and relative scale of parts using the EM method. Felzenswalb and Huttenlocher [2005] presented a pictorial structure representation where an object is modeled by a collection of rigid parts arranged in a deformable configuration. Each part encodes local visual properties of objects, and the deformable configuration is characterized by spring-like connections between certain pairs of parts. Fei-Fei *et al.* [2006] demonstrated that an object category may be learned from one or just a few training examples by making use of knowledge from previously learned object categories rather than starting from scratch, based on salient region features and their locations. Shotton *et al.* [2005] used contour segments as parts and learned classifiers by the gentle-boost algorithm [Friedman et al., 2000]. While these methods can handle deformations, a major issue is that an enumeration over possible matchings for object detection between object models and the observed images limits the number of parts to be small.

For a specific object instance, its representation is usually used for object tracking.

Various 2D representations for object instances are reviewed in Section 2.2, together with tracking approaches based on the corresponding representations.

All the above works target at building 2D models rather than 3D models for objects. Without 3D models, it would be hard to use these representations for some robotic applications such as object manipulation.

2.1.2 $2\frac{1}{2}$D and 3D Object Models

A 3D object or object category can be represented by multiple 2D views corresponding to various poses. Pope and Lowe [2000] handled large variations in different views by clustering training images and small variations by characterizing uncertainty of object presence, location and appearance. Lowe [2004] used SIFT features from multiple 2D views to represent specific objects, where object recognition is done using each feature to vote for all consistent object poses. Schneiderman and Kanade [2000] used multiple detectors to deal with large pose variations and statistical modeling for the remaining variation for face and car detection. Torralba *et al.* [2004] demonstrated good performance of object detection by using shared features between views. Ferrari *et al.* [2006] used groups of local affine invariant features and their relations between multiple model views for object recognition. Although these methods handle the pose relations between different views, they do not consider the pose relations between features in the 3D space.

Kushal *et al.* [2007] used partial surface models and their loose geometric constraints to represent object categories, where partial surface models are dense, locally rigid assembles of texture patches learned by matching repeating patterns of features. Savarese and Fei-Fei [2007] proposed a model to represent and learn generic 3D object categories by linking together diagnostic parts of the objects from different viewing points, where parts are large discriminative regions and connected by their mutual homographic transformation. The object representations and inference methods in [Kushal et al., 2007; Savarese and Fei-Fei, 2007] are designed for the purpose of object detection and recognition, where the real 3D

poses of object parts are not explicitly obtained.

The above methods represent a 3D object by 2D features, parts, or views, plus their relations in the 2D image space, and do not investigate the real 3D pose between features, parts, or views. Hence they can be called $2\frac{1}{2}$D models. Similar to the 2D models described in the previous section, they are not very useful for robotic applications such as object grasping.

Rothganger *et al.* [2006] represented a 3D object in terms of local affine-invariant features and their 3D spatial relations. Each local affine-invariant feature is a planar parallelogram texture patch. This method has high reliance on texture and needs a large number of patches to accurately model objects.

A lot of work has been done to reconstruct full 3D models represented by voxels, polygon meshes, or depth maps [Seitz et al., 2006]. While these methods are able to provide high-quality reconstruction results with great details, they usually require stereo vision and high-resolution images, hence the computational cost is generally very high. For objects with simple shapes, the complexity of these models will be very high (for example, these models will end up getting a far more complex representation than a compact 6-face model for a cubic box).

2.1.3 Hierarchical Object Models

Marr and Nishihara [1978] proposed a hierarchical representation for 3D object shape models and an approach to object recognition where the basic shape components are 3D cylinders. Biederman [1987] built a structural object description where basic components are 2D "geons" such as blocks, cylinders, spheres and wedges. In these representations, the discrimination among objects in the same category is difficult, since objects are modeled as a small set of components with simple shape descriptors.

Bouchard and Triggs [2005] proposed a hierarchical model of object parts and subparts, with the object at the top level, and local image features at the bottom. While this

method deals with a large number of local features, the number of parts at each level needs to be manually tuned and only three layers (object, part and feature) were tested in their experiments. Epshtein and Ullman [2007] proposed semantic hierarchies which can detect and represent multiple appearances of the same object at all levels. Sudderth *et al.* [2005] presented a hierarchical model for objects, the parts composing them, and the scenes surrounding them. Each object category has its own distribution over a set of parts which describes the expected appearance and location in the object centered coordinates, and parts are shared between objects. Parikh and Chen [2007] presented hierarchical semantics of objects (hSOs) that capture relationships among multiple objects in a scene as observed by their relative positions in a collection of images. This hierarchy is a decomposition of the scene in terms of multiple objects. Marszalek and Schmid [2007] proposed a semantic hierarchy for inter-class object relationships. All these methods build a hierarchical representation for objects, but none of them includes 3D object models. In addition, these methods are intended for object recognition, and are not suitable for reconstruction of the input sensory stream.

2.2 Feature Tracking

To separate an object from the background and to build models for the object, we need to track features in the background or on the object. Feature tracking is an important process in construction of the OSH.

Modayil and Kuipers [2004; 2006; 2008] developed a method whereby a learning agent can autonomously learn about object models, by detecting, tracking, and characterizing clusters of foreground "pixels" in the sensory stream. Their agent is a mobile robot that receives a stream of sensory information from a laser range-finder. It is assumed that the agent has learned the structure of its sensory array using the methods of Pierce and Kuipers [1997].

For camera images, various features can be used in object tracking, such as interest

points/patches [Lowe, 2004; Tran and Davis, 2007; Shi and Tomasi, 1994; Kwon and Lee, 2010; Molton et al., 2004], lines, edges/contours/boundaries [Pressigout and Marchand, 2006; Isard and Blake, 1998; Kass et al., 1988], blobs/regions [Comaniciu et al., 2003; Gall et al., 2008b; Matas et al., 2004], and combinations of them.

Background subtraction is a popular technique for object detection and tracking. Wren *et al.* [1997] used a single Gaussian to model pixel intensity in the background. Stauffer and Grimson [1999; 2000] used a mixture of Gaussians to handle multi-modal intensity distributions. Stereo disparity information was used to coarsely segment foreground objects from the background in Kim et al. [2006].

Comaniciu *et al.* [2003] proposed a kernel-based tracking algorithm where an object is represented by an ellipsoidal region and the mean-shift tracker [Comaniciu and Meer, 2002] maximizes the appearance similarity iteratively by comparing a histogram-based model of the object and the window around the hypothesized object location. Isard and Blake [1998] presented a particle filter based tracking algorithm where object shape is represented by B-splines. Tran and Davis [2007] presented a robust object tracking method using regional affine invariant features.

In particular, distinctive point features have been widely used in object tracking, such as by the KLT method [Shi and Tomasi, 1994] or the SIFT matching method [Lowe, 2004]. While point features have many successful applications, maintaining feature tracks over many frames may be quite difficult [Tran and Davis, 2007; Zinsser et al., 2004], especially when the input images are noisy. The KLT method is efficient, but it may suffer from the feature drift problem over a long sequence of images [Zinsser et al., 2004; Bourel et al., 2000; Gall et al., 2008a].

More robust tracking can be obtained by integrating other features with point features. Gall *et al.* [2008b] exploited region matching to avoid point feature drift. Pressigout and Marchand [2006] and Vacchetti *et al.* [2004] incorporated point and edge features for tracking by minimizing the re-projection errors.

2.3 2D Object Segmentation

Motion information is an important cue to separate out foreground from the background [Spelke, 1990]. There have been many approaches developed for foreground object segmentation based on motion information.

Background subtraction has been widely used for moving object segmentation such as Pfinder [Wren et al., 1997], non-parametric model [Elgammal et al., 2000], Gaussian Mixture Model [Stauffer and Grimson, 1999, 2000], and their variations [Zhong and Sclaroff, 2003; Mittal and Paragios, 2004; Monnet et al., 2008; Ko et al., 2010; McKenna et al., 2000]. These methods build a pixel-level background model which adapts to the changing environment, and identify those pixels that violate the background model as foreground pixels. These methods typically assume a stationary camera, which makes them hard to be used for robotic applications. To further relax this assumption, many ego-motion compensation [Hayman and Eklundh, 2003; Mittal and Huttenlocher, 2000] or image mosaic [Brown and Lowe, 2003; Szeliski, 2006; Azzari et al., 2005] methods have been presented, but they may result in blurred edges due to accumulated image registration error.

Another way for moving object segmentation is to build parametric models for different motion layers. Motion analysis is applied to either sparse features, dense optical flows. In general, a set of affine/homography parameters [Ren and Gu, 2010; Han et al., 2006; Sivic et al., 2006; Wang and Adelson, 1994] or trajectory parameters [Sheikh et al., 2009; Brox and Malik, 2010] need to be estimated by iterative linear regression [Han et al., 2006], RANSAC [Sheikh et al., 2009; Ren and Gu, 2010; Wang and Adelson, 1994], or hierarchical k-means [Brox and Malik, 2010]. These approaches obtain good performance for moving object segmentation, but they may be computationally expensive due to the iterative parameter searching process.

Object segmentation can also be conducted based on region tracking [Lee et al., 2011; Ren and Malik, 2007; Grundmann et al., 2010; Yin and Collins, 2009], where low-level over-segmented regions are grouped into high-level object-like regions by static and

dynamic cues such as appearance similarity, spacial coherence, and motion coherence. These methods do not detect real 3D motion, for example, a static object may be separated out if it moves in the 2D image space but remains static in the 3D space with respect to other objects or background.

All the above methods appeal to using information from images only, and do not take advantage of robot pose information which is another important source of information to separate moving objects from the static background in robotic applications.

2.4 3D Pose Estimation and Structure Recovery

Information is accumulated while an object is tracked, and the 3D camera/object pose and the 3D object structure can then be recovered from this accumulated information. Two classic approaches for recovery of the 3D camera pose and object structure have been widely used from two views of the same 3D scene: the homography matrix based approach [Sturm, 2000; Zhang, 2000; Hartley and Zisserman, 2003; Ma, 2004; Cobzas et al., 2009; Molton et al., 2004] and the essential matrix based approach [Pollefeys et al., 2004; Nistér, 2004; Zhang, 1998; Longuet-Higgins, 1981; Ma, 2004; Hartley and Zisserman, 2003]. The homography matrix based approach may give two physically possible solutions [Ma, 2004], and it can be very difficult to select the correct solution when the camera motion is small or when the camera moves on a line without any rotation. The essential matrix based approach requires RANSAC-like fitting and a good number (at least 5 [Nistér, 2004], usually many more) of point features that do not lie on the same plane.

Other existing approaches for camera motion estimation and object structure recovery include Bundle Adjustment [Sibley et al., 2009; Engels et al., 2006; Hartley and Zisserman, 2003; Triggs et al., 2000] and Visual SLAM [Davison et al., 2007; Newcombe and Davison, 2010; Nister et al., 2006; Eade and Drummond, 2006; Cummins and Newman, 2009]. These approaches estimate high dimensional parameters simultaneously, and typically need prior knowledge on the initial camera pose and object structure. When the number of image fea-

tures is large, the size of the search space for parameter estimation will be extremely high, which will result in expensive computation.

Chapter 3

The Object Semantic Hierarchy

An intelligent agent must be able to perceive and reason robustly about its world in terms of *objects*, among other foundational concepts. The robot can draw on rich data for object perception from continuous sensory input, in contrast to the usual formulation that focuses on objects in isolated still images. Additionally, the robot needs multiple object representations to deal with different tasks and/or different classes of objects [Logothetis and Sheinberg, 1996].

We are developing the Object Semantic Hierarchy (OSH) [Xu and Kuipers, 2010, 2009] to build a collection of object representations at different layers, motivated by the work of the Spatial Semantic Hierarchy (SSH) which consists of multi-level representations of large-scale space [Kuipers, 2000; Kuipers et al., 2004; Beeson et al., 2010].

3.1 The OSH Framework

The framework of the OSH is shown in Fig. 3.1. The OSH has two types of layers: the Object Layers and the Model Layers. The Object Layers describe how the static background and each foreground object are individuated, and the Model Layers describe how the model for the static background and for each foreground object evolves from less structured to

more structured representations.

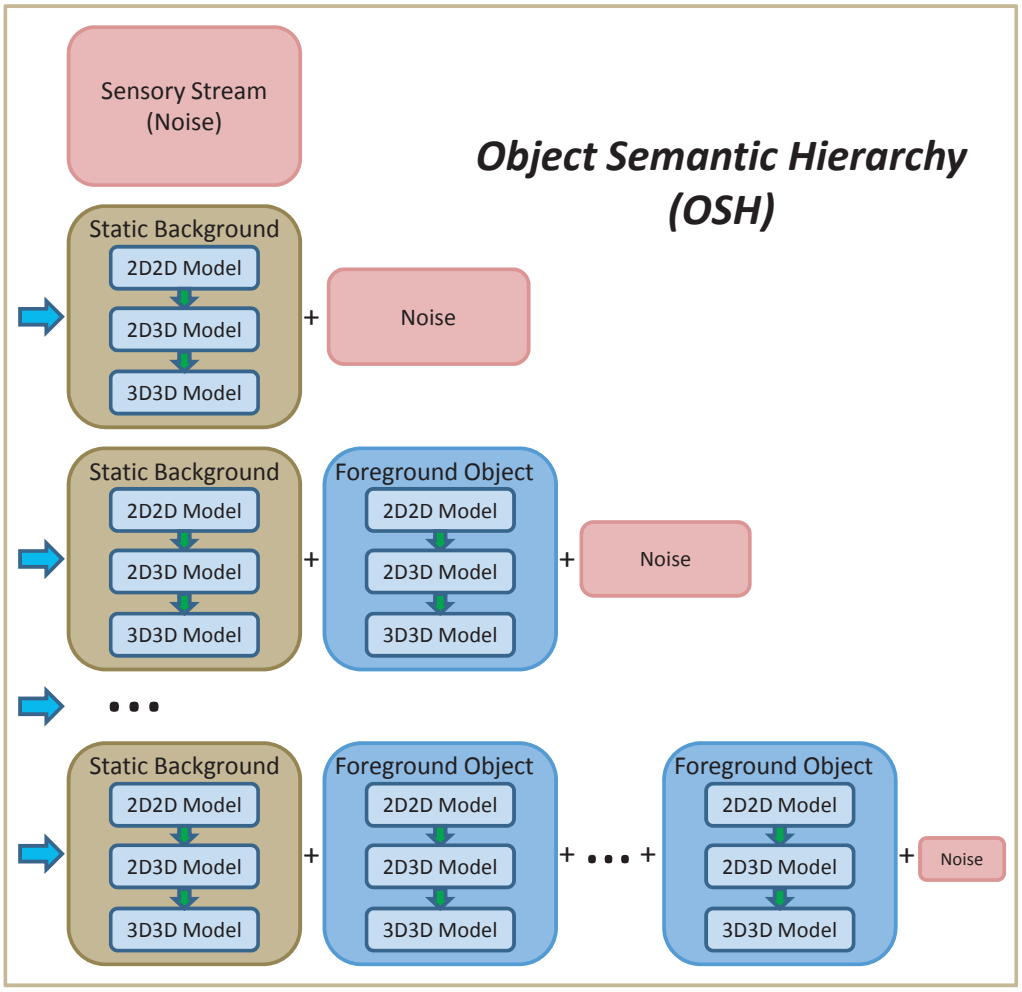


Figure 3.1: The framework of the OSH (best viewed in color). The agent initially treats everything in the sensory stream as noise. By repeatedly identifying new invariants to reduce the noise, the agent progressively builds models for the background world and foreground objects. For the background world or each foreground object, the model evolves from 2D2D to 2D3D to 3D3D (see text for details).

In the Object Layers, the agent starts by constructing a constant model of the static background world, where foreground objects are treated as noise. Then the foreground objects are progressively individuated from the background and their models are constructed

while they are tracked over time. Hierarchical models are created by repeatedly constructing and refining stochastic models of the observation stream generated by the agent's sensors in the environment. Such a stochastic model has the form $z_t = M_t + \varepsilon$, where M_t is a deterministic model explaining the contents of the observation stream z_t, and $\varepsilon = z_t - M_t$ is the residual between explanation and observation, interpreted as noise. At each layer, new invariants are identified within the data described by ε, leading to a revised model M_t' and ideally a reduced level of noise $\varepsilon' = z_t - M_t'$. In the end, the uncertainty in the sensory stream is factored into a collection of relatively compact representations: static background model, pose trajectory of the observer, constant foreground object models, pose trajectories of the foreground objects, and any remaining noise. The "blooming, buzzing confusion" of the initial pixel-level input is concisely explained in terms of a relatively small number of object-level concepts and relations.

In the Model Layers, the static background is treated as just another object. The construction of the static background model is taken in the same way as of any foreground object model. The models for each object model contain the following layers:

(a) **2D object in 2D space (2D2D)**: a set of constant 2D object views, and the time-variant 2D object poses;

(b) **2D object in 3D space (2D3D)**: a collection of constant 2D components, with their individual time-variant 3D poses;

(c) **3D object in 3D space (3D3D)**: the same collection of constant 2D components but with invariant relations among their 3D poses, and the time-variant 3D pose of the object as a whole.

Note that in the these layers, it is possible that a part of the object has evolved to a higher-level model but the remaining part is still represented in lower-level models.

The idea of the Model Layers is that early stages of analysis can robustly derive certain properties of the visual scene, that are then used as assumptions to make later pro-

cessing layers simpler and more robust. For example, early separation of the background provides the agent a set of 2D views of the foreground object, which allows the agent to focus on only the foreground object without the need to worry about the background. This facilitates later processing for 3D model construction for the foreground object. When and if later layers fail, the earlier layers still allow objects to be tracked in the image, until they are more accessible to the more sophisticated kinds of analysis.

3.2 Object Layers

In the Object Layers, the agent starts by building a constant model of the static background world, where foreground objects are treated as noise. Once the background model is built as an explanation of its sensory stream, the agent continues to progressively individuate foreground objects by identifying new invariants within the discrepancy between the agent's explanation and observation.

Layer 0: Noisy world

The agent perceives its environment through a high dimensional pixel-level sensory stream. In this layer, everything is considered as noise.

$$z_t = \varepsilon_0 \tag{3.1}$$

where z_t is the sensor input at time t, and ε_0 is a random variable that represents the sensor input but treats it as noise.

Layer 1: Static background

In its learning process, the agent starts by constructing a constant model of the background world, treating any foreground objects as noise.

$$z_t = G_1(M^b, x_t) + \varepsilon_1 \tag{3.2}$$

where M^b is the static background model, x_t is the agent's observing pose, G_1 is a function mapping the background model M^b to a 2D image given the observer's pose x_t, and ε_1 represents the discrepancy between the prediction of the model $G_1(M^b, x_t)$ and what is actually observed z_t.

Layer 2: Foreground object 1

After the static background model is constructed, new invariants are identified within the data described by ε_1. The identified invariants contribute the first foreground object.

$$z_t = G_1(M^b, x_t) + G_2(M_1^o, y_{1t}) + \varepsilon_2 \tag{3.3}$$

where M_1^o is the constant model for the first foreground object, y_{1t} is the object's pose, G_2 is a function mapping the object model M_1^o to a 2D image given y_{1t}, and ε_2 is the remaining noise. The plus sign is an operator that layers the foreground object image on top of the background image.

$$\vdots$$

Layer n: Foreground object n-1

At Layer n, the agent continues to identify new invariants within the noise term ε_{n-1} in the previous layer, and constructs a model for the $n-1^{th}$ foreground object.

$$\begin{aligned} z_t = G_1(M^b, x_t) + G_2(M_1^o, y_{1t}) + ... + \\ G_n(M_{n-1}^o, y_{(n-1)t}) + \varepsilon_n \end{aligned} \tag{3.4}$$

where M_{n-1}^o is the constant model for the $n-1^{th}$ foreground object, $y_{(n-1)t}$ is the object's pose, G_n is a function mapping the object model M_{n-1}^o to a 2D image given $y_{(n-1)t}$, and ε_n is the remaining noise.

3.3 Model Layers

In the Model Layers, the static background model and each foreground object model have the following layers: 2D object in 2D space, 2D object in 3D space, and 3D object in 3D space. Here an *object* can be either the background world or any foreground object. In other words, the background world is treated as just another object. The object pose of the background world in the egocentric frame of reference is an implicit representation for the agent's observing pose in the allocentric frame of reference.

While the agent always tries to build all the Model Layers for an object, it can fall back to the already-constructed layers if at a certain layer the transition to the next is not feasible. Thus, the agent will still be able to work under lower-level models when higher-level models are not available.

3.3.1 Layer 2D2D: 2D Object in 2D Space

From the high-dimensional pixel-level object image stream, the agent identifies a sparse set of 2D object views as the object model v. The multi-view object representation has been shown very useful for object recognition [Bulthoff and Edelman, 1992; Ullman, 1998; Lowe, 2001].

The 2D object view model is described by

$$v = \{v_1, v_2, ..., v_{n^v}\} \tag{3.5}$$

where n^v is the number of the object views. The object views are connected by shared image features and/or the agent's motor signals.

The view model v should satisfy two constraints: (i) v is sparse compared to all the input object images, and (ii) v is complete such that any observed object image can be generated from v.

At each time step t, within the object model v, we locate the view that has closest

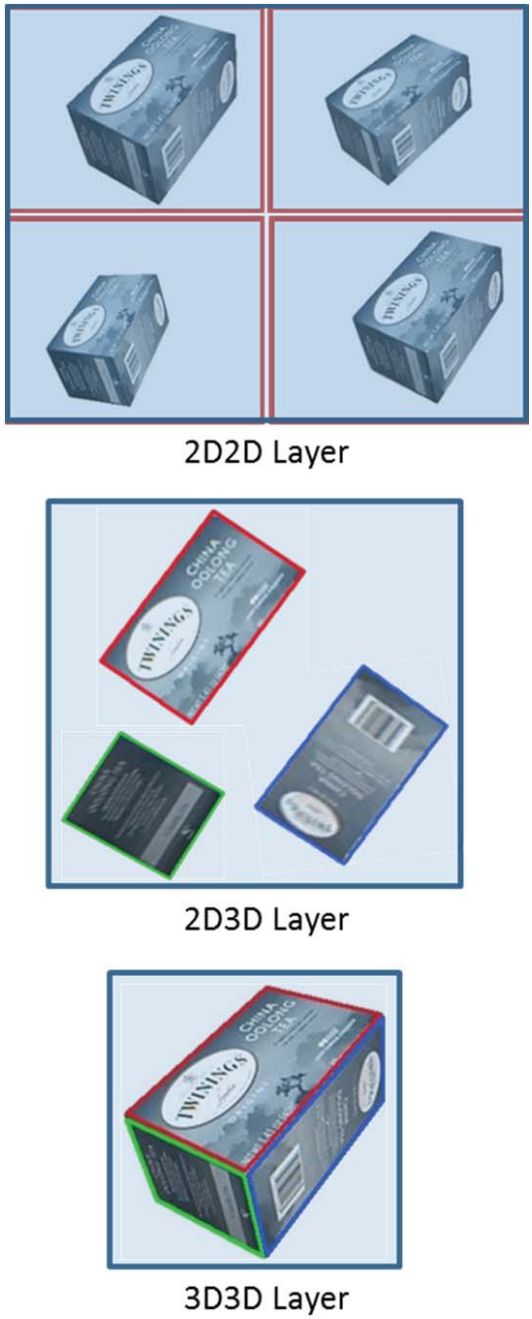


Figure 3.2: Examples for the Model Layers. 2D2D layer – a collection of 2D views, 2D3D layer – a set of 2D planar components, 3D3D layer – full 3D object.

observing pose with the new input image, by checking the overlapping ratio between each view and the input image. This located view is called the base view. The homography transformation between the relevant portion of the input image and the base view, plus the pointer to the base view, is defined as the 2D object pose y_t^v. With this y_t^v, any observed object image can be reconstructed as an image transformed from the base view and the neighboring views of the base view.

Now we have

$$z_t = G_v(v, y_t^v) + \varepsilon_v \tag{3.6}$$

where G_v is a function mapping v to an image under y_t^v, and ε_v is the remaining noise.

In Eq. 3.6, the object image stream is decomposed into the constant 2D object view model v (Fig. 3.2), dynamic 2D object pose y_t^v, plus the remaining noise.

3.3.2 Layer 2D3D: 2D Object in 3D Space

Psychological experiments have shown that humans focus their study time on object views that are close to planar views (such as front, back, and side views) and ignore other views when actively interacting with objects [James et al., 2002].

Based on the 2D2D layer, we identify new invariants as a collection of constant 2D components, which are planar or approximately planar surfaces embedded in 3D space and are denoted by

$$c = \{c_1, ..., c_{n^c}\} \tag{3.7}$$

where n^c is the number of components, and each component in c is represented by its normal view. A normal view is defined as the component image that is observed when the optical axis is aligned with the normal of the component surface.

The object view model v in the previous layer can then be described by

$$v = G_q(c, q) \tag{3.8}$$

where $q = \{q_1, ..., q_{n^c}\}$ are the 3D poses of the components appearing in the 2D views in v, and G_q is a function mapping c and q to v.

Let y_t^c denote the dynamic 3D component poses. The 2D object pose y_t^v in the 2D2D layer and the component poses q in Eq. 3.8 can both be represented as functions of y_t^c. Thus, by combining Eq. 3.6 and Eq. 3.8, we get

$$\begin{aligned} z_t &= G_v(G_q(c, q), y_t^v) + \varepsilon_v \\ &= G_c(c, y_t^c) + \varepsilon_c \end{aligned} \tag{3.9}$$

where G_c is a function mapping c to an image under y_t^c, and ε_c is the remaining noise. Note that y_t^c contains a history of the 3D poses for each individual component, where the 3D poses between different components are not related yet.

In Eq. 3.9, the object image stream is decomposed into the constant 2D object component model c (Fig. 3.2), dynamic 3D component poses y_t^c, plus the remaining noise.

3.3.3 Layer 3D3D: 3D Object in 3D Space

Compared to the multi-view representation in the 2D2D layer, a structured description (parts with relative 3D pose relations) of objects allows the agent to evaluate components and their relations independently [Hummel, 2000]. In addition, a structured description tends to be more concise than the multi-view representation.

We now begin to relate individual components to each other, to create a fixed 3D structure with a number of different components. The relation between the 3D poses of two

components is invariant under the assumption of rigid object.

$$y_t^c = G_p(p, y_t^o) \tag{3.10}$$

where y_t^o is the object's 3D pose at t, $p = \{p_1, ..., p_{n^c}\}$ are the 3D poses of the components with respect to the object pose y_t^o, and G_p is a function that maps p to y_t^c under y_t^o. All the changing component poses in Eq. 3.9 are explained in terms of the changing pose of the 3D object as a whole.

We define the 3D object model in 3D space as

$$o = \{c, p\} \tag{3.11}$$

where both c and p are constant.

By combining Eq. 3.9, Eq. 3.10, and Eq. 3.11, we get

$$\begin{aligned} z_t &= G_c(c, G_p(p, y_t^o)) + \varepsilon_c \\ &= G_o(o, y_t^o) + \varepsilon_o \end{aligned} \tag{3.12}$$

where G_o is a function mapping the 3D object model o to an image under the 3D object pose y_t^o, and ε_o is the remaining noise. Note that the subscript of ε_o in the above equation is "o" instead of "0" (zero) as in Eq. 3.1.

In Eq. 3.12, the object image stream is decomposed into the constant 3D object model o (Fig. 3.2), dynamic 3D object pose y_t^o, plus the remaining noise.

3.4 Re-projection Functions

The re-projection functions G_v, G_c, G_o, G_q and G_p are summarized in Table 3.1. Each of these functions is a well-understood transformation matrix [Hartley and Zisserman, 2003;

Table 3.1: Summary of transformation functions

Function	Description
G_v	Given the constant 2D object view model and the dynamic 2D object pose, predict sensor input.
G_c	Given the constant 2D component models and the dynamic 3D component poses, predict sensor input.
G_o	Given the constant 3D object model and the dynamic 3D object pose, predict sensor input.
G_q	Given the models of a set of 2D components and their poses, predict 2D views in image space.
G_p	Given 3D object pose, and 3D component poses wrt object frame, predict 3D component poses in world frame.

Table 3.2: Acquired information in the OSH

Layer	Acquired information
2D2D	v - constant 2D object view model y_t^v - dynamic 2D object pose
2D3D	c - constant 2D object component models y_t^c - dynamic 3D component poses
3D3D	o - constant 3D object model y_t^o - dynamic 3D object pose

Ma, 2004]. Table 3.2 is a summary of the information that is acquired at different layers.

3.5 Static Background as a Special Object

In the Model Layers, we have described object poses as y_t^v, y_t^c and y_t^o in the agent's egocentric frame of reference. For the static background as a special object, y_t^v, y_t^c and y_t^o are implicit representations for the agent's observing pose in the allocentric frame of reference. In later discussion, we will use x_t^v, x_t^c and x_t^o to denote the agent's observing pose in different Model Layers in the OSH (Fig. 3.3).

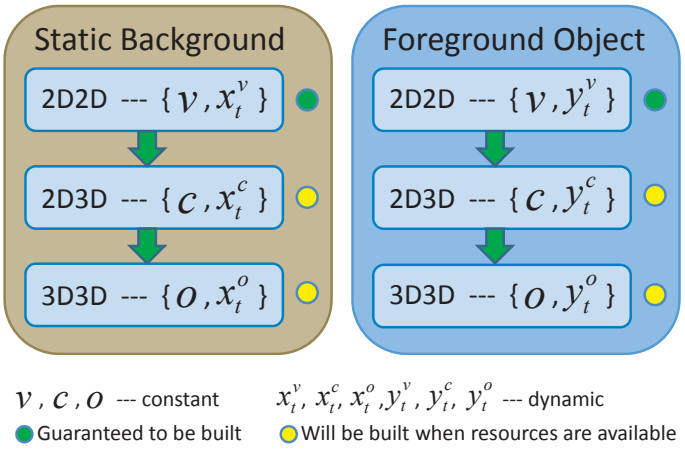


Figure 3.3: Model Layers in the OSH. The background model and foreground object models $\{v, c, o\}$ are constant. The observer's pose $\{x_t^v, x_t^c, x_t^o\}$ and the foreground object poses $\{y_t^v, y_t^c, y_t^o\}$ are time-variant. The 2D2D model is guaranteed to be built, and the 2D3D and 3D3D models will be built when the agent has necessary resources available.

3.6 Construction of the OSH

We will briefly describe our approach to building the OSH in this section, where the details for 2D background/object segmentation and 3D pose estimation will be discussed in Chapter 5 and Chapter 6. In the current implementation of the OSH, we assume the foreground objects are rigid.

3.6.1 Static Background Model

The static background environment usually has more complex structure than foreground objects. While our ultimate goal is to build a full 3D3D model for the background environment (in Tsai, Xu, Liu, and Kuipers [2011], we have presented a method to construct 3D3D background models in planar environments containing only ground and walls), we only focus on constructing a 2D2D background model in this thesis.

When the observing pose is fixed, we can use the Gaussian mixture model (GMM) [Stauffer and Grimson, 1999] to build a pixel-level background image and wash out noises

Figure 3.4: A robot looking at a checker box. A box is moving around in front of the robot, which has a webcam mounted on a pan tilt unit.

due to dynamic changes [Xu and Kuipers, 2009]. When the observing pose is dynamic, the variant GMM method by stitching images in a single view [Brown and Lowe, 2003; Szeliski, 2006; Torr and Zisserman, 2000; Azzari et al., 2005; Hayman and Eklundh, 2003] suffers from accumulated image registration error. Instead, we identify a sparse set of views as the 2D2D background model v, where these views are connected by homography transformations. The homography transformations are obtained based on the robot's motor signals and the shared image features between overlapping views. When a new input image has small or no overlapping with all views in the 2D2D background model, the image is added to the model as a new view. If the input image has a large overlapping with a view in the model, the image is used to update the view.

At each time step, within the 2D2D background view model, we find the base view that has the largest overlapping ratio with the new input image. Features are detected and matched between the input image and the base view. A homography matrix is calculated between these features. This homography matrix, together with the pointer to the base view, is the agent's observing pose x_t (here x_t is actually the observing pose x_t^v in the 2D2D layer in the OSH).

Fig. 3.4 shows a simple robot which has a webcam mounted on a pan tilt unit (PTU). The robot has access to its motor signals, that is, pan and tilt positions of the camera.

To construct the 2D view model for the static background, the agent needs to be

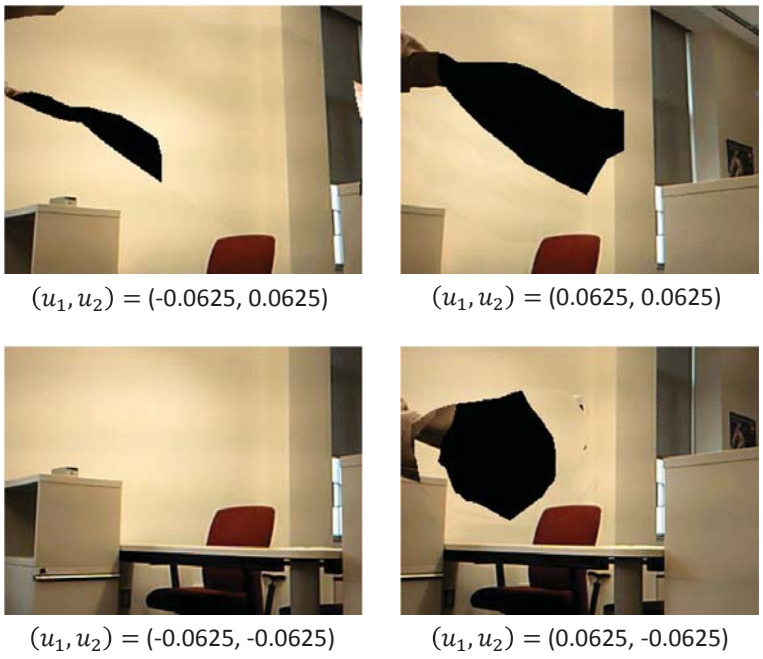


Figure 3.5: 2D view examples in the 2D2D background model. The views correspond to different observing poses, where each observing pose is determined by the pan and tilt positions (shown below the views) of the PTU. The black holes in the views are due to permanent occlusion by foreground objects.

able to identify which part in each input image comes from the background (the remaining part is treated as noise). Motor signals allow the robot to predict the motion patterns of background features. In contrast, the motion patterns of non-background features will be different from the predictions because they have independent motions from the robot. This observation provides us a way to cluster image features based on their discrepancy with their predictions. Based on this observation, we first detect sparse features and label them as background features and non-background features. Then we propagate their labels to all pixels in the input image. This method segments an image into background pixels and non-background pixels based on only a few neighboring frames. It is robust to illumination changes, adapts quickly to the environment, and does not suffer from accumulated image registration error. The details of the method is presented in Chapter 5.

Fig. 3.5 shows some view examples (and the corresponding motor signals) in a constructed 2D2D background model.

3.6.2 Foreground Object Model

2D Object Model

Once the background model is constructed, foreground object pixels can be individuated from the background. For each new input image, we detect sparse features and match them with the 2D2D background model. Those features that violate the background model are labeled as foreground features. Then the labels are propagated to all image pixels.

Within the 2D foreground object images where the background pixels have been filtered out, the 2D2D foreground object model v and the object pose y_t^v are obtained in a similar way as in the 2D2D background model construction. Please see Chapter 5 for details. Once the foreground object is segmented out from the background, various information such as contours can be extracted and used for object recognition [Xu and Kuipers, 2011].

3D Object Model

In the foreground object image sequence, we track a single planar component and estimate its 3D pose. The tracking method includes both sparse point feature tracking and dense pixel alignment, which provides accurate and robust tracking results for pose estimation.

Pose estimation takes advantage of both planar features and point features, and fuses constraints from both the homography and essential matrices. Once the 3D pose of the planar component is estimated, we are able to obtain camera motion parameters and then recover the 3D object structure. The details are described in Chapter 6 and Chapter 7.

Chapter 4

Feature Tracking

Feature tracking is an important building block for the OSH. Tracking allows the robot to make correspondence between images, perform foreground/background separation, and recover scene/object structure.

Various features can be used in tracking [Yilmaz et al., 2006], such as interest points or patches [Lowe, 2004; Tran and Davis, 2007; Shi and Tomasi, 1994; Bay et al., 2008; Molton et al., 2004; Kwon and Lee, 2010], lines, edges/contours [Isard and Blake, 1998; Kass et al., 1988; Pressigout and Marchand, 2006], blobs/regions [Comaniciu et al., 2003; Gall et al., 2008b; Matas et al., 2004], and combinations of them. In this chapter, we will discuss tracking of point/patch features and planar region features which are used in the current implementation of the OSH.

4.1 Point/Patch Feature Tracking

The most popular point/patch feature tracking methods are KLT tracking [Shi and Tomasi, 1994] and SIFT matching [Lowe, 2004].

The KLT [Shi and Tomasi, 1994] tracker estimates the optical flow between two frames by minimizing the sum of squared difference within each image patch between one

frame and the other. First, salient corner features are detected by checking the eigenvalues of each local 2×2 gradient matrix. Then the detected corner features (represented as windows of small patches) in the first frame are tracked by iteratively searching for the best matching windows in the following frames. Multi-resolution tracking (coarse-to-fine estimation) can be used when there are relatively large displacements between images [Shi and Tomasi, 1994].

Tracking by SIFT matching [Lowe, 2004] first detects point/patch features independently between frames, and then find feature correspondence between these frames. SIFT matching is based on the Best-Bin-First algorithm, which uses a modified search ordering for the k-d tree algorithm [Friedman et al., 1977] so that bins in feature space are searched in the order of their closest distance from the query location. This search order requires the use of a heap-based priority queue for efficient determination of the search order [Lowe, 2004].

In our current OSH implementation, we use the KLT method, due to its efficiency and effectiveness in videos where only small motion takes place between consecutive frames, for 2D foreground segmentation, 3D pose estimation, and 3D structure recovery.

4.2 Planar Region Feature Tracking

In building the 2D3D models in the OSH, we develop a 3D pose estimation method that uses both point features and planar region features, and we demonstrate that using planar region features greatly improves the results of using only point features. A planar region feature is a region that lie on a planar surface, where the region is usually larger than small patches described in the above section (please see Section 6.2 for planar region feature detection).

For tracking a planar region feature, we can detect interest points inside the region, and track the planar region feature by tracking the detected interest points based on the KLT method. This generally works well for a small number of consecutive frames. However, for a long sequence of images, it may suffer from the feature drift problem (the features

drift slowly away from the correct positions from frame to frame), and will accumulate errors due to intensity changes, image noise and inaccurate parameter estimation. The drift between adjacent frames is very small, but the accumulated error across a long sequence of frames can be very problematic for 3D pose estimation.

Fig. 4.1 shows the feature drift problem in KLT tracking. We investigate two methods to solve this problem, by line feature integration and by dense pixel alignment.

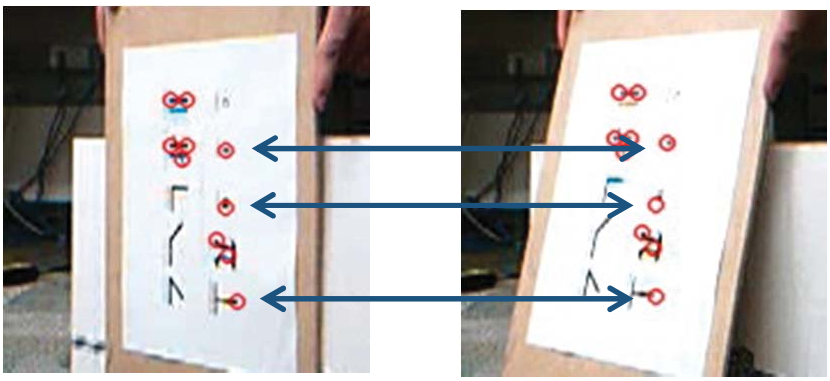


Figure 4.1: Feature drift problem. Some of the features (shown as red circles) detected in the left image may gradually drift away from the correct positions, as shown in the right image, especially for the linked feature pairs. When the number of features is low, even small drifts can seriously affect the tracking results and the accuracy of later pose estimation results.

4.2.1 Sparse Point Feature Tracking and Line Feature Integration

We consider the case where the to-be-tracked planar region feature has a clear boundary composed of piecewise line segments. Compared to point features, line features are much more robust when the lighting condition changes or when significant noise exists, so they can be integrated to improve tracking accuracy.

We exploit the KLT algorithm to maintain only temporary correspondence between each two adjacent frames based on point features, while the permanent correspondence across all the frames is maintained by boundary line features. The temporary correspondence from the KLT algorithm is only used to predict the boundary location from one

frame to the next. Then, within the local areas of the predicted boundary, a correction step is taken by selecting the best matched line segment detected using the Hough transform [Duda and Hart, 1972].

Point Feature Tracking

We track KLT point features only between each two adjacent frames instead of across multiple frames due to the possible feature drift problem. That is, features are re-detected at each time step and different sets of features are used between different adjacent frames.

Salient point features are first detected inside the boundary at time $t-1$. Then the features are tracked by the KLT tracker at time t. The RANSAC algorithm [Fischler and Bolles, 1981] is used to remove incorrectly matched features. In our experiments, the KLT tracker works very robustly since it only has to maintain the feature correspondence between two consecutive frames. This feature correspondence will be used to predict the boundary location at time t, based on the already-known boundary location at time $t-1$. This tracking scheme is demonstrated in Fig. 4.2.

Boundary Prediction

The detected features at time $t-1$ and the tracked features at time t are related by a planar homography transformation H_t^p (the superscript p denotes *point*). Given at least four pairs of matched point features, H_t^p is calculated through the Direct Linear Transformation (DLT) method [Hartley and Zisserman, 2003; Ma, 2004; Agarwal et al., 2005].

Let the boundary at time $t-1$ be s_{t-1}^c, then we have

$$\hat{s}_t^c = H_t^p s_{t-1}^c \tag{4.1}$$

where $\hat{s}_t^c$ is the predicted boundary at time t.

Boundary Correction

In order to avoid accumulated error from point feature tracking, we update the predicted

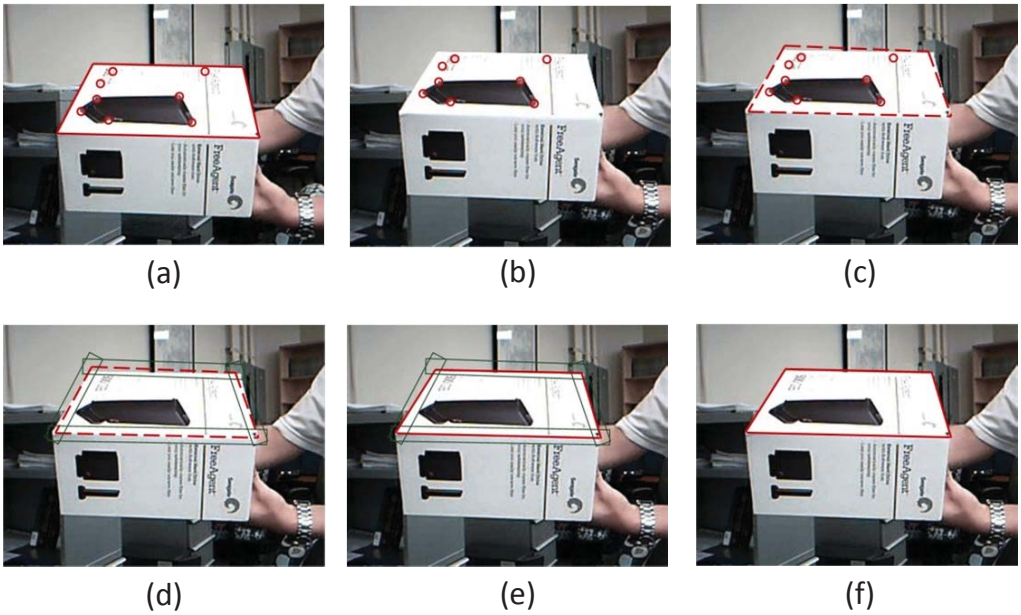


Figure 4.2: Planar region feature tracking with line feature integration. The first figure is based on the image at $t-1$, and all the others are based on the image at t. (a) The red solid contour is the boundary of the planar region in frame $t-1$, and point features are detected inside the boundary shown as red circles. (b) The point features are tracked in frame t using the KLT tracker. (c) The boundary in frame t is predicted as a dashed contour from feature correspondence between these two adjacent frames. (d) Local interest regions (shown as green rectangles) are formed around each line segment. (e) Within each local interest region, lines are detected using Hough transform and the best matched line is selected (shown as a red solid line). (f) The final boundary in frame t is computed.

boundary to fit the observed data by matching line segments in their neighborhood areas. Compared to point features, line features are much more robust to intensity changes and image noise. Also line features tend to give more accurate position estimation than point features.

Around each line segment on the predicted boundary $\hat{s}_t^c$, a local interest region is formed in the image at time t. The interest region is a rectangle with a user-defined height and width, where the rectangle's centroid is the line segment's centroid, and two laterals of the rectangle are parallel to the line segment. Within this rectangle, candidate line segments are detected using the Hough transform [Duda and Hart, 1972] after the Canny edge

detection process [Canny, 1986], and the best matched line segment is used to correct the predicted line segment.

From at least four pairs of matched line features, another homography matrix H_t^b (the superscript b denotes *boundary*) is obtained [Guerrero and Sagues, 2003], and the boundary at time t is finally updated as

$$s_t^c = H_t^b \hat{s}_t^c \tag{4.2}$$

or

$$s_t^c = H_t^b H_t^p s_{t-1}^c \tag{4.3}$$

by substituting Eq. 4.1.

Let H_t be the homography transformation between frame 0 and frame t. Then the tracked boundary at time $t-1$ is

$$s_{t-1}^c = H_{t-1} s_0^c \tag{4.4}$$

where s_0^c is the boundary at time 0. Combining Eq. 4.3, we can easily get H_t as

$$H_t = H_t^b H_t^p H_{t-1}. \tag{4.5}$$

where H_t will be used later for 3D pose estimation.

Discussions

In this tracking scheme, point features maintain only temporary correspondence between each two adjacent frames, while line features maintain the permanent correspondence across all the frames for the tracked planar region. Since line features are more robust to image noise, this tracking scheme can work well with low quality videos.

The Hough transform is applied only within the local interest regions of the predicted boundary. In general the KLT tracking is fairly accurate between adjacent frames, so the interest regions are typically small such that the computational cost for boundary correction is low.

4.2.2 Sparse Point Feature Tracking and Dense Pixel Alignment

In a more general case, where the planar region feature does not have a natural linear boundary, a dense pixel alignment step can be applied to avoid the feature drift problem and improve tracking accuracy.

This planar region feature tracking scheme includes two steps: sparse point feature tracking and dense pixel alignment. Sparse point feature tracking gives a rough estimate of the planar region feature location, and dense pixel alignment refines the estimate. Combining sparse feature tracking and dense pixel alignment allows us to track a planar region feature robustly and accurately. The tracking scheme is shown in Fig. 4.3.

In this section, we also describe a solution to deal with the problem when the number of detected point features inside the to-be-tracked planar region is very small and the result of point feature tracking is poor. This solution can be applied as well to the tracking scheme with line feature integration described in the above section, when we have only a low number of point features inside the planar region. The idea is to exploit information from outside the planar region as well as information inside the region.

Traditionally feature tracking [Shi and Tomasi, 1994; Comaniciu et al., 2003] or matching [Lowe, 2004] uses information only inside the tracked region, while the information outside the region is completely or mostly overlooked. When the tracked region is small or similar to its neighboring areas, it will be hard to robustly locate the region without using any context information. We take advantage of information both inside and outside the tracked region to improve tracking stability and accuracy. Our tracking method takes two steps: sparse feature tracking on the entire image (both inside and outside the tracked

region) and dense pixel alignment inside the tracked region. The sparse feature tracking step avoids getting stuck at local minima and improves tracking stability, and the dense pixel alignment step refines the tracked feature location and improves tracking accuracy.

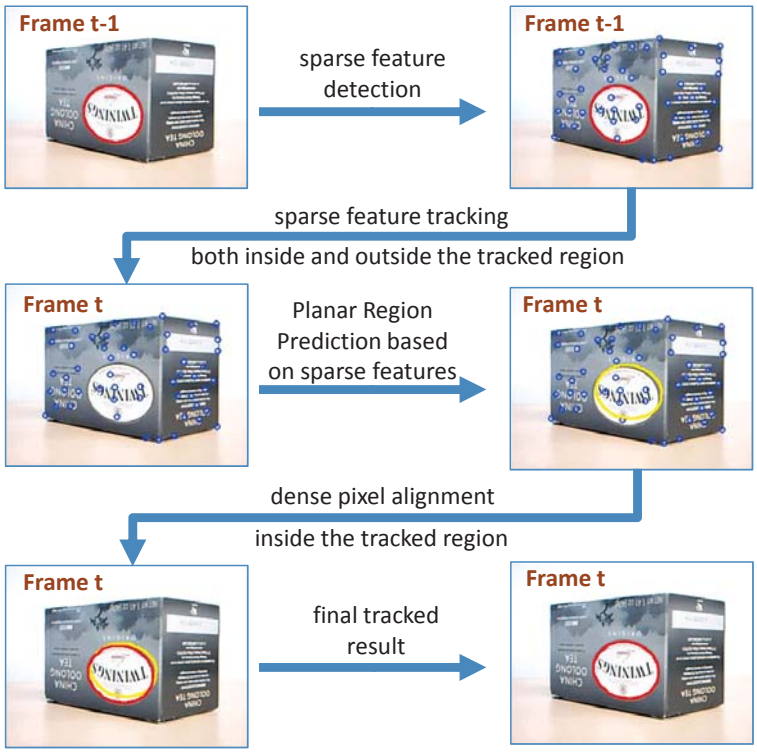


Figure 4.3: Planar region feature tracking with dense pixel alignment (best viewed in color). The red contour in the top left image shows the tracked planar region feature at time $t-1$. Sparse local point features are detected in frame $t-1$ (top right image) both inside and outside the tracked planar region. These sparse point features are tracked at time t (middle left image), and the homography transformation between the detected point features and the corresponding tracked point features are calculated. This homography transformation is used to predict the rough location of the planar region feature at time t (yellow contour in the middle right image). The roughly estimated location of the planar region feature is then refined by dense pixel alignment inside the tracked region (the refined location is shown as a new red contour in the bottom left image). The bottom right image shows the final tracked planar region feature at time t. Note that in our work the sparse feature tracking step is taken only between two adjacent frames, but we have intentionally enlarged the frame interval and tracking errors in this figure to clearly show the tracking scheme.

In the image sequence, frame 0 is referred to as the reference frame. Let f_0 be a detected planar region feature in the reference frame I_0, which is tracked as f_t in frame I_t. The corresponding transformation from f_0 to f_t is denoted by H_t.

Sparse Feature Tracking

At time $t-1$, the planar region feature is tracked as f_{t-1} with a homography transformation H_{t-1}. We detect salient point features in image I_{t-1} (note that the detection is on the entire image instead of just inside the planar region feature) and track these features in I_t using the KLT method [Shi and Tomasi, 1994]. The detected point features at time $t-1$ and the corresponding tracked features at time t are related by a homography transformation H_t^s (the superscript s denotes *sparse*). Given at least four pairs of matched point features, H_t^s is calculated through the Direct Linear Transformation (DLT) method [Hartley and Zisserman, 2003; Ma, 2004]. Thus from the following equation,

$$\hat{H}_t = H_t^s H_{t-1} \tag{4.6}$$

we get a rough estimate of the homography transformation $\hat{H}_t$ at time t.

Dense Pixel Alignment

We then adopt the Inverse Compositional algorithm [Baker and Matthews, 2004] with some modification for pixel-based image alignment.

Let $W(p;H)$ denote the parameterized set of warps, where $W(p;H)$ takes a pixel p in the coordinate frame of I_0 as input and maps it to another pixel (or sub-pixel) location in a new image under transformation H.

Given the roughly estimated matrix $\hat{H}_t$ in the sparse feature tracking step, we can warp image I_t back onto the coordinate frame of I_0 as $I_t(W(p;\hat{H}_t))$. Then we seek another matrix H_t^d which warps image I_0 as $I_0(W(p;H_t^d))$ such that the difference between $I_t(W(p;\hat{H}_t))$ and $I_0(W(p;H_t^d))$ is minimized over all pixels inside the planar region feature

f_0 in I_0. The difference is defined as

$$\Sigma_{p \in f_0}[I_0(W(p;H_t^d)) - I_t(W(p;\hat{H}_t))]^2 \tag{4.7}$$

where H_t^d (the superscript d denotes *dense*) is the refinement transformation.

It is easily seen that the final transformation from I_0 to I_t is

$$H_t = \hat{H}_t (H_t^d)^{-1}. \tag{4.8}$$

Combining Eq. 4.6 and 4.8, we have

$$H_t = H_t^s H_{t-1} (H_t^d)^{-1} \tag{4.9}$$

which is the final update equation from H_{t-1} to H_t.

The Inverse Compositional algorithm described in Baker and Matthews [2004] is based on only gray images. To better deal with illumination changes, we extend the algorithm to track planar regions based on gradient images as well as gray images, since gradient images are generally more robust to lighting condition changes.

Discussions

First, in the sparse feature tracking step, besides using information inside the planar region, we also use information outside that region. When there are no objects moving in the background environment, all sparse features in the input image have similar motion patterns to the planar region feature. Thus all these sparse features can be used to assist tracking the planar region feature, which avoids getting stuck at local minima and improves tracking stability. When there exist moving objects in the background environment, one can locate sparse features with similar motion patterns to the planar region feature using Generalized Hough Transform [Grabner et al., 2010]. Second, sparse features are tracked only between each two adjacent frames. That is, at each frame, new features are detected and tracked only

in the next frame using the KLT method. The KLT method works extremely well between adjacent frames, but may cause the feature drift problem across a long sequence of frames. Thus, we take advantage of the merit of the KLT method and in the meanwhile we avoid its drawbacks. Third, dense pixel alignment refines the planar region feature location, avoids the feature drift problem, and improves tracking accuracy.

4.3 Demos

4.3.1 Sparse Point Feature Tracking and Line Feature Integration

We test the scheme of sparse point feature tracking with line feature integration. All of the to-be-tracked objects have clear piecewise linear boundaries, where the boundaries can be either convex or non-convex. Some typical tracked frames from the videos are shown in Fig. 4.4.

To demonstrate the importance of integration of boundary information, we also test our tracking algorithm where the boundary correction step is disabled. That is, only point features are used. This test is done for two cases, (i) the same features are tracked over time, and (ii) features are detected at each frame and tracked only in the next frame. In case (i), the current boundary can be mapped from the reference boundary, by a homography transformation calculated based on the features in the current frame and the reference frame. In case (ii), the current boundary can be obtained the same way, except that the homography transformation has to be accumulated by transformations calculated from features between each pair of adjacent frames.

In either case (i) or (ii), tracking only point features works fine for the checker board, because it is highly-textured and the corner points are very salient. But for all the other videos, tracking only point features is obviously not sufficient. Some failed tracking frames are shown in Fig. 4.5. The primary reason for failures in case (i) is that features drift slowly away from the correct positions. The failures in case (ii) are caused primarily by

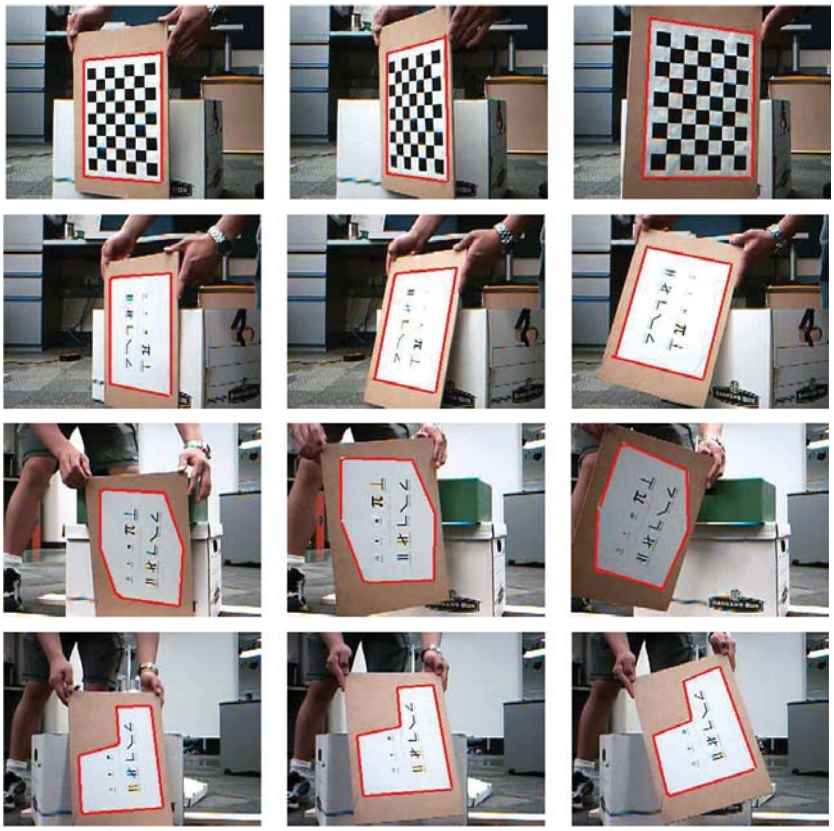


Figure 4.4: Tracking results with line feature integration. Results are shown for a checker board, a rectangular letter board, a hexagonal letter board and a concave letter board. Tracked features are shown with red contours. Tracking is based on both point features and line features.

accumulated parameter estimation errors. It's also worth noticing in case (i) some features may be lost in a long sequence of images.

4.3.2 Sparse Point Feature Tracking and Dense Pixel Alignment

In Fig. 4.6, we show tracking results for a dataset with large translation, rotation, scale, and illumination changes, based on our tracking method (sparse point feature tracking and dense pixel alignment).

In our tracking scheme, the sparse feature tracking step plays an important role.

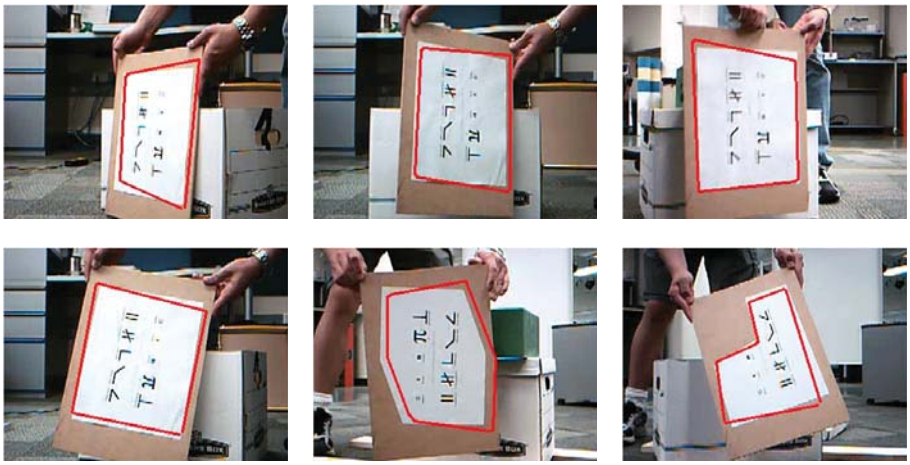


Figure 4.5: Tracking failures without line feature integration. Some failure examples are shown when the boundary correction step is disabled. The failures are caused primarily by either accumulated feature position error or accumulated parameter estimation error.

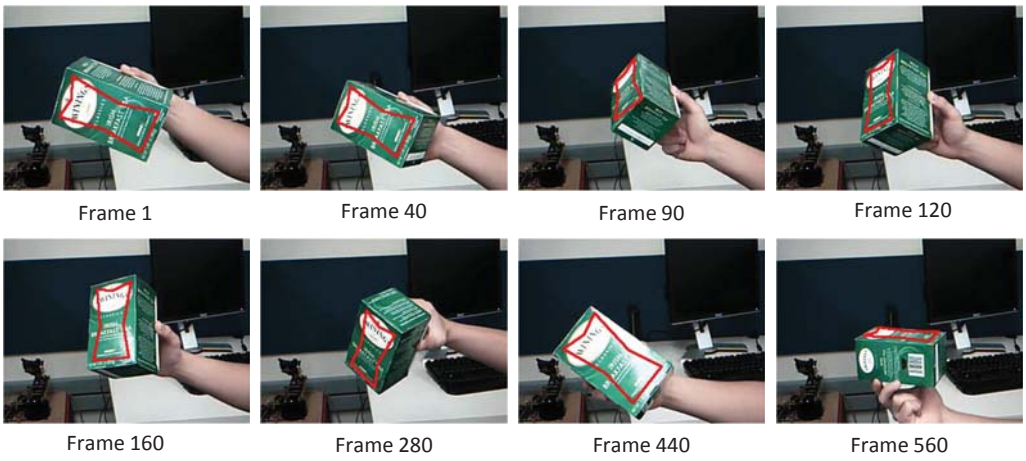


Figure 4.6: Tracking results for a dataset with large translation, rotation, scale, and illumination changes.

Compared to the traditional Inverse Compositional approach [Baker and Matthews, 2004], sparse feature tracking significantly improves the overall tracking robustness. Fig. 4.7 shows some tracking examples with or without the sparse feature tracking step.

Thanks to the sparse feature tracking step providing a transformation matrix $\hat{H}_t$ which is already very close to the optimal solution H_t, the minimization of Eq. 4.7 usually

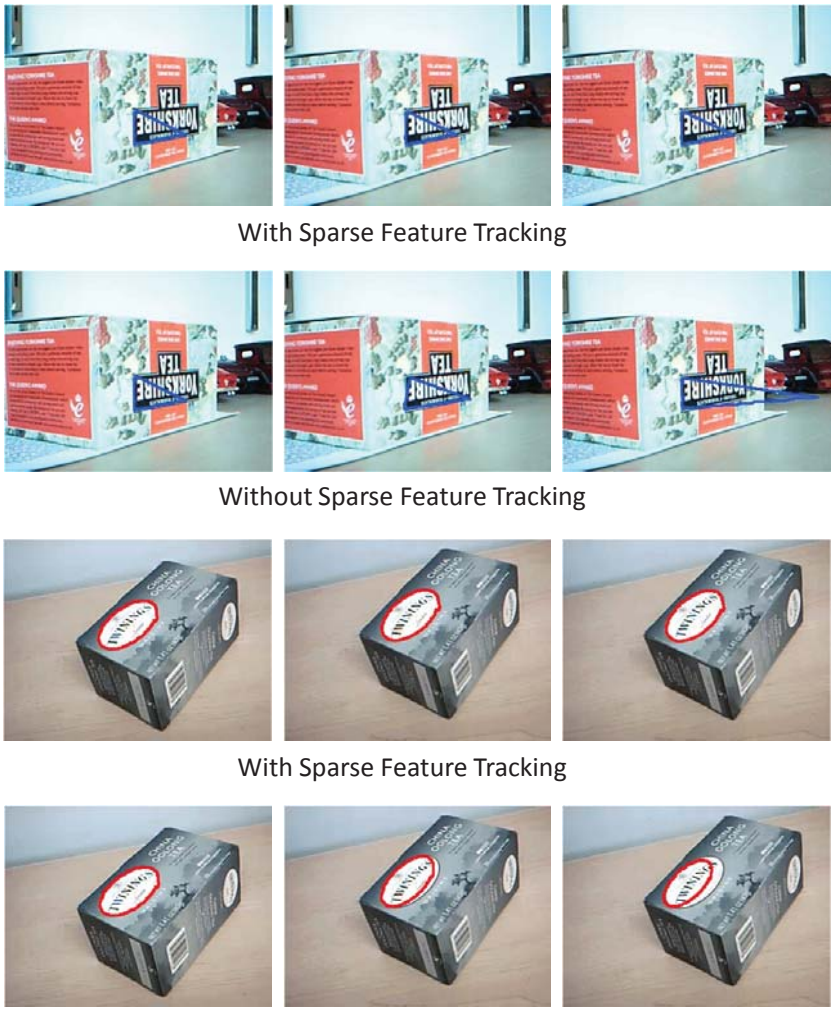


Figure 4.7: Tracking results using dense pixel alignment with or without sparse feature tracking. The odd-numbered rows show some examples using both sparse feature tracking and dense pixel alignment. As a comparison, the even-numbered rows show the corresponding results for dense pixel alignment without the sparse feature tracking step, where the pixel alignment process does not converge to the correct location.

needs only a few (less than 5) iterations to obtain the refinement transformation H_t^d.

The Inverse Compositional algorithm in the dense pixel alignment step requires that there is a good overlap for the tracked region between the current frame and the next

frame. If the planar region feature is small, the requirement may not be well satisfied and the Inverse Compositional algorithm may get stuck at local minima. But the sparse feature tracking step provides a rough estimate of the location of the planar region feature, and guarantees a good start point for the Inverse Compositional algorithm to converge to the correct location.

Chapter 5

2D Object Segmentation

To construct object models in the OSH, a fundamental problem is to separate out the object of interest in the pixel-level sensory input from the background environment. Motion information is an important cue for an intelligent robot to perform the separation between moving foreground objects and the static background.

5.1 Introduction

5.1.1 Existing Works

There have been many approaches developed for foreground object segmentation based on motion information. Among them, background subtraction is a standard mechanism for a specific environment, typically with stationary cameras. Statistical pixel-level background models such as Pfinder [Wren et al., 1997], non-parametric model [Elgammal et al., 2000], Gaussian Mixture Model [Stauffer and Grimson, 1999, 2000], and their variations [Zhong and Sclaroff, 2003; Mittal and Paragios, 2004; Monnet et al., 2008; Ko et al., 2010; McKenna et al., 2000] have achieved many successful applications in visual surveillance; nevertheless the strong assumption on fixed field of view prevents their applications to dynamic cameras mounted on robots. Moreover, in the robotic application of object manipu-

lation, the close-up view on objects may result in a large portion of the field of view taken up by the foreground. So pixel-level background models can completely fail because the most frequently observed intensity/color values may come from foreground pixels. To further relax this assumption, many ego-motion compensation [Hayman and Eklundh, 2003; Mittal and Huttenlocher, 2000] or image mosaic [Brown and Lowe, 2003; Szeliski, 2006; Azzari et al., 2005] methods have been presented for background modeling. Nevertheless, they may result in blurred edges due to accumulated image registration error, and their scope is restricted to scenes where the background can be well approximated by a plane.

To handle dynamic cameras, motion analysis is applied to either sparse features or dense optical flows. Dense optical flows estimated on corners and edges are used to generate a motion vector at every pixel. Then motion layers are extracted from the flows [Black and Anandan, 1996; Xiao and Shah, 2005; Ren and Gu, 2010]. Motion segmentation can also be achieved through tracking image features and clustering them into foreground/background by estimating image transformations or motion trajectories for different motion patterns. In general, a set of affine/homography parameters [Ren and Gu, 2010; Han et al., 2006; Sivic et al., 2006] or trajectory parameters [Sheikh et al., 2009] need to be estimated by iterative linear regression [Han et al., 2006] or RANSAC [Sheikh et al., 2009; Ren and Gu, 2010]. These approaches obtain good performance for moving object segmentation, though they may be computationally expensive.

5.1.2 Our Method

In building the OSH, the first step is to identify invariants for the static background, where the static background model is coupled with the robot pose. The robot pose is dependent on motor signals, so we learn the relation between motor signals and visual background motion as part of the invariants in the static background model. This learned relation separates out the background and facilitates the construction of the foreground models.

We observe that previous approaches ignore a fundamental cause of the background

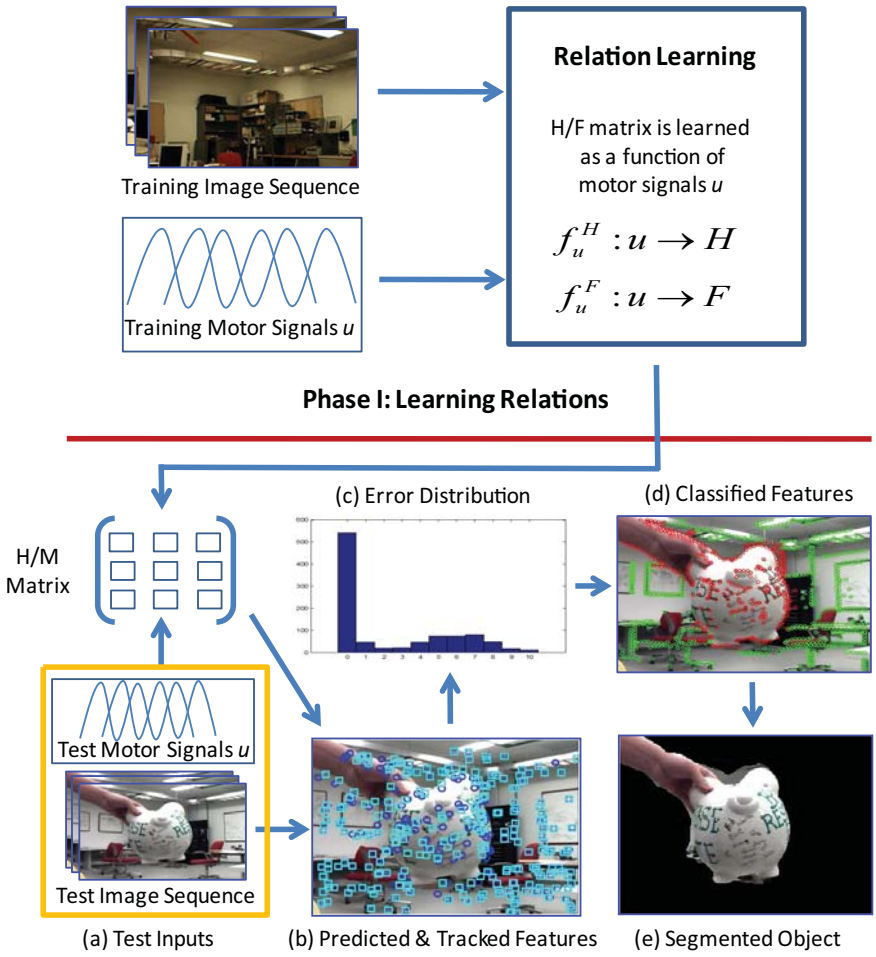


Figure 5.1: Overall framework for motor signal based motion segmentation. There are two phases: off-line one-time learning and online object segmentation. In phase I, the homography (H) and fundamental (F) matrices are learned as functions of motor signal changes, which is a one-time process. In phase II, given motor signals (a1), the H/F matrix is calculated based on the learned functions from phase I and is used to predict feature locations (b, blue circles). Meanwhile, features are tracked (b, cyan squares) across frames (a2). Then the errors between the predicted and tracked locations are computed. Its distribution (c) shows two modes that correspond to background and foreground features respectively. The feature clustering is done by EM (d, background in green and foreground in red). Finally, the object segmentation is obtained by propagating sparse feature labels to every pixel (e).

motion when performing motion analysis. No approach has exploited the relationship between the robot's motor signals and the background motion. To make up this gap, we propose an automatic system for moving object segmentation using the robot's motor signals, as well as the information from the image stream. This is feasible since the acquisition of motor signals is very handy for robotic applications like object manipulation.

The idea of employing motor signals for moving object segmentation is motivated by the human visual system. The human visual system does not rely only upon information from the retina to perceive object motion, because identical retinal stimulations can be evoked by the movement of objects as well as by self-evoked eye movements [Galletti and Fattori, 2003] or head/body movements. By considering motor signals of a robot corresponding to the signals for eye, head, and/or body movements of humans, we can predict the motion patterns of background features using the motor signals. In contrast, the real motion patterns of foreground features will be different from their predictions using motor signals as they have independent motions from the robot. This is because the background motion has stronger correlation to motor signals than the foreground object motion. This observation provides a way to separate the image features into background and foreground (as illustrated in Fig. 5.1 (c), the two error modes correspond to background and foreground) based on their discrepancy with the predictions.

As Fig. 5.1 illustrates, the proposed framework starts with learning the relation between motor signal change and background motion. As we know, motor signal change of a robot leads to visual change in its input images. The visual change is constrained by a homography matrix in the case where the robot's camera has only rotation but no (or small) translation (e.g., a pan tilt camera), or by a fundamental matrix when the camera has large translation [Hartley and Zisserman, 2003]. Hence, our task is to learn the relation from the motor signals to the homography/fundamental matrices. Notice that our learning process is fully *automatic* and *one-time*. Unlike some previous works [Hayman and Eklundh, 2003; Mittal and Huttenlocher, 2000; Brown and Lowe, 2003; Szeliski, 2006; Azzari et al.,

2005], we do not assume that the background can be well-approximated by a plane due to incorporation of the fundamental matrix constraint (as opposed to using only the homography matrix constraint). In addition, we do not need to build a pixel-level background model, so our method does not suffer from the blurry edge problem.

We would like to point out that learning the relation between motor signal changes and homography/fundamental matrices is different from the traditional camera alignment or calibration to certain robotic platform (such as [Knight and Reid, 2006]), where the alignment/calibration is carried out in 3D space. However, our method for moving object segmentation works completely in the 2D image space, and does not involve any reasoning or computation in 3D space.

With the learned relation, given any motor signal change, we compute the corresponding homography/fundamental matrices, which are used to predict the new locations of the features (i.e., sparse corner and sampled edge features). The errors between the predicted feature locations and their actual tracked locations are clustered to separate foreground features from background features. In the ideal case, the errors for the background features will be zero, and we can simply group those features with zero error as background and others as foreground. However, due to noisy feature tracking results, the background feature errors will typically not be zero and it is difficult to pre-select a constant threshold to separate the features. Therefore, we automatically determine the threshold on-line by fitting Gaussian mixture models using the Expectation-Maximization algorithm.

Our approach can be categorized as motion analysis by feature clustering like [Han et al., 2006; Sivic et al., 2006; Ren and Gu, 2010; Sheikh et al., 2009]. But unlike them, thanks to the use of motor signals, our feature clustering is conducted in a one-dimensional space. Moreover, our segmentation can be completed in fewer frames (i.e., 3 to 5 frames as compared to about 30 frames in [Sheikh et al., 2009]). Our approach is also similar to [Ren and Gu, 2010] which attempts to segment hand-held objects by training a linear classifier using groundtruth segmentations. Our method is fully automatic and self-supervised,

and does not need groundtruth segmentations, when we learn the homography and fundamental matrices from motor signals.

After the sparse features are labeled, we further apply the Active Contours [Kass et al., 1988; Isard and Blake, 1998] and Graph-based Transduction methods to segment the dense foreground. By treating the labeled sparse features as training samples and the unlabeled pixels as test data, we solve the foreground segmentation problem by transductive learning [Joachims, 2003; Dhillon, 2001]. In our work, we use the modified Normalized Graph Cuts [Shi and Malik, 2000; Joachims, 2003] as the transductive classifier.

To summarize, we present a novel system for moving object segmentation using motor signals [Xu et al., 2011]. Our major contributions are as follows:

- To the best of our knowledge, we are the first to use motor signals for motion segmentation by clustering feature prediction errors in one dimensional space.
- The threshold for background/foreground separation in the error space is automatically determined using Expectation-Maximization.
- We learn homography and fundamental matrices from motor signals, which allows us to deal with both the situation where the camera has only rotation and no or small translation, and the situation where the camera has significant translation.
- Graph-based Transduction techniques are applied to smoothly and robustly separate out the moving objects.

5.2 Relation Learning from Motor Signal Changes to Visual Changes

We learn two types of relation between motor signals and motion patterns of background features: *homography* and *fundamental* matrices. When the camera translation is very small or the environment is planar, two images of the same scene can be well related by a homography matrix; when the camera translation is large, a fundamental matrix can be used to model the relation between two images [Hartley and Zisserman, 2003].

Figure 5.2: Robots used for moving object segmentation. Left image: a webcam on a pan tilt unit, right image: a webcam on a mobile robot (the webcam is a built-in device at the center of the top edge of the laptop monitor).

We use the term *motor signals* to refer to feedback signals that are directly related to the robot's actions. Specifically in this work, the robot's actions we consider are various motions such as head turning and/or body moving.

Fig. 5.2 shows two typical robot hardware settings, where the first robot looks around the world with its position fixed, and the second moves around in the world. The motor signals for the first robot are camera pan and tilt change, $u = \{\phi, \psi\}$. For the second robot, because no odometry signals are available in the current robot setup, we instead take the robot's location and orientation change as indirect motor signals. The location and orientation change, $u = \{x, y, \theta\}$, are obtained from a laser-based SLAM method [Beeson et al., 2010].

Although the experimental setups in Fig. 5.2 are specific, the reasoning of this work is very general and can be easily extended to other setups, for example, with higher dimensional motor signals.

Let H or F be the 3×3 homography or fundamental matrix between two frames. Our goal is to learn a mapping function $f = f_u^H$ or $f = f_u^F$ from motor signal change u to visual change H or F. The mapping function f may be constructed by hand based on the physical spatial setup of the camera on the robot body. However, manual construction may include large systematic errors, for example, if the camera's principle axis (when the robot

is at its initial position) is not exactly parallel to the standard horizontal plane. In addition, manual reconstruction requires acquisition of camera intrinsic parameters through camera calibration.

To avoid camera calibration, systematic errors, and complexity of manual construction, we learn the function f automatically in our system. Due to the fact that f is an invariant and hence independent of the environment, we learn it in an environment which has good textures in order for different frames of images to be well registered and has no objects moving in it. The robot collects a set of images plus the corresponding motor signals, under various motion patterns. The relation between images is obtained as homography/fundamental matrices through image feature tracking. The robot then takes the motor signals and corresponding homography/fundamental matrices as input and learns the relation f without human intervention.

In the learning process, the space of the motor signal change u is sampled such that the minimum and maximum possible changes (within a predefined time interval) are included. Once the relation between the motor signal changes and the homography/fundamental matrices is learned in one environment, it is repeatable in any other environment, because the homography/fundamental matrices are determined (up to a global scale factor which is discussed in Section 5.2.1 and 5.2.2) by the motor signal changes (which lead to rotation and translation in the physical world) [Ma, 2004], hence are not dependent on the environment.

The selection of whether the homography or fundamental matrix case applies can be made online based on the robot/camera translation (regardless of its rotation). Zero or small translation corresponds to the homography matrix case, and large translation corresponds to the fundamental matrix case [Ma, 2004].

5.2.1 Homography Matrix Case

The 3×3 homography matrix between two images can be calculated from the tracked KLT features [Shi and Tomasi, 1994]. Since H has 8 degrees of freedom, it can only be obtained up to a scale factor from at least four pairs of corresponding points [Hartley and Zisserman, 2003]. So we need to normalize H in order to learn a continuous function between the motor signal change, $u = \{\phi, \psi\}$, and the visual change, H. When there is no or very small translation, the homography matrix is equivalent or close to a pure rotation matrix multiplied by a scale factor. Across a small number of frames, the camera pan and tilt do not have large changes, and the last element of the rotation matrix will be guaranteed to be non-zero. Thus we normalize H such that its last element is always 1.

Each mapping relation (which is a non-linear function) from the motor signal change to an element in the homography matrix is fitted as a polynomial. In our experiments, the fitting error becomes very small when the degree of the polynomials grows to three. Let V^H denote the stacked 8-dimensional row vector of H. The 10-dimensional motor signal vector V^u is defined as

$$V^u = [\phi^3, \psi^3, \phi^2\psi, \phi\psi^2, \phi^2, \psi^2, \phi\psi, \phi, \psi, 1]. \tag{5.1}$$

For each two frames that are captured within a certain number of time steps, we obtain a pair of V^u and V^H. Suppose we have a number of n pairs of these vectors, denoted by $\{V_k^u, V_k^H\}$ $(k = 1, ..., n)$. We stack all V_k^u as rows in a $n \times 10$ matrix A^u, and stack all V_k^H as rows in a $n \times 8$ matrix B^H. Then the relation function f_u^H between motor signals and homography matrices is learned as third order bivariate polynomials from the following equation,

$$A^u f_u^H = B^H \tag{5.2}$$

where f_u^H is a 10×8 matrix.

5.2.2 Fundamental Matrix Case

The fundamental matrix between two images can be calculated from at least 8 pairs of corresponding points by solving a linear equation [Hartley and Zisserman, 2003]. Similar to the homography matrix case, the fundamental matrix is also obtained up to a scale factor. However, the fundamental matrix cannot be normalized by dividing the last element any more, since there is no guarantee that the last element is non-zero. A fundamental matrix F has a singular value decomposition [Ma, 2004] in form of $F = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}$, where $\Sigma = diag\{\sigma_1, \sigma_2, 0\}$ ($\sigma_1, \sigma_2 > 0$) is a diagonal matrix, and $\mathbf{U}$ and $\mathbf{V}$ are rotation matrices. Based on this decomposition, we normalize a fundamental matrix by dividing it by $(\sigma_1 + \sigma_2)/2$.

Each element in the fundamental matrix is also fitted as a polynomial of the motor signal change. In our experiments, polynomials of order three give us small fitting errors. Let V^F denote the stacked 9-dimensional row vector of F. The 20-dimensional motor signal vector V^u is defined as

$$\begin{aligned} V^u = [&x^3, y^3, \theta^3, x^2y, x^2\theta, xy^2, xy\theta, x\theta^2, y^2\theta, \\ &y\theta^2, x^2, y^2, \theta^2, xy, x\theta, y\theta, x, y, \theta, 1]. \end{aligned} \tag{5.3}$$

Suppose we have a number of m pairs of these vectors, denoted by $\{V_k^u, V_k^F\}$ ($k = 1, ..., m$), from m pairs of frames. We stack all V_k^u as rows in a $m \times 20$ matrix A^u, and stack all V_k^F as rows in a $m \times 9$ matrix B^F. Then the relation function f_u^F between motor signals and fundamental matrices is learned as third order trivariate polynomials,

$$A^u f_u^F = B^F \tag{5.4}$$

where f_u^F is a 20×9 matrix.

5.3 Sparse Feature Classification

For image I_t at time t, we detect two types of features: corners and edges. The locations of the corners and sampled edge points form our sparse feature set P_t. These sparse features are tracked in I_t's neighboring frames I_{t+k} $(k = \{-M, ..., -1, 1, ..., M\})$. The tracked features in frame I_{t+k} are denoted as P_{t+k}.

Given the motor signals at two frames t and $t+k$, the homography/fundamental matrix between the two frames is calculated from

$$\begin{aligned} V_k^H &= V_k^u f_u^H \\ V_k^F &= V_k^u f_u^F \end{aligned} \tag{5.5}$$

where V_k^H and V_k^F can be unstacked to get the homography matrix H_k and the fundamental matrix F_k respectively.

From frame I_t to I_{t+k}, the background features should be consistent with the transformation H_k or F_k, while the foreground features will violate this transformation. Thus we can classify the features based on the errors between the actual tracked feature locations and their estimated locations predicted from H_k or F_k.

5.3.1 Homography Matrix Case

For each background feature $P_{i,t+k}$ in I_{t+k} tracked from $P_{i,t}$ in I_t, they are related by $P_{i,t+k} \propto H_k P_{i,t}$. We define the error term as

$$d_{i,t+k} = \|\hat{P}_{i,t+k} - P_{i,t+k}\| \tag{5.6}$$

where $\hat{P}_{i,t+k} \propto H_k P_{i,t}$ and the last element in $\hat{P}_{i,t+k}$ is normalized to 1 s.t. the error is measured in the image space.

5.3.2 Fundamental Matrix Case

In the fundamental matrix case, the background features $P_{i,t}$ and $P_{i,t+k}$ are related by $P_{i,t+k}^T F_k P_{i,t} = 0$ [Hartley and Zisserman, 2003]. We define the error term as

$$d_{i,t+k} = |P_{i,t+k}^T F_k P_{i,t}| / \gamma \tag{5.7}$$

where the error is the distance from the point $P_{i,t+k}$ to the epipolar line $F_k P_{i,t}$ corresponding to $P_{i,t}$. Here γ is a normalization term such that the error is measured in the image space, and it is the root-sum-square of the first two elements in the 3D vector $F_k P_{i,t}$.

We then cluster the tracked features based on the error set $\{d_{i,t+k}\}$ $(i = 1,2,...,N_p)$. Note that this clustering process is taken in only one dimensional space. To avoid distractions from incorrectly tracked features which may produce unexpected large errors, we assign $\{d_{i,t+k}\}$ a maximum limit (10 pixels in our experiments). Due to inaccurate parameter estimation in f_u^H and f_u^F and noisy feature tracking results, it is difficult to pre-determine a threshold to divide $\{d_{i,t+k}\}$ into two groups. We use the Expectation-Maximization algorithm to fit a two-component Gaussian mixture model (corresponding to background/foreground) on $\{d_{i,t+k}\}$. The model is described by

$$G_{t+k}(x) = \sum_{j=\{bg,fg\}} w_{t+k}^j g(x; \mu_{t+k}^j, \sigma_{t+k}^j) \tag{5.8}$$

where $g(.)$ is the normal distribution, and $w_{t+k}^{bg} + w_{t+k}^{fg} = 1$. Here the superscripts bg and fg correspond to background and foreground respectively. At each frame t, the two Gaussian components are initialized with the Gaussians estimated in frame $t-1$. Those features with a high average of likelihood from Eq. 5.8 across frames I_{t+k} $(k = \{-M,...,-1,1,...,M\})$ are classified as background features and others as foreground features in frame I_t. In our experiments, we set $M = 3$ for the homography matrix case and $M = 5$ for the fundamental matrix case.

5.4 Dense Foreground Segmentation

After sparse features have been classified, we propagate their labels to all image pixels to achieve a dense foreground segmentation. Given a set of labeled foreground and background features $P = \{(x_1, l_1), \ldots, (x_{N_p}, l_{N_p})\}$ (N_p is the number of features in P), where x_i is a pixel feature vector (consisting of HSV color and 2D location) and $l_i \in \{+1, -1\}$ is the foreground/background label, our goal is to classify the remaining unlabeled pixels $U = \{x_{N_p+1}, \ldots, x_{N_p+N_u}\}$ into either background or foreground, where N_u is the number of unlabeled pixels. We apply two approaches to achieve this goal: Active Contours and Graph-Based Transduction.

5.4.1 Active Contours

Given sparse foreground features $P^{fg} \subset P$, we filter out outliers by agglomerative clustering, where the largest cluster is preserved as the final foreground features, since we assume there is one moving object in each dataset. We then find the convex hull for the foreground features, initialize an active contour model with the convex hull, and fit the active contour model to image edges. The active contour model uses piecewise splines to represent objects, and fits the splines to object boundaries by minimizing a sum of two energy terms: the Internal Energy and External Energy. The Internal Energy accounts for boundary smoothness, and the External Energy evolves the model to fit with observed image edges (see [Isard and Blake, 1998; Kass et al., 1988] for details). This method is very efficient in computation and works well for many applications. However, since our active contour model is initialized with a convex hull, it may never converge to perfect object boundaries when the object shapes are non-convex. In addition, the weights for the energy terms in the active contour model are hard to be tuned. As a result, small non-boundary edges around the object boundaries can cause serious distractions. Hence, we further propose a graph-based transductive learning approach to classify the pixels in U.

5.4.2 Graph-based Transduction

We treat labeled pixels P as training data and unlabeled pixels U as test data. Then we formulate the foreground segmentation problem as a binary classification problem via transductive learning. We aim at finding a transductive classifier $f(x) \in \{+1, -1\}$ over the feature space to classify the test data. The advantage of transductive learning is that one can explore both the training and test data structure when training the transductive classifier. In our work, we choose graph as a tool to solve the problem.

Let us define a graph with P and U as vertices and adjacent weight matrix W. Each entry $w(x_i, x_j)$ of W is defined by a Gaussian kernel $\mathscr{K}(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$. We seek a function $f(x)$ that projects the graph vertices onto $\{+1, -1\}$ such that we have low training error on P and precise label assignments (clustering) on $P + U$. The objective function [Joachims, 2003; Dhillon, 2001] is formulated as

$$\begin{aligned} &\min_{\mathbf{f}} \mathbf{f}^T L\mathbf{f} + \lambda(\mathbf{f} - \mathbf{b})^T C(\mathbf{f} - \mathbf{b}), \\ &\text{subject to} \quad \mathbf{f}^T \mathbf{1} = 0 \text{ and } \mathbf{f}^T \mathbf{f} = n \end{aligned} \tag{5.9}$$

where n is the pixel number of an image, $\mathbf{b} \in R^n$ with each dimension $\mathbf{b}(i) = \sqrt[2]{(n_-/n_+)}$ for positive labeled data and $\mathbf{b}(i) = -\sqrt[2]{(n_+/n_-)}$ for negative data (n_+ and n_- are the numbers of positive and negative labeled data), Laplacian matrix $L = D - W$ with $D_{ii} = \sum_{x_j} w(x_i, x_j)$, and C is a diagonal matrix assigning penalty to any misclassification of the training examples. The first term measures the discontinuity of the graph bi-partition and the second term computes the training errors on the labeled data. The parameter λ controls the tradeoff between training error and clustering quality. We adopt the Spectral Graph Transducer [Joachims, 2003; Shi and Malik, 2000] as our transductive classifier.

5.5 Experimental Results

In this section, the following abbreviations are used: MSMS (motor signal based motion segmentation), GMM (Gaussian Mixture Model), HM (homography matrix case), FM (fundamental matrix case), AC (Active Contours), and GBT (Graph-based Transduction).

5.5.1 Relation Learning

We first separately learned the homography and fundamental matrices as functions of motor signal changes, as shown in Fig. 5.1. The detailed process is described in section 5.2.

HM Case

For the PTU robot shown in Fig. 5.2, in the 2D motor signal space $\{\phi, \psi\}$, we drew 32 evenly distributed rays shooting out from the point $(\phi, \psi) = (0,0)$. On each ray, we selected 16 evenly spaced points including the point $(0,0)$. Thus we had $32 \times 15 + 1$ different points, and at each such point we collected an image. The transformations between close images were obtained by tracked KLT features. Fig. 5.3 shows some typical images for the environment used for relation learning in the HM case.

Each element in the homography matrix is fitted as a polynomial of the motor vectors. When the order of the polynomial grows to 3, the fitting error becomes very small. The fitting error is measured as the average difference between each calculated homography matrix based on tracked features and the corresponding predicted homography matrix based on motor signals and the learned relation. The error for the HM case in our experiments is

$$error^H = \begin{pmatrix} 0.00004592 & 0.00005175 & 0.00625605 \\ 0.00003530 & 0.00006063 & 0.00586136 \\ 0.00000034 & 0.00000045 & 0 \end{pmatrix}. \tag{5.10}$$

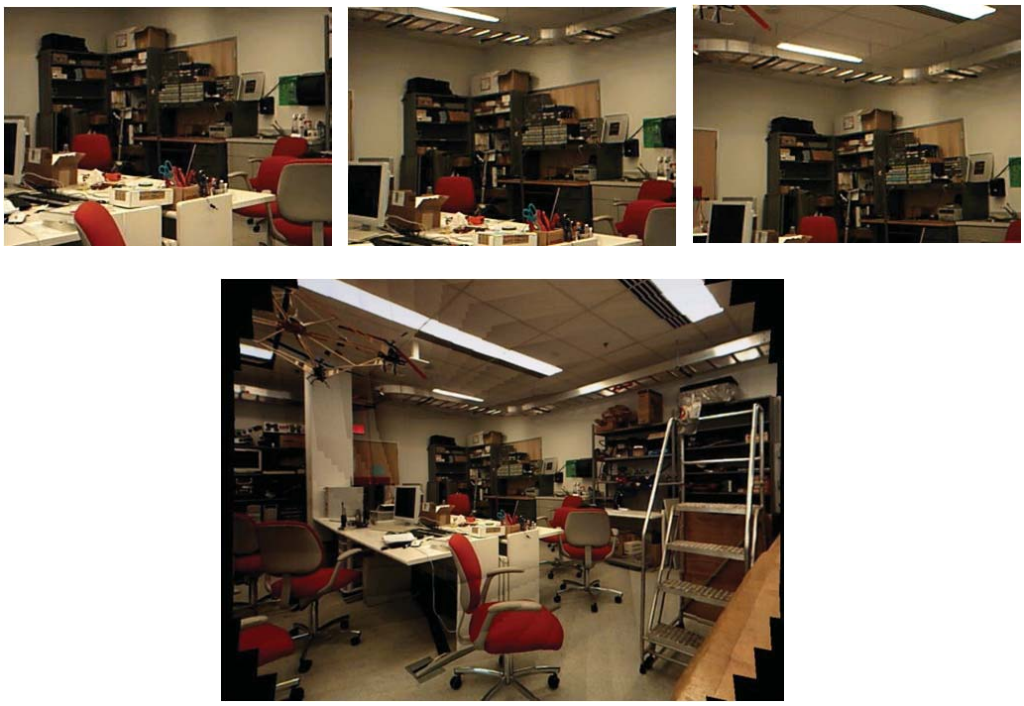


Figure 5.3: Environment for relation learning in the HM case. The top row shows original images captured in the environment. The bottom row shows the synthesized image of the environment from the original images based on known motor signals.

FM Case

In the FM case, for the mobile robot shown in Fig. 5.2, the motor signals $u = \{x, y, \theta\}$ were obtained from a laser-based SLAM method. We collected images and motor signals whenever the motor signal was updated while the robot was moving. Compared to the HM case, the motor signals in the FM case are less accurate due to the complex robot setup and the estimation errors from laser data. We collected four groups of data in the same environment, where the robot moves forward on a line, about 1500 images in total were recorded, and the final f_u^F was averaged over the results from all groups. Fig. 5.4 shows some typical images for the environment used for relation learning in the FM case.

Similar to the HM case, each element in the homography matrix for the FM case is

also fitted as a polynomial of the motor vectors. When the order of the polynomial grows to 3, the fitting error becomes very small. Similarly, the fitting error is measured as the average difference between each calculated fundamental matrix based on tracked features and the corresponding predicted fundamental matrix based on motor signals and the learned relation. The error for the FM case in our experiments is

$$error^F = \begin{pmatrix} 0.00000013 & 0.00003398 & 0.00352693 \\ 0.00003402 & 0.00000026 & 0.00451349 \\ 0.00349927 & 0.00447559 & 0.00454410 \end{pmatrix}. \quad (5.11)$$

Figure 5.4: Environment for relation learning in the FM case.

5.5.2 Datasets and Comparison Baselines

The learned relation functions are then used for segmenting moving objects in videos taken from any environment different to that used in the learning.

Datasets. We collected eight test videos to quantitatively evaluate our system on moving object segmentation (two for static camera case, three for pan-tilt camera case, and the remaining three for free-moving camera case).

These videos have a close-up view on various hand-held objects, as well as the corresponding motor signals. For all videos, the foreground objects in sampled frames are manually labeled as the ground truth. As far as we know, there is no similar dataset publicly

available. Though Ren *et. al* [Ren and Gu, 2010] also attempt to segment hand-held objects, their datasets do not have motor signals. We also collected two additional far-view videos on the moving objects to show our system works well for this situation as well.

Comparison Baselines. For the static camera case, we use GMM-based background subtraction as the baseline approach. For the pan-tilt and free-moving camera cases, RANSAC-based homography or fundamental matrix fitting [Ren and Gu, 2010; Uemura et al., 2008; Deniz et al., 2010; Xiao and Shah, 2005; Goshen and Shimshoni, 2008; Han et al., 2006] is popularly used for motion segmentation, and is chosen as the comparison baseline in our experiments to test the performance of our method.

The segmentation accuracy is defined as A_I/A_U, a common evaluation criterion [Gulshan et al., 2010; Sivic et al., 2008], where A_I (intersection area) is the number of pixels that are labeled as foreground in both the segmentation result and the ground truth, and A_U (union area) is the number of pixels that are labeled as foreground in either the segmentation result or the ground truth.

5.5.3 Static Camera Case

First we compare the performance of MSMS (our method) with GMM (baseline) when the camera is static. We use the GMM implementation [KaewTraKulPong and Bowden, 2001] in OpenCV. Since the camera is static, there is no motor signal change and the homography transformation H remains constant as an identity matrix. Thus the step of learning f is excluded in the system. We apply only the sparse feature classification and dense pixel segmentation steps, and test the system on a "book" dataset and a "hard-drive box" dataset. Both datasets have significant illumination changes because the webcam's light auto-adjust function is enabled. The light auto-adjust function usually takes effect when the object moves from very close to the camera to far away, or vice versa. The ground truth is obtained by manually labeling the foreground boundaries in the images.

Fig. 5.5 illustrates some visual results for qualitative comparison on the "book" and

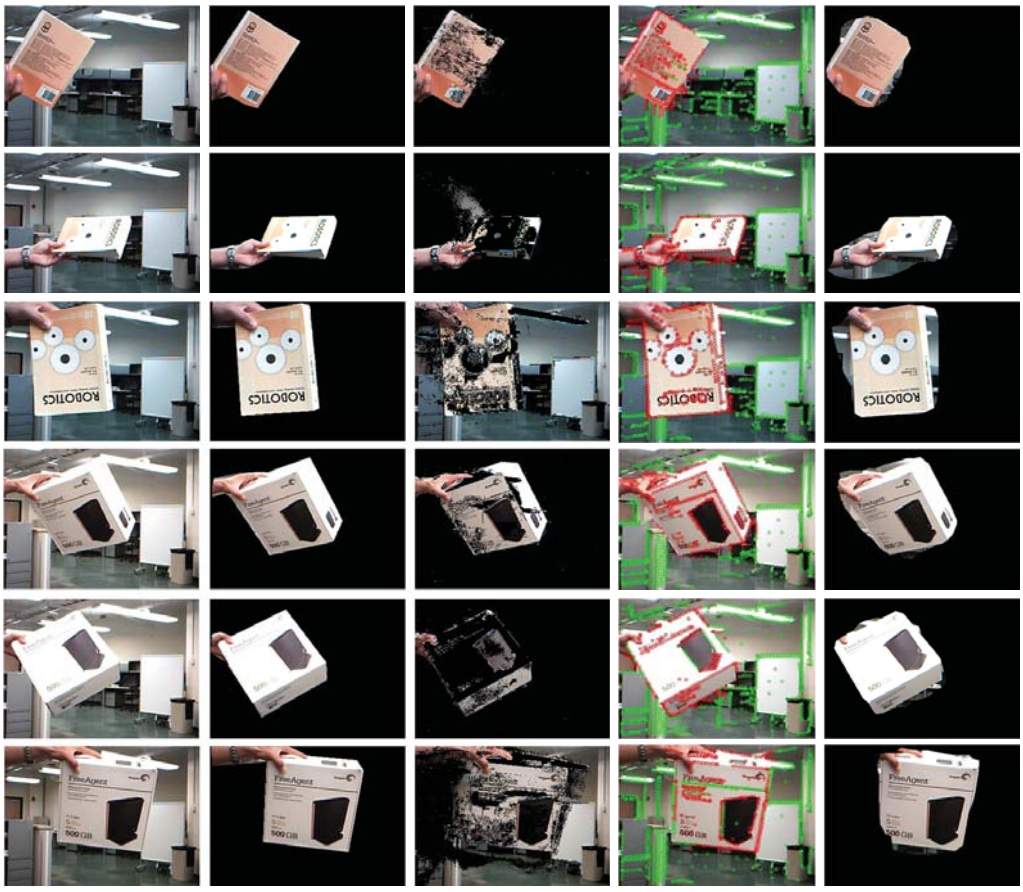
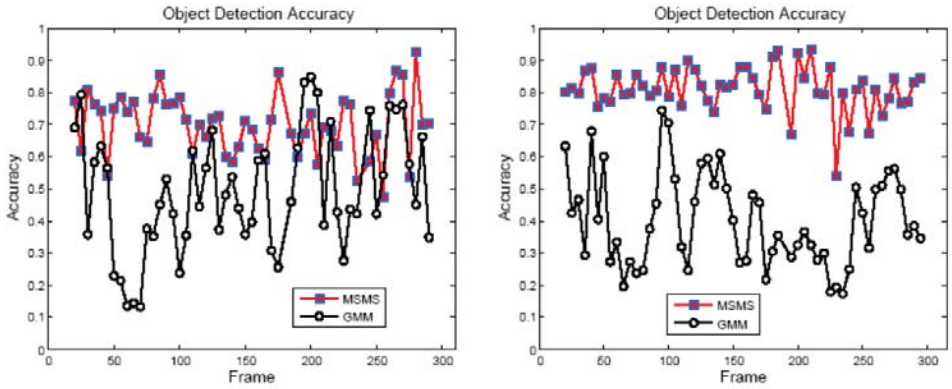


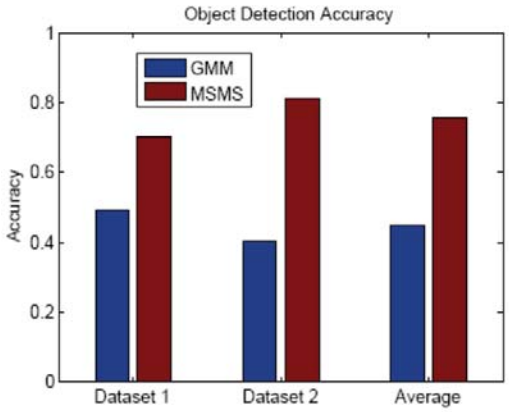
Figure 5.5: Detection results for the static camera case (best viewed in pdf). The columns show the original images, ground truth, detection results by GMM, classified sparse features (green and red represent background and foreground respectively) by MSMS, and segmentation results by MSMS-AC. The GMM method tends to produce black holes on the foreground objects and spreading misclassified pixels over the whole image when illumination condition changes. In contrast, the MSMS method gives better results in these situations.

"hard-drive box" datasets, and Fig. 5.6 (a) shows the average foreground detection accuracy. It clearly shows that MSMS is superior to GMM (with more than 20% improvement). This is because the GMM method often produces a large number of noisy pixels spreading over the whole image when the illumination changes due to objects moving close to or far away



Dataset 1: Book

Dataset 2: Hard-drive Box



Overall Accuracy

Figure 5.6: Quantitative evaluation results for static camera case.

from the camera and reflections on the objects. Moreover, GMM needs sufficient frames to learn a stable pixel-level background model. In contrast, the proposed MSMS method is more robust to illumination changes, and more importantly it only needs a few frames to detect foreground and background sparse features.

5.5.4 Pan-Tilt Camera Case

We run our system on three hand-held objects: “tea-box”, “football”, and “toy-pig”, taken from a pan-tilt camera. The camera has significant orientation change in the videos, and

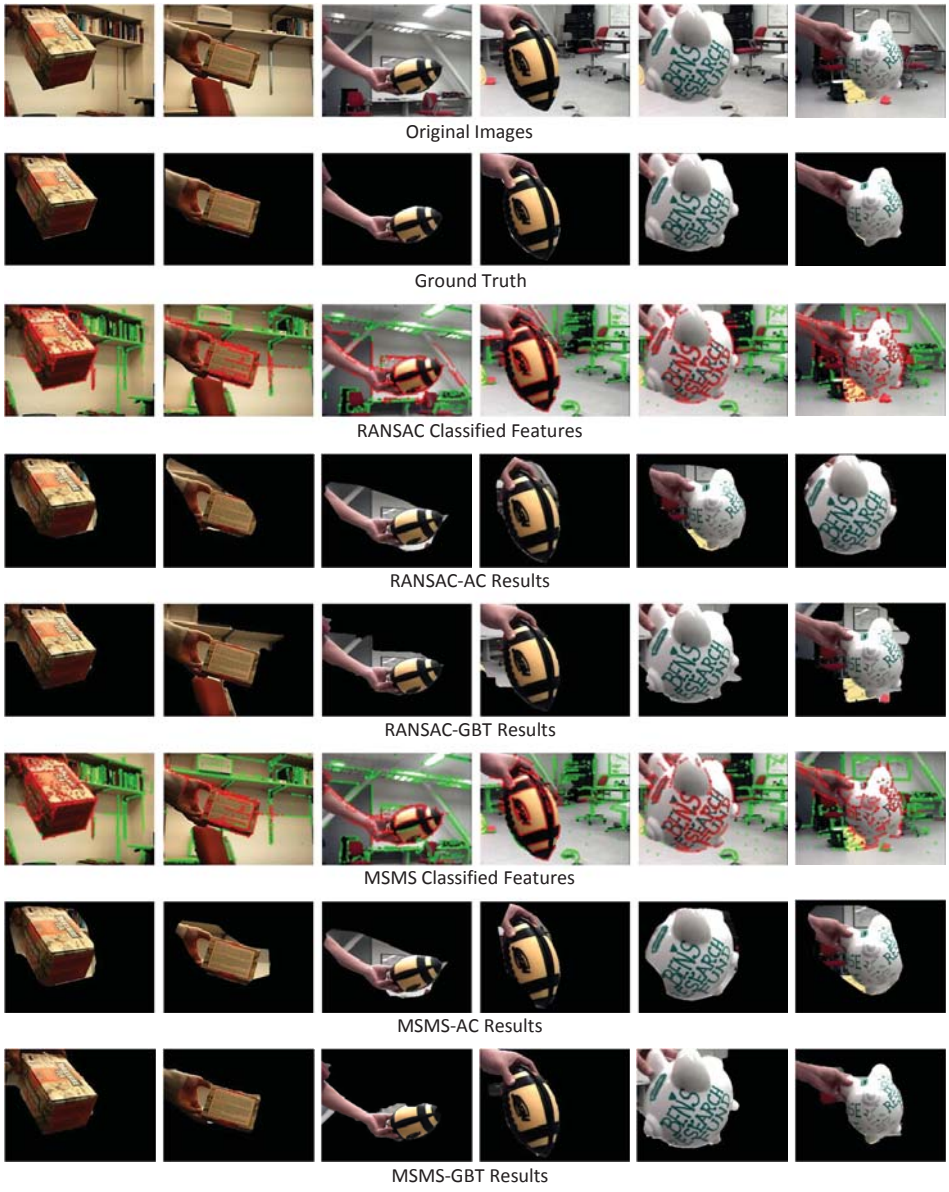


Figure 5.7: Detection results for the HM case (pan tilt camera). The rows show the original images, ground truth, classified sparse features by RANSAC (background in green and foreground in red), segmentation results by RANSAC-AC, segmentation results by RANSAC-GBT, classified sparse features by MSMS, segmentation results by MSMS-AC, and segmentation results by MSMS-GBT, respectively (best viewed in pdf).

Detection accuracy comparison for the HM case (pan tilt camera)

Accuracy (%)	RANSAC-AC	RANSAC-GBT	MSMS-AC	MSMS-GBT
Tea-box	63.6	69.7	75.3	**82.9**
Football	61.1	65.5	68.0	**79.5**
Toy-pig	64.5	68.1	69.3	**79.5**
Average	63.1	67.8	70.9	**80.6**

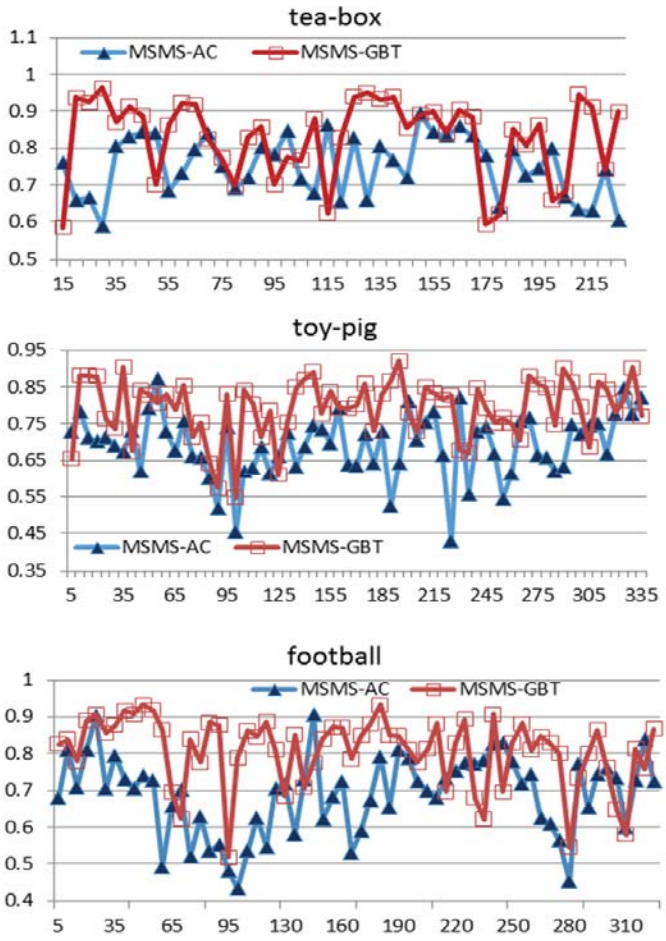


Figure 5.8: Quantitative evaluation results for the HM case. The table shows the detection accuracy for RANSAC-AC, RANSAC-GBT, MSMS-AC, and MSMS-GBT. The plots show the detailed comparisons between MSMS-AC and MSMS-GBT (x-axis: frame number, y-axis: detection accuracy).

the foreground objects have large translation, rotation, and scale change. The qualitative results for MSMS-AC and MSMS-GBT are listed in Fig. 5.7. The AC method is simple and fast, but it may miss boundary details. The AC performance highly relies on the quality of sparse feature detection. Moreover, AC may extract extra regions from the background because its initialized shape by a convex hull is significantly different from the real object boundary. In order to preserve more details on the boundaries, we apply the GBT method in our system. In general, GBT segments the moving objects with better boundaries, since it makes use of the distribution of pixel features for segmentation.

Similar to the work by Sheikh et al. [2009], in the baseline experiments we use RANSAC to fit a homography matrix between two frames, and take the features that are consistent with the fitted homography matrix as background features and others as foreground features. Then we further use AC and GBT for dense foreground object segmentation. The results are shown in Fig. 5.7. Comparing the classified sparse features obtained by RANSAC and MSMS, we can see that RANSAC misclassifies many features that are close to the moving objects. As a result, background regions may be segmented into foreground .

The table in Fig. 5.8 illustrates the quantitative results for both RANSAC (baseline) and MSMS (our method). On average, MSMS-GBT improves the performance about 13% over RANSAC-GBT, and about 10% over MSMS-AC. The detailed comparison between MSMS-AC and MSMS-GBT is shown in the graphs in Fig. 5.8.

5.5.5 Free-Moving Camera Case

Our approach is qualitatively and quantitatively evaluated on another three hand-held moving object videos: “football”, “toy-pig”, and “soccer”, taken from a camera mounted on a moving robot. Beside the change of object position, orientation, and scale, the videos also have significant robot translation.

Fig. 5.9 shows some typical images for detected “football”, “toy-pig” and “soccer”. The baseline comparison method fits a fundamental matrix from RANSAC, and labels the

Figure 5.9: Detection results for the FM case (camera on a mobile robot). The rows show the original images, ground truth, classified sparse features by RANSAC (background in green and foreground in red), segmentation results by RANSAC-AC, segmentation results by RANSAC-GBT, classified sparse features by MSMS, segmentation results by MSMS-AC, and segmentation results by MSMS-GBT, respectively (best viewed in pdf).

Detection accuracy comparison for the FM case (camera on a mobile robot)

Accuracy (%)	RANSAC-AC	RANSAC-GBT	MSMS-AC	MSMS-GBT
Football	53.2	57.1	56.5	**64.0**
Soccer	53.0	58.7	59.8	**68.5**
Toy-pig	45.3	51.2	54.3	**62.0**
Average	50.5	55.7	56.9	**64.8**

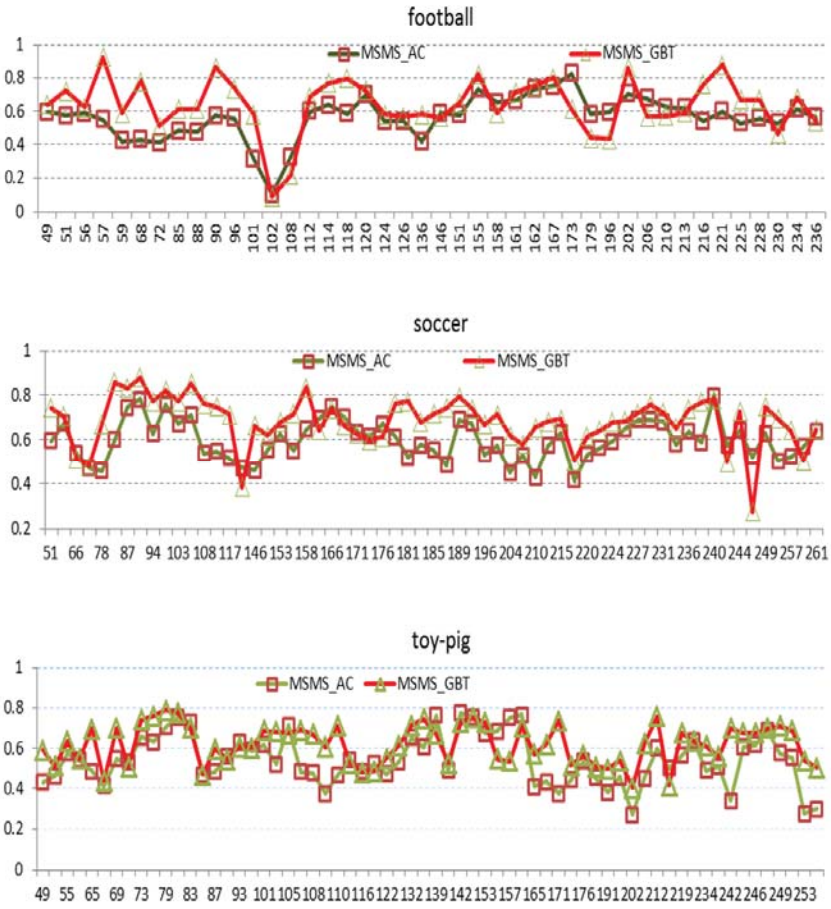


Figure 5.10: Quantitative evaluation results for the FM case. The table shows the detection accuracy for RANSAC-AC, RANSAC-GBT, MSMS-AC, and MSMS-GBT. The plots show the detailed comparisons between MSMS-AC and MSMS-GBT (x-axis: frame number, y-axis: detection accuracy).

features that are consistent with the fundamental matrix as background and other features as foreground. The MSMS method gets better accuracy in overall than RANSAC based fundamental matrix fitting. And again, GBT segmentation results are much better than those from AC.

The quantitative results are shown in the table in Fig. 5.10. On average, MSMS-GBT improves the performance about 9% over RANSAC-GBT, and about 8% over MSMS-AC. The detailed comparison between MSMS-AC and MSMS-GBT is shown in the graphs in Fig. 5.10.

Although we aim at segmenting moving objects in close-up view, we also applied our method on moving objects in far view, and the results are reasonable. Fig. 5.11 (a) shows some detection results for a non-rigid object, a walking person, taken by a tilt-pan camera. Our motion segmentation approach performs well in this video, where the foreground object has large depth and shape change. In addition, Fig. 5.11 (b) shows some detection results on the "wheelchair" video, taken from a moving robot, using MSMS-GBT. The performance is reasonable even though the foreground object is far away from the robot as long as there are still a set of features detected on the objects.

Figure 5.11: (a) Detection results on a walking person video for the HM case, (b) Detection results on a moving wheelchair for the FM case. The rows show the original images and MSMS-GBT detection results.

Chapter 6

3D Pose Estimation

After the 2D2D object model is built, we move forward to build the 2D3D object model in the OSH, which involves tracking a planar component and estimating its 3D normal. The estimation of a planar component's 3D normal is usually coupled with the problem of camera motion estimation.

6.1 Introduction

6.1.1 Existing Works

Two classic approaches have been widely used for camera motion estimation from two views of the same 3D scene: the homography matrix based approach and the essential matrix based approach.

The homography matrix based (HMB) approach [Sturm, 2000; Zhang, 2000; Ma, 2004; Hartley and Zisserman, 2003; Cobzas et al., 2009; Molton et al., 2004] works for a planar environment. Two views of a planar surface are related by a homography matrix. The camera motion parameters as well as the plane normal can be obtained from decomposing the homography matrix. The essential matrix based (EMB) approach [Pollefeys et al., 2004; Nistér, 2004; Zhang, 1998; Longuet-Higgins, 1981; Hartley and Zisserman, 2003;

Ma, 2004] works for a more general environment. For two sets of calibrated corresponding points in two images where not all points lie on the same planar surface, they can be related by an essential matrix (in the uncalibrated case, it is called the fundamental matrix). By decomposing the essential matrix, we can get the camera motion parameters.

Other existing approaches for recovery of camera motion and scene/object structure include Bundle Adjustment (BA) [Snavely et al., 2008; Klein and Murray, 2007; Sibley et al., 2009; Triggs et al., 2000; Engels et al., 2006] and Visual SLAM [Newcombe and Davison, 2010; Davison et al., 2007; Cummins and Newman, 2009; Eade and Drummond, 2006]. The Bundle Adjustment approach takes multiple views as input and finds a solution for camera motion and scene structure parameters by minimizing the total re-projection error with respect to all 3D feature points and camera motion parameters. The minimization is taken iteratively until the solution converges, where a good start point needs to be provided in order for the approach to converge to global minimum. The dimension of the parameters to be estimated in this approach is linear in the number of images and features. When the number of features or images is large, the size of the search space for the parameters will be extremely high. The Visual SLAM approach estimates the camera pose and 3D feature points in at least a $6+3n$ dimensional space at each time step, where n is the number of features, and in general the parameter dimension is much larger due to over-parametrization [Davison et al., 2007].

The HMB approach takes planar features (a planar region or a set of point features on a plane) as input. The input to existing EMB, Bundle Adjustment, and Visual SLAM approaches are usually a set of general point features that do not lie on the same plane. Point features and planar features each have their own merits. Compared to non-planar features, a planar feature has larger range of visibility (hence tracking duration), that is, all pixels in a non-occluded planar region are observable until the region degenerates to a line in the image. In addition, a planar region feature has a unique normal and hence a simpler 3D geometric representation, and the 2D images of a planar region feature can be well re-

lated by well-understood homography transformations [Hartley and Zisserman, 2003; Ma, 2004]. More importantly, a single planar region typically contains at least hundreds of pixels, which impose strong constraints for acquisition of high-precision pose and structure information. Compared to planar features, point features are not restricted to planar environments. They can be easily detected and a natural image usually contains a large number of point features.

6.1.2 Our Method

The 2D3D layer in the OSH is represented by a set of 2D planar components with their individual 3D poses. This representation leads naturally to an emphasis on recovering the 3D pose (mainly the 3D normal) of a planar component. With the recovered component pose, we will be able to calculate the camera motion and then reconstruct the object/scene structure. In this chapter and Chapter 7, we will present a 3D model construction method that starts with estimating the 3D normal of a planar region feature and discuss the method's advantages.

Our 3D structure recovery method (LSMGS: from Local Structure to Motion then to Global Structure) takes advantage of both planar features and point features. Instead of estimating the high-dimensional global structure parameters directly, a simple local structure estimation is first carried out. This local structure estimation facilitates later recovery of camera motion and global scene/object structure. The method includes the following steps:

- ***Local Structure***: we start by selecting a single planar region feature to track, and estimating its 3D normal; the estimation step exploits information from both the tracked planar region feature and a set of tracked point features.

- ***Motion***: then camera motion is obtained based on the estimated normal of the planar region feature.

- ***Global Structure***: the full 3D scene structure is then recovered from the camera pose and the tracked point features in the scene.

In this chapter, we will focus on local structure recovery and motion estimation, the recovery of global structure will be presented in Chapter 7.

The proposed method fuses constraints from both the homography matrix and the essential matrix in a single probabilistic framework. It does not simply take the average of the solutions from the two classic HMB [Sturm, 2000; Zhang, 2000; Hartley and Zisserman, 2003; Ma, 2004; Cobzas et al., 2009; Molton et al., 2004] and EMB [Pollefeys et al., 2004; Nistér, 2004; Zhang, 1998; Longuet-Higgins, 1981; Hartley and Zisserman, 2003; Ma, 2004] approaches, since both approaches have well known restrictions. The HMB approach may give two physically possible solutions [Ma, 2004], because in the decomposition of a homography matrix, the camera translation and plane normal terms are interchangeable (see Section 6.5.4). It can be very difficult to select the correct solution when the camera motion is small or when the camera moves on a line without any rotation. Our method provides a unique solution based on all observations in either multiple views or an image sequence. The EMB approach requires RANSAC-like fitting and a good number (at least 5 [Nistér, 2004], usually many more) of point features that do not lie on the same plane. In contrast, no RANSAC-like fitting is needed in our method, and usually a small set of point features are good enough, thanks to using a planar region feature. While the proposed method avoids the drawbacks of the HMB and EMB approaches, it also improves the estimation accuracy.

Compared to the Bundle Adjustment and Visual SLAM approaches [Snavely et al., 2008; Klein and Murray, 2007; Sibley et al., 2009; Triggs et al., 2000; Engels et al., 2006; Davison et al., 2007; Newcombe and Davison, 2010; Nister et al., 2006; Eade and Drummond, 2006; Cummins and Newman, 2009] that estimate high dimensional parameters simultaneously, our method maintains a probability distribution over only two dimensional normal parameters for a single tracked planar region feature, and the other motion and structure parameters are calculated based on the two estimated normal parameters. Low dimensional

parameter estimation allows robust and fast convergence to the solution. We do not need prior knowledge on the initial camera pose and scene structure, that is, no special care needs to be taken for parameter initialization. In contrast to Visual SLAM which requires continuous image input, the proposed method can be applied to both continuous tracked features in an image sequence and discrete matched features in a set of static images (which is similar to Bundle Adjustment). Our method dynamically updates the distribution of the normal of the planar region feature when each new view or frame comes in, and works in an incremental manner such that the computational cost for each new view or frame remains constant and does not increase with the growing number of views or frames.

We test our method on videos taken from a hand-held webcam. Each video contains only about 80 to 200 frames. Our method works well for these noisy short-time videos. The capability of estimating motion parameters in a short time will allow a robot to quickly construct a temporary 3D model of the environment before it shifts its attention and changes its visual field significantly.

In summary, we propose a novel camera motion and scene structure recovery method based on probabilistic normal estimation of a single tracked planar region feature. The primary novelties of the method are: **(1)** To our best knowledge, this is the first work that fuses the constraints from both the homography matrix and the essential matrix into a single framework for camera motion and scene structure recovery. Compared to the classic homography and essential matrix based approaches, the proposed method gives more accurate estimation results and avoids the drawbacks of the two approaches. **(2)** The proposed method takes advantage of both planar features and point features. Using planar features greatly improves estimation accuracy over using point features only, and with the help of point features, the solution ambiguity from planar features is resolved. **(3)** Compared to traditional Bundle Adjustment and Visual SLAM approaches, the proposed method maintains a probability distribution over parameters in two dimensional space instead of a high dimensional space, which allows robust and fast convergence to the solution.

6.2 Detection and Tracking of a Planar Region Feature

To detect planar region features, we start by extracting contour fragments from the input image. Then we examine two types of planar region feature candidates. The first type of candidates are based on single contour fragments. For each contour fragment, we form a closed contour by connecting its two ending points. If the area of the region bounded by the closed contour is above some threshold, we add this region into the candidate feature list.

The second type of candidates are based on contour fragment pairs. For each pair of neighboring contour fragments (one fragment has an ending point close to an ending point on the other fragment), we connect the two close ending points and also the remaining two ending points to form a closed region. If the region has an area larger than the pre-selected threshold, it is added to the candidate feature list. Since the contour fragments in the two types of feature candidates usually have similar color or texture around them, they are very likely to lie on the same planar surface. Those that do not lie on the same planar surface will have large tracking errors and will be removed during tracking.

Figure 6.1: Planar region feature detection (best viewed in color). Left: input image, middle: a set of planar region feature candidates in different colors, right: the best planar region feature (red contour) used for camera pose recovery.

The candidate features are tracked for some time, and the one with the lowest tracking error is selected as the planar region feature for further tracking and for camera pose recovery. Fig. 6.1 shows some candidate features and the best feature used for camera pose recovery.

Tracking a planar region feature takes two steps: sparse feature tracking and dense

pixel alignment. Sparse feature tracking gives a rough estimate of the planar region feature location, and dense pixel alignment refines the estimate. Combining sparse feature tracking and dense pixel alignment allows us to track a planar region feature robustly and accurately. The tracking scheme is explained in Fig. 6.2.

Figure 6.2: Planar region feature tracking scheme (best viewed in color). The left image shows the tracked planar region feature (red contour) and point features in frame $t-1$. The point features are tracked in frame t in the middle image using the KLT method [Shi and Tomasi, 1994], and the boundary of the planar region feature is predicted in frame t as the yellow contour using the point feature correspondences between the two frames. Then the location of the planar region feature is refined in frame t by dense pixel alignment, shown as a new red contour in the right image. Note that in our work the sparse feature tracking step is taken only between two adjacent frames, but we have intentionally enlarged the frame interval and tracking errors in this figure to clearly show the tracking scheme.

In the sparse feature tracking step, besides using information inside the planar region, we also use information outside that region. When there are no objects moving in the background environment, all sparse features in the input image have similar motion patterns to the planar region feature and all of them can be used to assist tracking the planar region feature; otherwise, we can locate sparse features with similar motion patterns to the planar region feature using Generalized Hough Transform [Grabner et al., 2010]. Tracking the sparse features helps avoid getting stuck at local minima and improve tracking stability. In the dense pixel alignment step, we adopt the Inverse Compositional algorithm [Baker and Matthews, 2004] and refine the location of the planar region feature. This refinement provides more accurate input to camera pose estimation.

6.3 Probabilistic Normal Estimation for a Planar Region Feature

The coordinates of image points between two poses of a planar region feature are related by a homography matrix H, and we have

$$H = R + TN^T/d \tag{6.1}$$

where R and T are the rotation and translation matrices relating the two poses, d and N are the distance and unit normal of the plane in the reference camera space [Ma, 2004]. From at least four pairs of matching points, H can be determined up to a scaling factor using the Direct Linear Transform method. The computed H can then be decomposed to give two sets of solutions for $\{N, R, T/d\}$ (Algorithm 5.2, [Ma, 2004]). While this method is fast, it's sensitive to noise, especially when the number of input features is small. Also the selection of the correct solution from the two candidate solutions can be difficult without prior knowledge. One way to overcome the problem of noise and unstable pose estimates is to minimize an error measure over several images at the same time. However, this method is hard to converge with poor initializations and the algorithm sometimes converges to the wrong minima.

We present a probabilistic method to estimate the planar region feature's normal in 3D space, given the tracked locations of the planar region feature and a set of tracked point features on the entire image. This method provides a unique solution for the planar region feature's normal based on all observations up to the current frame, and the computational cost at each time step does not increase with the growing number of frames.

6.3.1 Probabilistic Framework

We refer to a planar region feature as a planar component, and define a *component space*, where the x-axis is arbitrarily chosen on the component, the z-axis is along the direction

of the component normal, and the origin can be any arbitrary point on the component. Note that every point belonging to the component will have zero value on the z-axis in this component space. At any time t, the component normal is denoted as N_t in the camera space (in the direction from the component away from the camera origin), and the distance between the origin of the camera space and the component plane is denoted as d_t. The camera and component spaces are shown in Fig. 6.3.

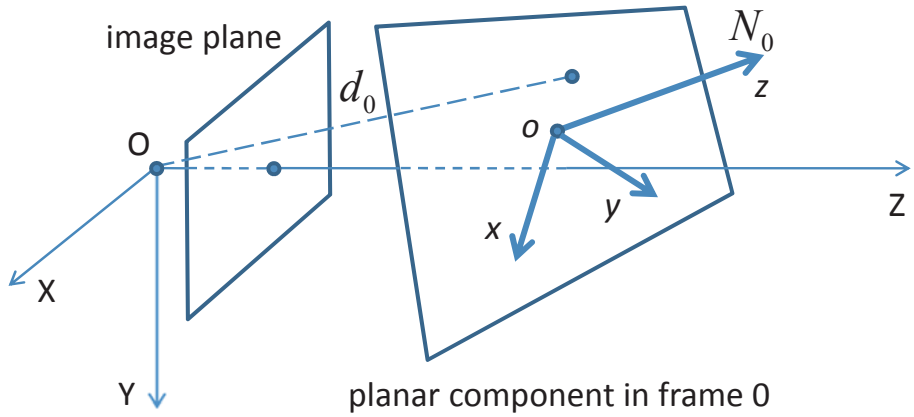


Figure 6.3: Camera space and component space. The origin of the camera space is the camera's optical center, and the X-axis and Y-axis are parallel to the image plane, pointing right and down, respectively. For the component space, the x-axis is arbitrarily chosen on the component, the z-axis is along the direction of the component normal (pointing from the component away from the camera origin), and the origin can be any arbitrary point on the component.

In an image sequence, we refer to frame 0 as the reference frame. The 3D component in the reference frame can be represented as

$$N_0 P = d_0 \tag{6.2}$$

for any 3D point $P = (P_x, P_y, P_z)^T$ on the component. Here N_0 and P are both represented in the camera space.

The key step for motion and structure recovery in the image sequence is to estimate

the component's normal N_0 (without loss of generality, d_0 can be set to 1). At every time step, we dynamically update N_0 based on the tracked features in frame t. With the estimated N_0, the camera motion can then be easily recovered at that time step.

We represent N_0 in a spherical coordinates as

$$N_0 = (sin\theta_0^N cos\phi_0^N, sin\theta_0^N sin\phi_0^N, cos\theta_0^N)^T \tag{6.3}$$

where $\theta_0^N \in [0, \pi/2]$ and $\phi_0^N \in [0, 2\pi)$ are the normal parameters.

Our goal is to estimate the probability density function $Pr(\theta_0^N, \phi_0^N | z_{0:t})$, where z are the observations, that is, the tracked features.

By applying Bayes' theorem, we have

$$\begin{aligned} & Pr(\theta_0^N, \phi_0^N | z_{0:t}) \\ \propto \quad & Pr(\theta_0^N, \phi_0^N | z_0) Pr(z_{1:t} | \theta_0^N, \phi_0^N, z_0). \end{aligned} \tag{6.4}$$

Under the independent observation assumption that z_t is independent of $z_{1:(t-1)}$, we rewrite Eq. 6.4 as

$$\begin{aligned} & Pr(\theta_0^N, \phi_0^N | z_{0:t}) \\ \propto \quad & Pr(\theta_0^N, \phi_0^N | z_0) \prod_{k=1}^{t} Pr(z_k | \theta_0^N, \phi_0^N, z_0) \end{aligned} \tag{6.5}$$

which enables us to obtain a recursive formulation as

$$\begin{aligned} & Pr(\theta_0^N, \phi_0^N | z_{0:t}) \\ \propto \quad & Pr(z_t | \theta_0^N, \phi_0^N, z_0) Pr(\theta_0^N, \phi_0^N | z_{0:(t-1)}). \end{aligned} \tag{6.6}$$

Based on Eq. 6.6, the problem is then reduced to choosing the likelihood function $Pr(z_t | \theta_0^N, \phi_0^N, z_0)$ and the prior function $Pr(\theta_0^N, \phi_0^N | z_0)$, where the prior function can be chosen as a uniform distribution if no prior knowledge about the tracked planar region

feature is available.

Since the formulation in Eq. 6.6 is recursive such that at each time step only the current observation is used to update the estimation, the computational cost at each time step stays constant and does not grow with increasing number of frames. From Eq. 6.5 and 6.6, we can see that this method will also apply to a set of static images (as opposed to videos), as long as feature correspondences can be obtained among the images.

6.3.2 Likelihood Formulation

Let $p_t = (p_{ut}, p_{vt})^T$ be the calibrated image coordinates at time t corresponding to a 3D point P on the planar component. Using a perspective camera model, we can calculate P from Eq. 6.2 given N_0, d_0 and p_0. Thus a component space can be built, and the corresponding point of P is denoted as

$$P^c = \begin{pmatrix} P_x^c \\ P_y^c \\ 0 \end{pmatrix} \tag{6.7}$$

in the component space. Note that P^c is time-invariant.

At time t, let the translation and rotation from the component space to the camera space be T_t and

$$R_t = (R_{1t} \;\; R_{2t} \;\; R_{3t}) \tag{6.8}$$

where $R_{kt} (k = 1, 2, 3)$ are the column vectors in R_t. The point P^c on the component and its

image coordinates p_t are related by

$$\begin{aligned}
\lambda^P \begin{pmatrix} p_{ut} \\ p_{vt} \\ 1 \end{pmatrix} &= (R_{1t}\ R_{2t}\ R_{3t}\ T_t) \begin{pmatrix} P_x^c \\ P_y^c \\ 0 \\ 1 \end{pmatrix} \\
&= (R_{1t}\ R_{2t}\ T_t) \begin{pmatrix} P_x^c \\ P_y^c \\ 1 \end{pmatrix} \\
&= H_t \begin{pmatrix} P_x^c \\ P_y^c \\ 1 \end{pmatrix} \qquad (6.9)
\end{aligned}$$

where λ^P is the point depth in the camera space, and H_t is a homography matrix that maps points from the component plane to the image plane. Note that we are using H_t to refer to the transformation between a 3D plane and its projected 2D image here, which is different from the H_t in Chapter 4 which refers to a transformation from one 2D image to another 2D image of the same 3D component.

In our system, the transformation H_t is obtained (up to a scaling factor) from the tracking step. Since R_t is a rotation matrix, it satisfies

$$\|R_{1t}\| = \|R_{2t}\| = 1 \qquad (6.10)$$

and

$$R_{1t} \perp R_{2t} \qquad (6.11)$$

Equivalently we have the following constraints,

$$\|H_{1t}\| - \|H_{2t}\| = 0 \tag{6.12}$$

$$H_{1t}^T H_{2t} = 0 \tag{6.13}$$

where H_{1t} and H_{2t} are the first two column vectors in H_t.

The likelihood function $Pr(z_t|\theta_0^N, \phi_0^N, z_0)$ in Eq. 6.6 can directly be designed based on the two homography constraints in Eq. 6.12 and 6.13 (such as in [Xu et al., 2009]), but it will be difficult to choose the weights for the two error terms corresponding to the constraints. We instead formulate these constraints on a new single error term, and this error term is expressed in the intuitive 2D image space rather than in the parameter space of the homography matrix. In addition, we take advantage of tracked point features on the entire image (instead of just on the component) and integrate constraints from the essential matrix on these features to improve estimation accuracy.

From H_t we get the estimated rotation matrix $\hat{R}_t$ as

$$\begin{aligned} \hat{R}_{1t} &= H_{1t}/\beta \\ \hat{R}_{2t} &= H_{2t}/\beta \\ \hat{R}_{3t} &= \hat{R}_{1t} \times \hat{R}_{2t} \end{aligned} \tag{6.14}$$

where

$$\beta = (\|H_{1t}\| + \|H_{2t}\|)/2 \tag{6.15}$$

is a normalizing term. Ideally $\hat{R}_t$ should be a real rotation matrix, but in general it is not, due to tracking inaccuracy and/or wrong normal assumptions. Thus, we seek to find a real

rotation matrix R_t such that

$$R_t = arg\min_R \|R - \hat{R}_t\| \tag{6.16}$$

subject to

$$det(R) = 1$$

where I is the identity matrix. Let the singular value decomposition of $\hat{R}_t$ be USV^T, then the solution to Eq. 6.16 is $R_t = UV^T$ [Zhang, 2000]. The translation T_t is estimated as

$$T_t = H_{3t}/\beta. \tag{6.17}$$

It can be easily verified that the relative motion from the reference camera space to the current camera space is

$$\begin{aligned} \delta R_t &= R_t R_0^{-1} \\ \delta T_t &= T_t - R_t R_0^{-1} T_0 \end{aligned} \tag{6.18}$$

where δR_t is the relative rotation and δT_t is the relative translation.

Given δR_t, δT_t, and N_0, the constraint from the homography matrix can be expressed [Ma, 2004] as

$$p_t = \mathbb{S}^h((\delta R_t + \delta T_t N_0^T/d_0)p_0) \tag{6.19}$$

for an arbitrary pair of image points $\{p_0, p_t\}$ on the component, where $d_0 = 1$ and $\mathbb{S}^h(\cdot)$ is a scaling function of a three-dimensional vector such that $\mathbb{S}^h((a,b,c)^T) = (a/c, b/c, 1)^T$ where $c \neq 0$.

The constraint from the essential matrix [Hartley and Zisserman, 2003] is

$$p_t^T \mathbb{S}^e([\delta T_t]_\times \delta R_t p_0) = 0 \tag{6.20}$$

for an arbitrary pair of points $\{p_0, p_t\}$ on the entire image, where $[\delta T_t]_\times$ is the skew-symmetric matrix for δT_t and $\mathbb{S}^e(\cdot)$ is a scaling function of a three-dimensional vector such that $\mathbb{S}^e((a,b,c)^T) = (a,b,c)^T/\sqrt{a^2+b^2}$ where $a^2+b^2>0$. The scaling function $\mathbb{S}^e(\cdot)$ guarantees that the left hand side of Eq. 6.20 is measured in the image space. The introduction of $\mathbb{S}^h(\cdot)$ and $\mathbb{S}^e(\cdot)$ is due to the fact that a homography or essential matrix can only be determined up to a scale.

Based on Eq. 6.19, we define a likelihood term for the homography constraint as

$$L_t^h \propto \prod_{\{p_0,p_t\}} \exp(-\frac{\|\mathbb{S}^h((\delta R_t + \delta T_t N_0^T/d_0)p_0) - p_t\|^2}{2(\sigma^h)^2}) \tag{6.21}$$

for a set of tracked point pairs $\{p_0, p_t\}$ on the planar component and σ^h is a constant. In our experiments, we use the boundary points of the planar component to calculate the above likelihood. Note that the two homography constraints in Eq. 6.12 and 6.13 in the homography parameter space have been expressed in a single error term in Eq. 6.21 in the intuitive image space. Similarly we define a likelihood term for the essential matrix constraint as

$$L_t^e \propto \prod_{\{p_0,p_t\}} \exp(-\frac{(p_t^T \mathbb{S}^e([\delta T_t]_\times \delta R_t p_0))^2}{2(\sigma^e)^2}) \tag{6.22}$$

for a set of tracked point pairs $\{p_0, p_t\}$ on the entire image and σ^e is a constant. In our experiments, we use all tracked point features in the image to calculate the above likelihood.

We then design the likelihood function in Eq. 6.6 as

$$Pr(z_t|\theta_0^N, \phi_0^N, z_0) = \gamma L_t^h L_t^e \tag{6.23}$$

where γ is a constant normalizing term. Intuitively, lower errors lead to higher likelihood.

6.3.3 Optimal Solution

Initially the entire parameter space for $\{\theta_0^N, \phi_0^N\}$ is uniformly sampled, and with time going, those samples with low posterior probability are removed. The solution for the normal parameters is obtained as

$$(\hat{\theta}_0^N, \hat{\phi}_0^N) = \arg \max_{(\theta_0^N, \phi_0^N)} Pr(\theta_0^N, \phi_0^N | z_{0:t}) \tag{6.24}$$

where we choose the sample with the maximum posterior probability as the best estimate for $\{\theta_0^N, \phi_0^N\}$.

6.3.4 Discussions

Tracked point features outside the planar region feature help improve estimation accuracy. In particular, when the camera moves on a line without any rotation, the homography constraint will lead to a bimodal distribution (see experiments) corresponding to the two consistent solutions in homography decomposition [Ma, 2004]. With the help of point features outside the planar region feature, the distribution will adapt to be unimodal. Although tracked point features will help improve the estimation result, the method will still work without any point feature as long as we can track a single planar region feature. Accordingly, unlike the classic essential matrix based approach for camera motion estimation which requires at least a certain number (at least 5 [Nistér, 2004], usually 8 or more) of point features, we have no minimum limit for point feature number; even if there is only one point feature, we can still make use of it in Eq. 6.22 and 6.23. Fig. 6.4 shows some tracked poses of the planar region feature.

Figure 6.4: Planar region tracking and pose estimation (best viewed in color). The image sequence is captured from a commonly available webcam. Top row: tracked planar region feature and point features, bottom row: estimated normals for the planar region feature. The normals are shown in the component space (see text for details), where x-axis and y-axis (in light blue color) are on the same plane as the planar region, and z-axis (in dark blue color) is along the region normal.

6.4 Camera Motion Recovery

We recover the camera pose trajectory with respect to the camera space in the reference frame. Given the rotation R_0 and translation T_0 from the component space to the reference camera space, and R_t and T_t from the component space to the current camera space, we have

$$\begin{aligned} P_0 &= R_0 P^c + T_0 \\ P_t &= R_t P^c + T_t \end{aligned} \tag{6.25}$$

where P^c is an arbitrary point in the component space, and P_0 and P_t are the corresponding points in the reference and current camera space respectively. Let the relative rotation and translation from the reference camera space to the current camera space be δR_t and δT_t,

then we have

$$P_t = \delta R_t P_0 + \delta T_t \tag{6.26}$$

which maps P_0 to P_t directly in the two camera spaces. By combining the first equation in Eq. 6.25, we rewrite Eq. 6.26 as

$$P_t = \delta R_t (R_0 P^c + T_0) + \delta T_t \tag{6.27}$$

which holds true for any point P^c.

Combining Eq. 6.27 and the second equation in Eq. 6.25 leads to the solution of the relative motion as

$$\begin{aligned} \delta R_t &= R_t R_0^{-1} \\ \delta T_t &= T_t - R_t R_0^{-1} T_0 \end{aligned} \tag{6.28}$$

from the reference camera space to the current camera space.

Fig. 6.5 shows the camera pose trajectory for the video example used in Section 6.2 and 6.3.

6.5 Experimental Results

6.5.1 Datasets

The image sequences in our experiments are taken from a hand-held webcam with 320×240 resolution, each containing 150 to 200 frames. The image sequences are noisy because their resolution is low. We collect seven datasets: two tea boxes (Yorkshire and Twinings), a cell phone box (Pantech), a toy car (Ambulance), and three complex environments containing non-planar objects (Disk, Poster, Glue) (Fig. 6.6).

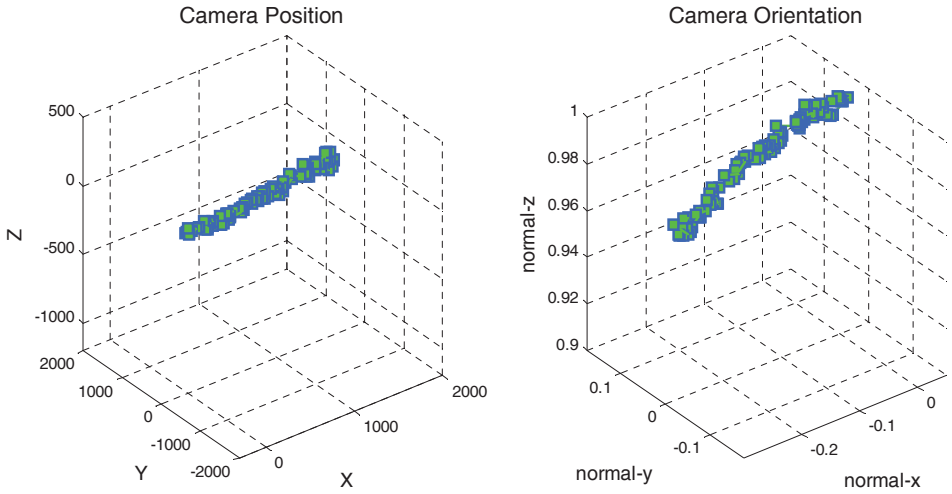


Figure 6.5: Recovered 3D camera pose trajectory (best viewed in pdf). The video is captured from a hand-held webcam, where the webcam moves roughly on a line while keeping the object around the image center. Left: position trajectory, right: orientation trajectory.

For each dataset, we manually mark a set of evaluation points across frames. For the first four datasets, because of their simple geometry, we measure the ground truth 3D positions for the evaluation points and evaluate the methods based on 3D reconstruction errors. For the last three datasets, it is difficult to get 3D ground truth data because of their geometric complexity, so the evaluation is carried out based on 2D re-projection errors.

Note that the reason we choose simple objects for the first four datasets is for easy acquisition of the 3D ground truth data to carry on evaluation in 3D space. Our method does not limit its application to simple objects as long as there exists a single detectable planar region feature (such as the last three datasets, or buildings, offices, walls, ground, and many other manufactured objects).

6.5.2 Comparison Baselines

To quantitatively evaluate the accuracy of the proposed method (LSMGS), we compare it against the classic homography matrix based (HMB) approach, the essential matrix based (EMB) approach, and Bundler [Snavely et al., 2008].

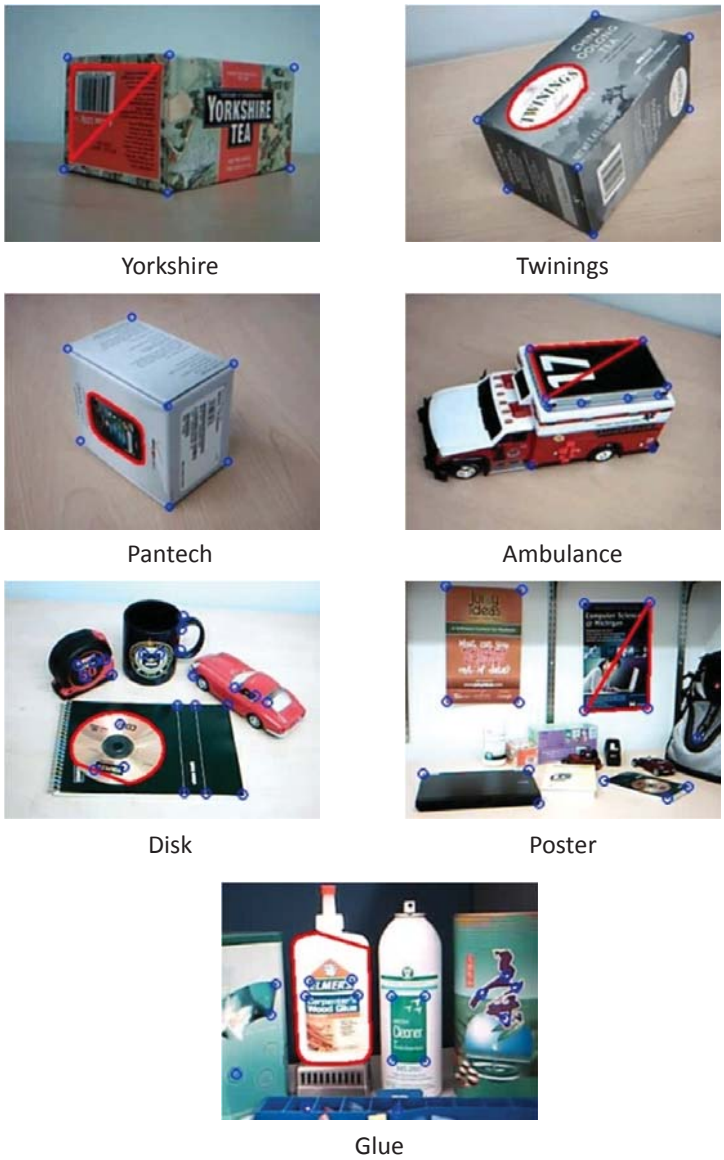


Figure 6.6: Evaluation datasets for pose estimation. The selected points (blue circles) are used for evaluation, and the red contours are the planar region features used for pose estimation. The first four datasets have simple geometry, and they are evaluated on 3D reconstruction errors. The last three contain non-planar objects and are difficult to get 3D ground truth data, so they are evaluated based on 2D re-projection errors.

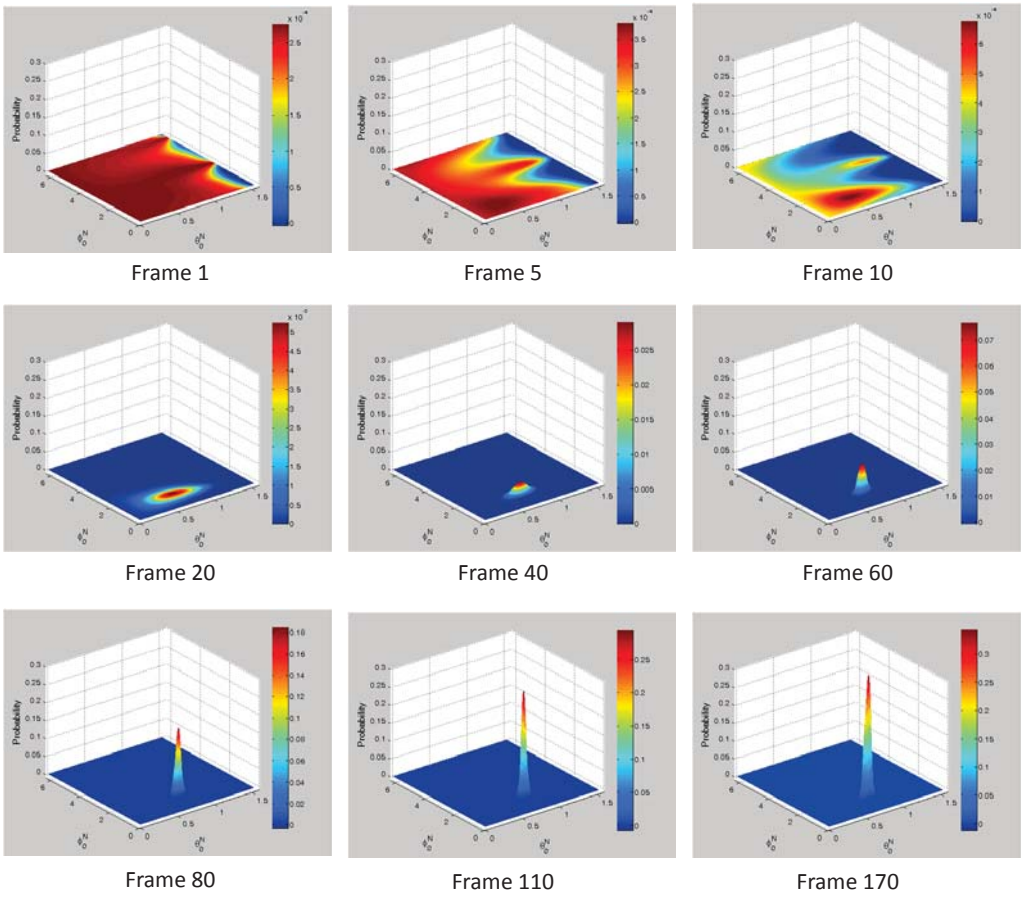


Figure 6.7: Probability distribution of component normal. The probability distribution of the normal of the component changes over time. It starts from a uniform distribution and evolves to a Gaussian-like distribution.

Our planar region feature tracking algorithm provides the homography matrices for each frame with respect to the reference frame. Besides the planar region feature, we also keep track of a set of point features. The input to HMB is the homography matrices, and the input to EMB is the tracked point features. Our method takes both the homography matrices and the point features as input.

The HMB approach decomposes the homography matrices to get the plane normal and camera motion parameters. It may give two physically possible solutions. In each frame, we select the solution in which the estimated plane normal has better consis-

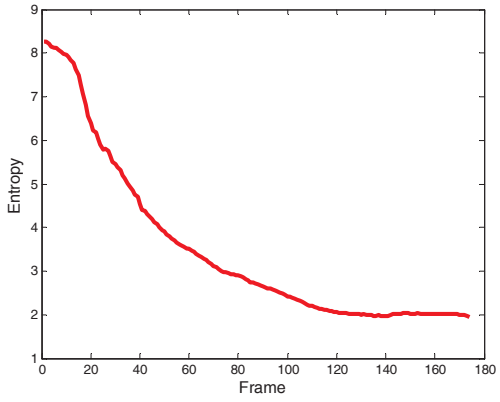


Figure 6.8: Entropy trajectory. The entropy of the distribution of component normal decreases over time.

tency with the normals in previous frames [Cobzas et al., 2009]. The EMB approach uses RANSAC to filter out outliers in the tracked point features, and fits an essential matrix based on the inliers. Then it decomposes the essential matrix to get the camera motion parameters. The LSMGS method always provides a unique solution and does not need any RANSAC-like fitting. Bundler takes the tracked point features as input, and computes motion and structure parameters by nonlinear optimization.

We conduct the comparison experiments on two groups, one with KLT [Shi and Tomasi, 1994] features and the other with SIFT [Lowe, 2004] features. For KLT features, we use the KLT detector/tracker implemented in OpenCV. SIFT features are directly imported from Bundler, where we use the default parameter setting in the software provided online. The number of SIFT features is from a couple of hundreds to over a thousand, while the number of tracked KLT features is around 30 to 50.

For all the experiments, we first calculate the motions parameters from the HMB, EMB, Bundler, and LSMGS methods, and then use the same way (described in Section 7.2) to compute the 3D positions of the evaluation points. Note that we can only get the motion parameters from Bundler and can not directly get the 3D positions for the evaluation points, because the evaluation points are a different set of points from the SIFT features.

Figure 6.9: Normal views under different normal samples. Each normal view corresponds to the view of the planar component under a sampled normal of the component. The last image shows the normal view for the best normal estimate.

In our experiments, we set both σ^h in Eq. 6.21 and σ^e in Eq. 6.22 to 10. Based on our observation, the estimation results are not sensitive to σ^h and σ^e, because what matters is the relative rather than absolute likelihood for the normal hypotheses for the tracked planar region feature.

6.5.3 Probability Distribution of Component Normal

The probability distribution of the normal of the tracked component evolves over time. Fig. 6.7 shows the probability distribution evolution process for the Twinings dataset. The probability distribution is initially uniform. With more and more observations coming in, it eventually converges to a Gaussian-like distribution. In this evolution process, the entropy of the distribution decreases over time, as shown in Fig. 6.8.

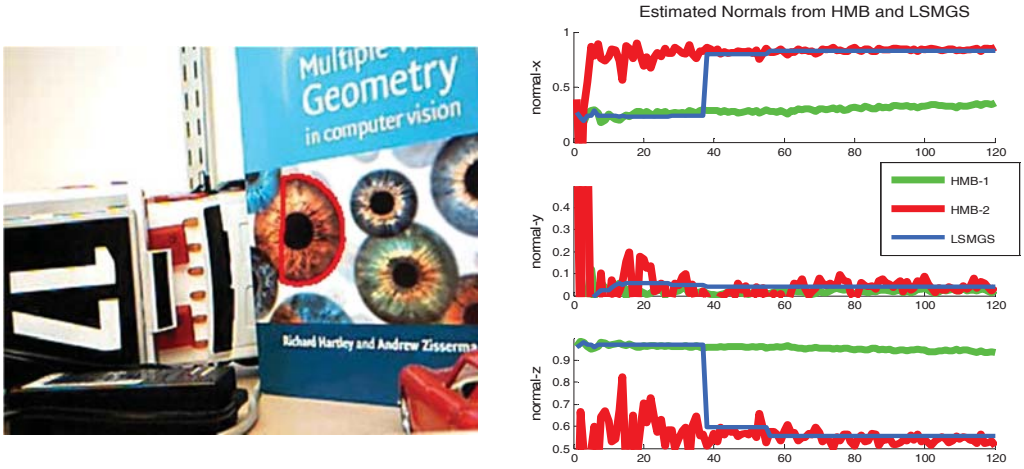
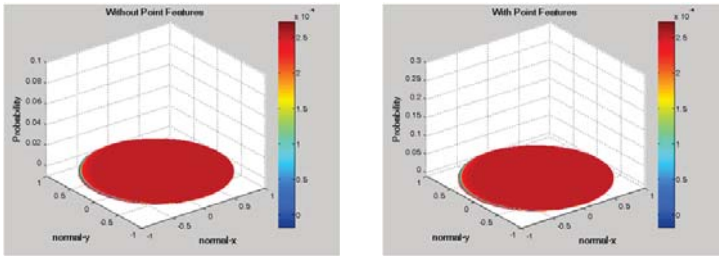
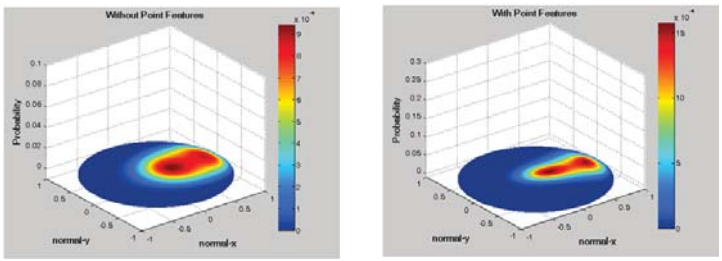


Figure 6.10: Normal trajectories for HMB and LSMGS when the camera moves on a line without rotation. The left image shows the scene with a tracked planar region feature (red contour on the book). The right graph shows the trajectories of the three normal elements (N_x, N_y, N_z) for the two solutions from HMB (red and green) and the unique solution from LSMGS (blue). Both solutions from HMB have very good consistency, which prevents us from selecting the correct solution (the red curve corresponds to the correct solution). In contrast, the solution from LSMGS converges correctly with the help of point features outside the planar region feature.

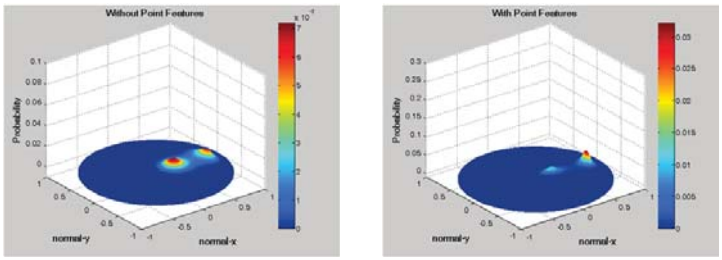
For each sampled normal, we can generate a normal view for the component. Fig. 6.9 shows the normal views for the component corresponding to a set of sampled normals, where the last one corresponds to the best estimated normal.



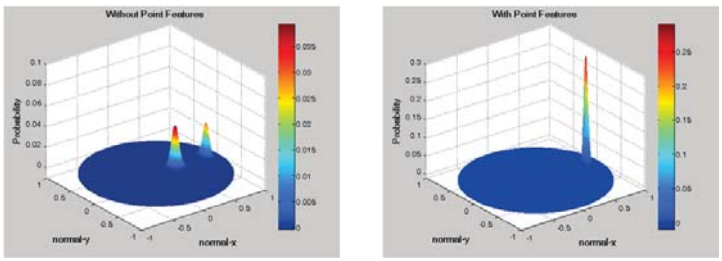
Frame 1



Frame 30



Frame 70



Frame 120

Figure 6.11: Probability distribution with and without point features. Without point features and the essential matrix constraint, the probability distribution will converge to two peaks, which leads to solution ambiguity. When point features and the essential matrix constraint are used, the distribution eventually converges to a single peak and resolves solution ambiguity.

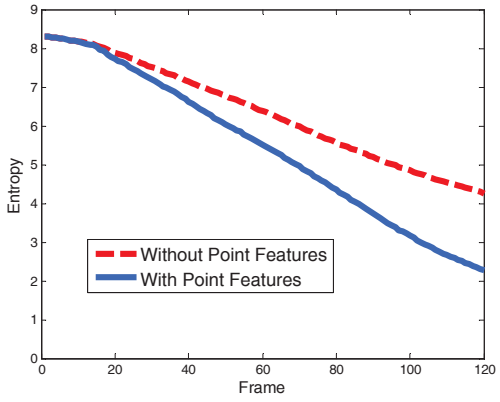


Figure 6.12: Entropy trajectory with and without point features. Using point features leads to a lower entropy.

6.5.4 Solution Disambiguation

In the HMB approach, the decomposition of a homography matrix has the form $H = R + TN_0^T/d_0$ (R and T are camera rotation and translation, and N_0 and d_0 is defined in Eq. 6.2). The decomposition can generate two physically possible solutions because T/d_0 and N_0 are interchangeable. The disambiguation of the two solutions is usually made by choosing the solution that has better consistency with solutions in previous frames [Ma, 2004; Cobzas et al., 2009]. When the camera moves on a line without any rotation (which is a common motion pattern for a mobile robot), for example, on the line along the camera's z-axis, it will be impossible to select the correct solution from the two solutions because both solutions will ideally keep unchanged all the time. In this case, no filtering process such as Kalman Filter can resolve the ambiguity. However, with the help of the points features that lie outside of the planar region feature, the LSMGS method converges well to the correct solution (Fig. 6.10).

In order to better demonstrate that the proposed LSMGS method avoids solution ambiguity, we investigate the evolution process of the probability distribution over normal parameters. When point features and the essential matrix constraint are disabled (L_t^e in Eq. 6.23 is fixed to 1), there will be two peaks in the probability distribution; when they

are enabled, the distribution will eventually converge to have only one peak (Fig. 6.11). Fig. 6.12 shows the entropy trajectory with or without point features, where the entropy is lower when point features are used.

6.5.5 Evaluation based on 3D Reconstruction Errors

Given the estimated camera rotation δR_t and translation δT_t, at each frame $t > 0$, the 3D positions of the selected evaluation point features are estimated by adapted linear triangulation [Hartley and Zisserman, 2003], for the HMB, EMB, Bundler, and LSMGS methods.

For the set of selected evaluation points, we normalize the sum of the distance between their 3D positions to 1, for both the ground truth data and the estimation results from HMB, EMB, Bundler, and LSMGS, which eliminates the different scaling factor. Let $P_1, P_2, ..., P_m$ be the normalized ground truth 3D positions for the selected points, and $\hat{P}_1, \hat{P}_2, ..., \hat{P}_m$ be the corresponding estimated 3D positions from HMB, EMB, Bundler, or LSMGS. The 3D reconstruction error is defined as

$$E^{3D} = \sum_{i,j} |\|P_i - P_j\| - \|\hat{P}_i - \hat{P}_j\|| \tag{6.29}$$

which is the sum of the absolute difference between each two points' 3D distance in the ground truth data and in the HMB, EMB, Bundler, or LSMGS estimation results.

The reconstruction errors are shown in Fig. 6.13 and Fig. 6.14, with KLT and SIFT features respectively. In the KLT feature case, our method (LSMGS) gets lower reconstruction errors than HMB, EMB, and Bundler, for all the four datasets. In the SIFT feature case, our method gets lower errors for the Twinings, Pantech, and Ambulance datasets, but it performs worse for the Yorkshire dataset (probably because the SIFT features are in high quality for this dataset because of its good texture). The EMB method gets large error mean in overall and its errors have significantly larger variance than the other methods, which indicates that the EMB approach may not be well suited for the situation where the camera motion is small or where the images are noisy. In order to better see the difference between

3D Reconstruction Errors (with KLT features)

Error	HMB	EMB	Bundler	LSMGS
Yorkshire	0.073±0.028	0.281±0.222	0.193±0.076	**0.059±0.028**
Twinings	0.133±0.079	0.248±0.183	0.642±0.214	**0.093±0.048**
Pantech	0.116±0.075	0.417±0.227	0.358±0.138	**0.072±0.027**
Ambulance	0.070±0.047	0.325±0.231	0.195±0.075	**0.059±0.013**

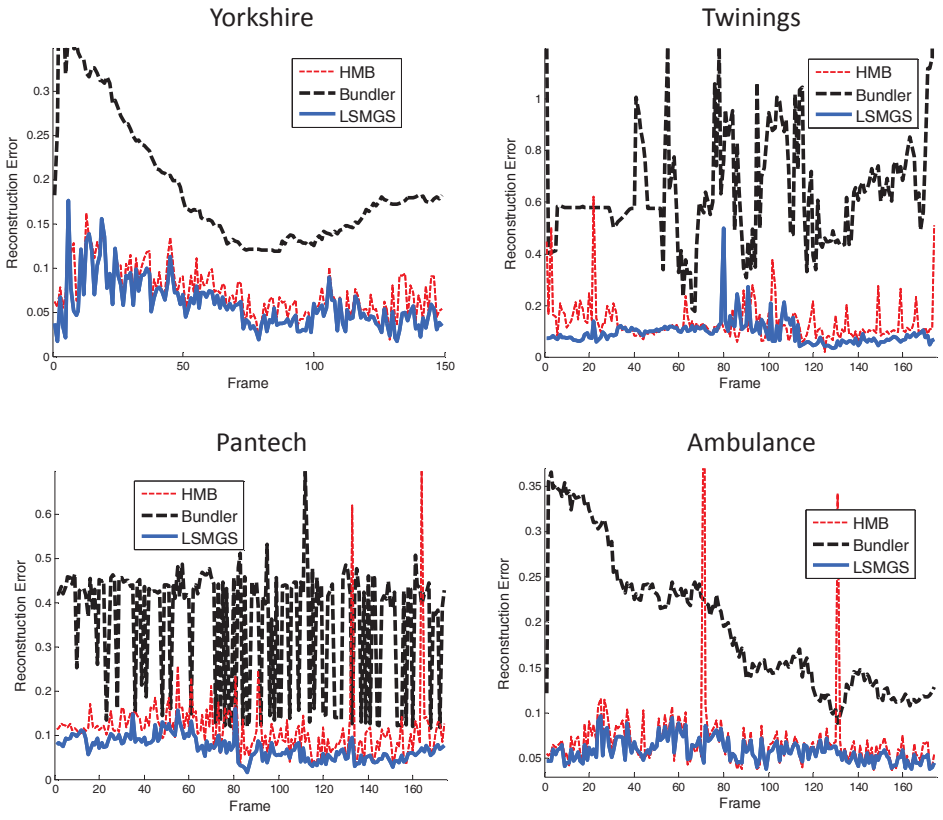


Figure 6.13: 3D reconstruction errors (with KLT features). The table shows the quantitative 3D reconstruction errors for HMB, EMB, Bundler, and our method (LSMGS). The errors for EMB have significantly larger variance than the other methods, and the EMB results are shown in only the table and not in the graphs. The table shows average error per frame with $\pm$ standard deviation. The point features used for EMB, Bundler, and LSMGS are KLT features.

3D Reconstruction Errors (with SIFT features)

Error	HMB	EMB	Bundler	LSMGS
Yorkshire	0.073±0.028	0.282±0.205	**0.045±0.031**	0.057±0.028
Twinings	0.133±0.079	0.252±0.211	0.207±0.232	**0.093±0.048**
Pantech	0.116±0.075	0.405±0.232	0.209±0.109	**0.072±0.027**
Ambulance	0.070±0.047	0.287±0.224	0.112±0.055	**0.053±0.012**

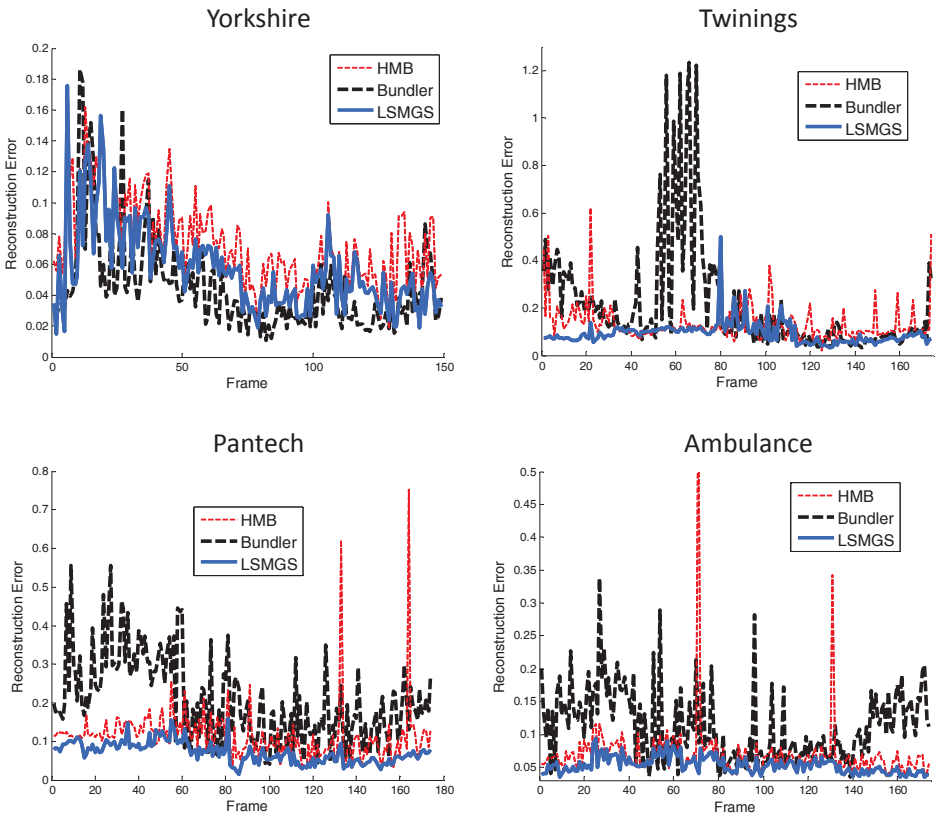


Figure 6.14: 3D reconstruction errors (with SIFT features). The table shows the quantitative 3D reconstruction errors for HMB, EMB, Bundler, and our method (LSMGS). The errors for EMB have significantly larger variance than the other methods, and the EMB results are shown in only the table and not in the graphs. The point features used for EMB, Bundler, and LSMGS are SIFT features.

HMB, Bundler, and LSMGS, the EMB errors are shown only in the table and not in the graphs.

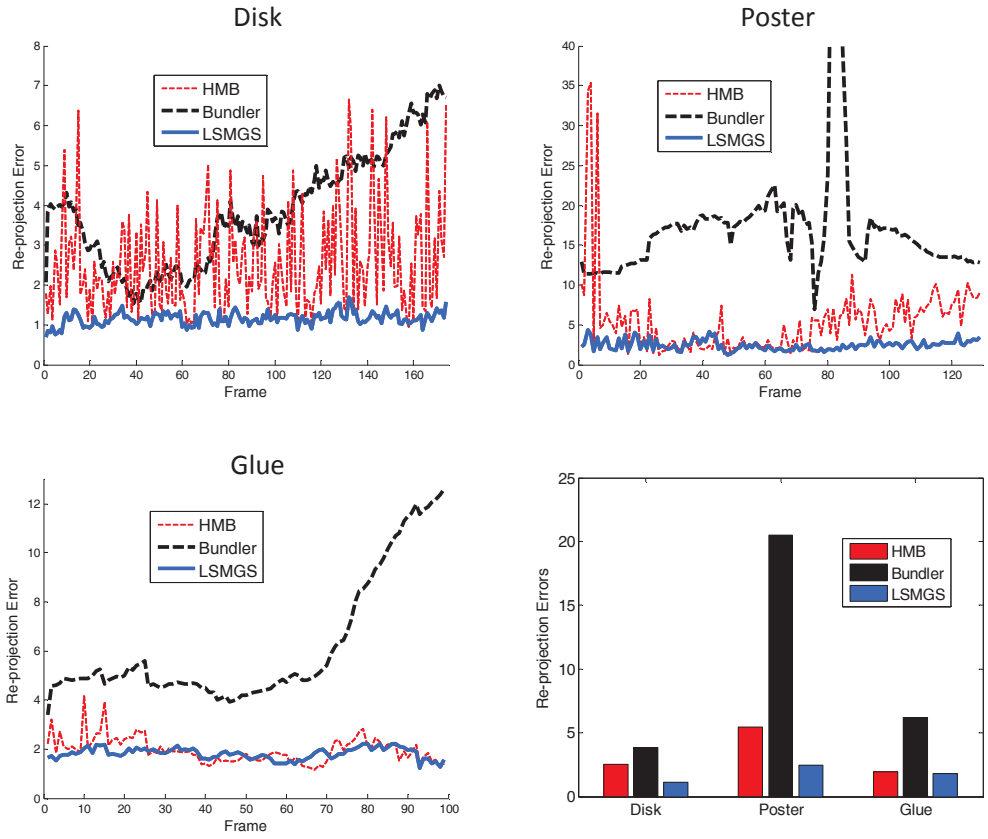


Figure 6.15: 2D re-projection errors (with KLT features). The reconstructed 3D points are re-projected in each frame, and the average of the errors between the 2D ground truth locations and the re-projected locations is defined as the re-projection error for that frame. The point features used for EMB, Bundler, and LSMGS are KLT features. The last figure shows the error mean for each dataset.

6.5.6 Evaluation based on 2D Re-projection Errors

For the last three datasets, it is difficult to measure the ground truth 3D positions for the evaluation points. Hence, they are evaluated based on their 2D re-projection errors. For each dataset, the 3D positions of the evaluation points in the reference frame are obtained.

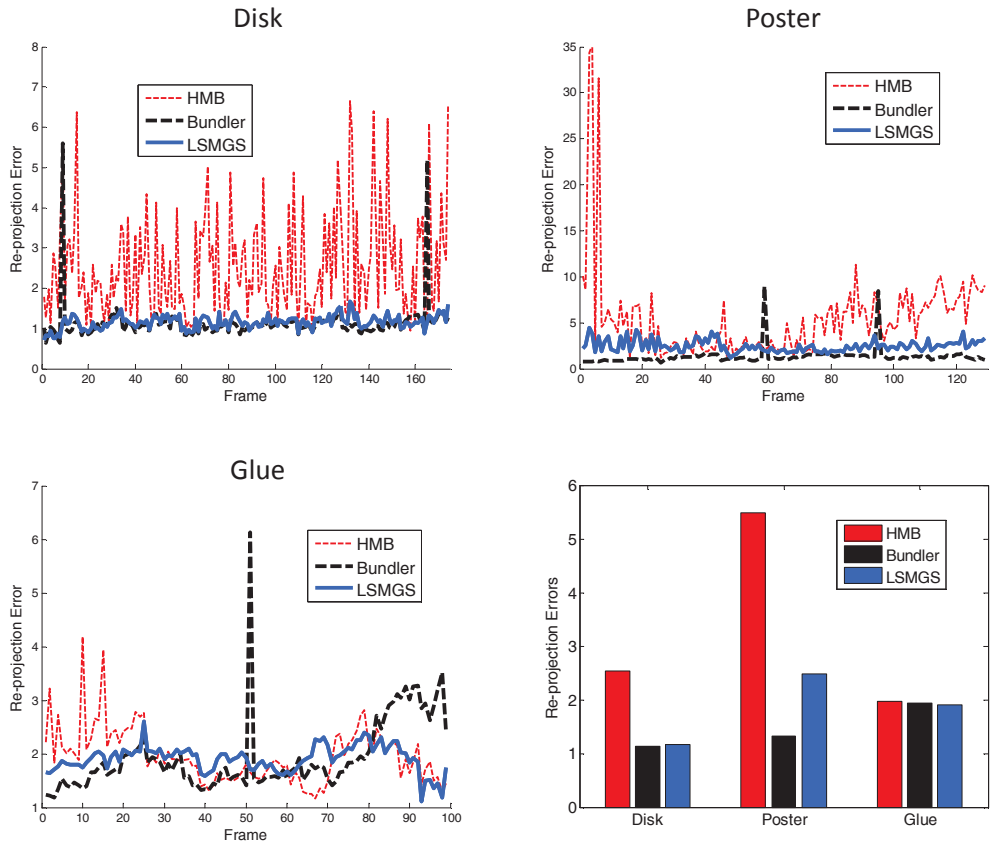


Figure 6.16: 2D re-projection errors (with SIFT features). The reconstructed 3D points are re-projected in each frame, and the average of the errors between the 2D ground truth locations and the re-projected locations is defined as the re-projection error in pixel for that frame. The point features used for EMB, Bundler, and LSMGS are SIFT features. The last figure shows the error mean for each dataset.

This is done by first calculating the 3D positions in each frame, then transforming them in the reference frame, and then taking the average of the 3D positions for each point. Once we have the estimated 3D points in the reference frame, we re-project them back into each frame. The average of the errors between their 2D ground truth locations and their re-projected locations is defined as the re-projection error,

$$E^{2D} = \frac{1}{n^p} \sum_{i=1}^{n^p} \|\hat{p}_i - p_i\| \tag{6.30}$$

where n^p is the number of evaluation points, and $\hat{p}_i$ is the corresponding re-projected point for p_i.

Fig. 6.15 and Fig. 6.16 show the 2D re-projection errors, with KLT and SIFT features, respectively. In the KLT feature case, LSMGS performs better than HMB and Bundler for all the three datasets. Bundler gets the worst results due to KLT feature drift. In contrast, LSMGS is very robust to feature drift, thanks to the help of the planar region feature. In the SIFT feature case, for the Disk and Glue datasets, LSMGS gets very similar overall accuracy to Bundler. Its accuracy is lower for the Poster dataset compared with Bundler, but the difference is only about one pixel.

Note that, in the Glue dataset, the planar region feature used for pose estimation lies on a non-planar surface. When the viewing direction is beyond a certain range (about $\pi/6$) around the feature's surface normal, the tracking fails because the feature's images in different frames cannot be approximated by a homography matrix anymore. In this case, the pose estimation also fails. But when the viewing direction is within the range (less than $\pi/6$), we find that the feature can be tracked well and the system works as well as for the other datasets (Fig. 6.15 and Fig. 6.16).

Chapter 7

3D Model Construction

7.1 Introduction

After the camera motion is estimated, we will be able to obtain the 3D object models based on tracked features. This chapter describes preliminary results for 3D3D model construction in the OSH.

Various 3D models have been explored to represent an object such as point clouds, voxels, meshes, and depth maps. Although these models can approximate arbitrary object shapes well, they do not automatically identify compact 3D object parts/surfaces (for example, these models will end up getting a far more complex representation than a compact 6-face model for a cubic box). In addition, construction of these models typically requires stereo vision [Seitz et al., 2006]. While stereo vision can provide one type of cue for 3D model construction, in this work we solve the problem based on another type of cue: motion cue from a monocular camera. A monocular camera is much easier to set up than stereo cameras, and we simply use a webcam in our experiments.

In Gallup et al. [2010], piecewise planar and non-planar regions are used to represent indoor and outdoor scenes using stereo vision. In our system we use similar compact surface-based models. However, the model construction in Gallup et al. [2010] highly re-

lies on appearance information as well as geometry. Although this approach fits well for indoor or outdoor scenes, it is not suitable to segment the surfaces on a single object. This is because different objects/regions in an indoor/outdoor scene usually have different appearance, but different surfaces on a single object tend to have similar rather than distinct appearance. Thus, in our method, we construct surface-based models based on only geometric surface smoothness.

A lot of state-of-the-art works [Snavely et al., 2008; Newcombe and Davison, 2010; Klein and Murray, 2007; Pollefeys et al., 2004] construct 3D object models from monocular images. These works usually give a large set of pixels or point features with estimated 3D positions, but do not generate higher-level surface-based models. In addition, these works use only point features and do not take advantage of planar region features. Planar features are used in Cobzas et al. [2009] to estimate the 3D poses between different object surfaces, but manual initialization is needed to mark each surface's boundary.

We present a method to construct compact 3D object models from monocular image sequences captured by a webcam. The constructed model contains a small number of surfaces and the boundary of each surface is automatically identified. The full 3D object structure is recovered from the camera motion as a collection of triangles in 3D space. The normal of each triangle is modeled as a Gaussian distribution, and based on the triangles' local geometric continuity, the final model is constructed as a compact set of surfaces through maximum a posteriori estimation.

7.2 Structure Recovery

In the reference frame I_0, we detect a set of local interest point features p_0, and track them as p_t in frame I_t. Based on the camera motion parameters δR_t and δT_t (see Eq. 6.28 in Chapter 6), for each point feature we can draw two rays in the reference camera space at time 0 and in the current camera space at time t, where each ray passes the optical center and the point feature in the image plane. We then simply take the average of the two closest

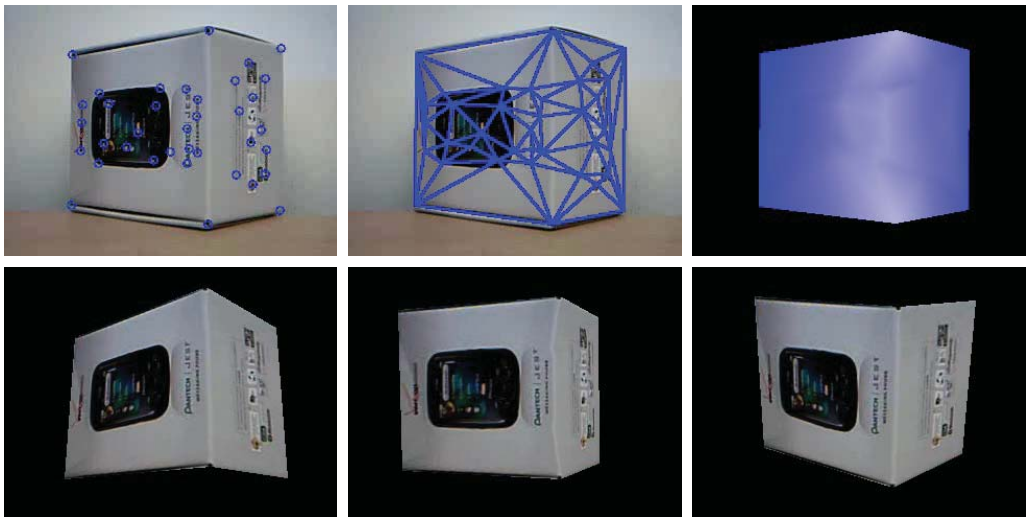


Figure 7.1: Object reconstruction. Top left: tracked point features, top middle: triangular meshes from the point features, top right: reconstructed depth map. The bottom row shows reconstructed images at different viewpoints based on the triangle positions with texture mapping. The bottom middle image has distortions around the middle vertical line on the object because the corresponding triangles cover areas that do not belong to a single planar surface (the bottom left and right images do not have significant distortions under these poses).

3D points on the two rays as the 3D position estimation of that point feature in the reference camera space.

Based on the feature points p_0 at time 0, the image I_0 is divided into triangular meshes using the Delaunay triangulation algorithm. The pixels within a triangle are assumed to be on the same 3D plane as that triangle. The 3D scene structure is then created as a collection of these 3D triangular surfaces. The texture of the 3D scene is mapped from the reference image I_0 by a linear transformation. Some reconstructed images are shown in Fig. 7.1 for the example video we have used in Chapter 6.

The position and normal of each triangle is modeled as a Gaussian distribution and is updated at each time step. The object structure is now represented as a collection of triangles $\mathbb{T} = \{\mathbb{T}_j\}$ in 3D space. The next step is to build a compact model of the object containing a small number of surfaces, where each surface has similar normals at points

inside the surface and two adjacent surfaces have dissimilar normals.

7.2.1 Probabilistic Framework

The task of building a compact object model is to seek for $\mathbb{M} = \{n_s, \mathbb{S}_1, ..., \mathbb{S}_{n_s}\}$ where $\mathbb{S}_i$ $(1 \leq i \leq n_s)$ are the object surfaces (having a boundary and a 3D normal) and n_s is the number of the surfaces, such that with a small number n_s, the model $\mathbb{M}$ explains well the observation of the triangles $\mathbb{T}$. The task now becomes finding $\hat{\mathbb{M}}$ by maximum a posteriori estimation,

$$\hat{\mathbb{M}} = \arg\max_{\mathbb{M}} Pr(\mathbb{M}|\mathbb{T}) \tag{7.1}$$

$$\text{where} \qquad Pr(\mathbb{M}|\mathbb{T}) \propto Pr(\mathbb{M})Pr(\mathbb{T}|\mathbb{M}). \tag{7.2}$$

To explore all configurations in $\mathbb{M}$ would be intractable. Instead, we take a bottom-up grouping step to reduce the space of $\mathbb{M}$. Neighboring triangles are grouped together if the absolute value of the dot product of their normals is above a threshold ρ. Each ρ leads to a configuration of $\mathbb{M}$. By adjusting ρ, we get a set of candidate configurations. Thus the number of candidate configurations has reduced to the number of discretization of ρ. Each candidate configuration contains a set of surfaces, where each surface's boundary is the outermost contour of all associated triangles and the surface's 3D normal is the average of the normals of all associated triangles.

A good configuration of $\mathbb{M}$ should have three constraints: (1) n_s is small, (2) each surface and its associated triangles have similar normals, and (3) adjacent surfaces have dissimilar normals. To expose the first constraint, we design the prior function in Eq. 7.2 as $Pr(\mathbb{M}) \propto 1 + \beta/n_s$ where β is pre-selected positive constant. Intuitively, the prior function prefers to select models with a low number of surfaces.

7.2.2 Likelihood Formulation

Now we formulate the likelihood function $Pr(\mathbb{T}|\mathbb{M})$ according to constraints (2) and (3). Let $\mathbb{T}_j$ be a triangle with area a_j. The similarity score between the triangle and its associated surface is denoted as $d_j \in [0,1]$, which is the absolute value of the dot product between the triangle's normal and the normal of the triangle's associated surface. Let $\mathbb{S}_i$ be a surface with area $\mathbb{A}_i$. The similarity score between two adjacent surfaces $\mathbb{S}_i$ and $\mathbb{S}_k$ is denoted as $\mathbb{D}_{ik} \in [0,1]$, which is the absolute value of the dot product between the surfaces' normals. The likelihood function is formulated as

$$Pr(\mathbb{T}|\mathbb{M}) \propto \frac{\sum_j a_j d_j}{\sum_j a_j}\Big(1 - \frac{\sum_{\{i,k:\mathbb{S}_i \leftrightarrow \mathbb{S}_k\}} \sqrt{\mathbb{A}_i \mathbb{A}_k}\mathbb{D}_{ik}}{\sum_{\{i,k:\mathbb{S}_i \leftrightarrow \mathbb{S}_k\}} \sqrt{\mathbb{A}_i \mathbb{A}_k}}\Big) \tag{7.3}$$

where the symbol $\leftrightarrow$ denotes the adjacency of two surfaces. The first term in the right-hand side evaluates the normal similarity between each triangle and its associated surface, and the second term evaluates the normal dissimilarity between adjacent surfaces.

7.3 Demos

When the 3D triangle based model is obtained for the object, we can generate various object views corresponding to different virtual poses. Some generated views with virtual poses for these datasets are shown in Fig. 7.2.

Fig. 7.3 shows the constructed compact models for the four datasets. Each contour corresponds to an object surface (surfaces with very small area have been removed), and the surface numbers are 4, 2, 3, and 2 for dataset Ambulance, Pantech, Twinings, and Yorkshire, respectively. Each of these surfaces corresponds to a 2D3D planar component in the OSH. In the Pantech dataset, the top surface and the bottom left surface are grouped together, primarily because the triangles around the intersection of the two surfaces cover across both surfaces and hence weaken their geometric dissimilarity.

Given the estimated camera pose, we have used only point features to build these

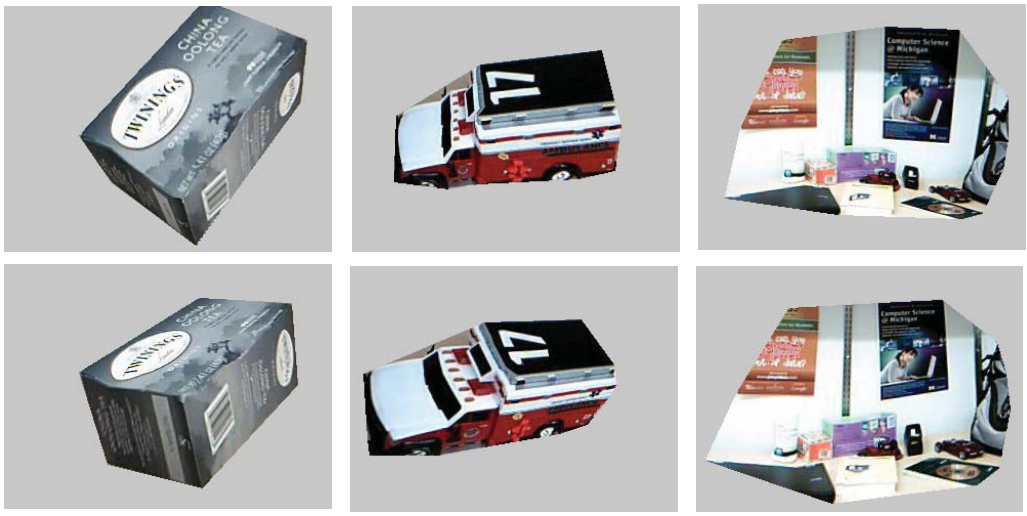


Figure 7.2: Reconstructed images. These images are reconstructed by setting arbitrary virtual viewing poses, and they do not correspond to any pose observed in the original videos. The deformations on object surfaces are not caused by incorrect pose estimation, but by (1) some triangles in Delaunay triangulation cover across different object surfaces and (2) some point features have drifted during tracking.

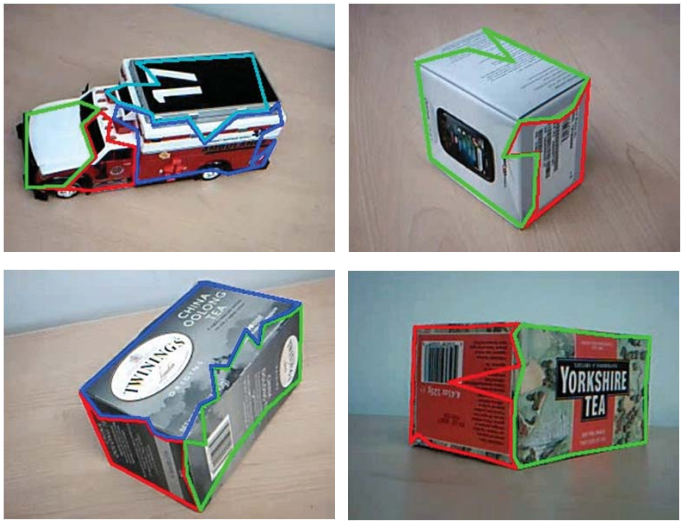


Figure 7.3: Constructed compact object models. Each model consists of a small set of surfaces in 3D space (shown as colored contours). The number of surfaces and their boundaries are automatically identified.

compact models. Since the actual boundaries of the components may not have enough detected point features, it is difficult to find the exact component boundaries using only these tracked points. One future direction to improve the results is to combine edge/contour information. For example, we can check whether the triangles on one side of a contour fragment have similar normals as the triangles on the other side. If so, the contour fragment will be labeled as a negative fragment; otherwise, it is a positive fragment. Then the positive fragments can be connected to form final closed boundaries.

Chapter 8

Conclusion and Future Directions

8.1 Summary

In this thesis, we have proposed the Object Semantic Hierarchy (OSH) and have described a few key steps towards building the OSH.

The models of objects and the surrounding background in the OSH are built in a progressive manner. Initially everything in the sensory input is treated as noise; then the agent identifies the background and builds a constant model for the background, where dynamic foreground objects are treated as noise; then foreground object models are constructed by identifying invariants in the remaining noise. For the background and each foreground object, their representations evolve from 2D views, to 2D planar surfaces in 3D space, then to full 3D models.

To build the 2D model for a foreground object, we have presented a novel MSMS (Motor Signal based Motion Segmentation) method [Xu et al., 2011] to separate the foreground object from the background based on motion cues. In contrast to existing approaches which use only sensor images for object segmentation, our method exploits information from both sensor images and motor signals. The use of motor signals allows the agent to achieve fast and robust segmentation.

We have evaluated the proposed MSMS method against the pixel-level background subtraction and RANSAC-based homography fitting approaches, and the results show that the proposed method gives better segmentation results on datasets with large translation, rotation, scaling, and illumination changes.

To build the 3D model for a separated foreground object, we have presented a new LSMGS (from Local Structure to Motion then to Global Structure) method for the recovery of camera pose and object structure. This method starts by estimating the normal (*Local Structure*) for a single tracked planar region feature, then obtain the dynamic camera pose (*Motion*) based on the estimated normal, then recover the object/scene structure (*Global Structure*) based on the camera poses. As opposed to existing approaches that use either point features or planar region features and apply either the homography matrix based constraint or the essential matrix based constraint, our method uses both types of features and applies both types of constraints.

We have evaluated the LSMGS method by comparing it against the HMB, EMB, and Bundler approaches on various datasets. Comparison results show that our method provides better estimation accuracy and avoids solution ambiguity, which suggests that fusing different types of features and geometric constraints improves the quality of the recovered camera pose and object structure.

To summarize, we have presented a multi-layer representation for objects and the surrounding background, and have described solutions to some key problems in building these representations, including feature tracking, 2D object segmentation, 3D pose estimation, and 3D structure recovery.

8.2 Future Work

Below we discuss a few ways to extend the current work, including combination of other cues to build 3D3D object models, incorporation of data from other types of sensors, investigation of situations containing multiple foreground objects, and evaluation of the whole

OSH framework.

Combination of other cues to build 3D3D models

We have used mainly geometric information to build 3D3D models after camera motion is estimated, which provides the agent compact object representations. Other cues such as information from edge/contour detection, static image segmentation (such as [Xu et al., 2008]), and common-sense geometric context (such as [Bao et al., 2010]) will help build more robust models.

Edge/contour and image segmentation will give better boundaries between different planar components in the 3D3D model. For each contour fragment, we can examine whether the region on one side has same or different normal compared to the region on the other side, and then determine whether this contour fragment belongs to component boundary. Image segmentation can also help get better component boundaries because the contours of the segmented regions tend to lie on edge pixels.

Common-sense geometric context will facilitate object pose estimation by applying prior knowledge of pose relations. For example, the normal of the ground in a typical image is usually in the vertical direction, and the surfaces of a box on the ground are usually parallel or perpendicular to the ground surface. With this prior knowledge, the searching space for the box pose will be greatly reduced.

Incorporation of data from other types of sensors

Camera images capture rich information about the appearance of the environment and provide a huge amount of details. We have used monocular vision sensors to infer depth information, which relies on robot/object motion. In the case where there is no robot or object motion, stereo cameras can be used for depth inference (such as [Murarka et al., 2008]). Time-of-flight cameras have made fast progress these days, and can also be used for distance measurement.

Laser-range finders have been widely used for indoor and outdoor robot navigation

(for example, [Kuipers et al., 2004] and [Beeson et al., 2010]), and allow a robot to have easy access to depth information. Fusing laser and vision sensors will help build high-quality 3D models of the environment for navigation.

Investigation of situations containing multiple foreground objects

As the OSH itself has no specific limit for the number of foreground objects, the current implementation has focused on building models for a single foreground object. Extension to handle multiple objects will be similar to the single object case, but how the agent should allocate resources for each object needs to be considered.

Evaluation of the whole framework

To solve the key problems in building the OSH, we have evaluated our proposed methods for feature tracking, foreground segmentation, and recovery of object pose and structure. These methods yet need to be eventually evaluated in the OSH framework as a whole.

Possible ways for the whole system evaluation include:

(a) *Evaluation by reconstruction*: reconstruct an image sequence from the background/object models in the OSH, and show how reconstruction errors and model complexity change across different layers.

(b) *Evaluation by recognition*: recognize objects in new images based on the constructed object models (such as [Xu and Kuipers, 2011]) in the OSH, and show how multiple representations can help improve recognition accuracy.

(c) *Evaluation by object manipulation*: let the robot manipulate objects using the constructed object models in the OSH, and investigate the success rate for object manipulation (such as [Mugan and Kuipers, 2009]).

Bibliography

A. Agarwal, CV Jawahar, and PJ Narayanan. A survey of planar homography estimation techniques. *Centre for Visual Information Technology, Tech. Rep. IIIT/TR/2005/12*, 2005.

P. Azzari, L. Di Stefano, and A. Bevilacqua. An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a PTZ camera. *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 511–516, 2005.

S. Baker and I. Matthews. Lucas-kanade 20 years on: a unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

S.Y.Z. Bao, M. Sun, and S. Savarese. Toward coherent object detection and scene layout understanding. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72, 2010.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded-up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

P. Beeson, J. Modayil, and B. Kuipers. Factoring the mapping problem: Mobile robot map-building in the Hybrid Spatial Semantic Hierarachy. *International Journal of Robotics Research*, 29(4):428–459, 2010. doi: 10.1177/0278364909100586.

I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological review*, 94:115–147, 1987.

M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996. ISSN 1077-3142.

G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 710–715, 2005.

F. Bourel, C.C. Chibelushi, and A.A. Low. Robust facial feature tracking. *British Machine Vision Conference*, 1:232–241, 2000.

M. Brown and D.G. Lowe. Recognising panoramas. In *IEEE International Conference on Computer Vision*, page 1218, 2003.

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. *European Conference on Computer Vision*, pages 282–295, 2010.

H.H. Bulthoff and S. Edelman. Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceedings of the National Academy of Sciences of the United States of America*, 89(1):60, 1992.

M.C. Burl and P. Perona. Recognition of planar object classes. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 223–230, 1996.

J. Canny. A computational approach to edge detection. *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 184–203, 1986.

D. Cobzas, M. Jagersand, and P. Sturm. 3D SSD tracking with estimated 3d planes. *Image and Vision Computing*, 27(1-2):69–79, 2009.

D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 564–575, 2003.

M. Cummins and P. Newman. Highly scalable appearance-only SLAM–FAB-MAP 2.0. In *Robotics Science and Systems*, 2009.

A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1052–1067, 2007.

O. Deniz, G. Bueno, E. Bermejo, and R. Sukthankar. Fast and accurate global motion compensation. *Pattern Recognition*, 2010. ISSN 0031-3203.

I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

R.O. Duda and P.E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

E. Eade and T. Drummond. Scalable monocular SLAM. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 469–476, 2006.

A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *European Conference on Computer Vision*, pages 751–767, 2000.

C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. *Photogrammetric Computer Vision*, 2, 2006.

B. Epshtein and S. Ullman. Semantic hierarchies for recognizing objects and parts. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 594–611, 2006.

P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 2003.

V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67 (2):159–188, 2006.

M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, pages 23–37. Springer, 1995.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

J. Gall, B. Rosenhahn, and H.P. Seidel. Drift-free tracking of rigid and articulated objects. *CVPR*, 2008a.

J. Gall, B. Rosenhahn, and H.P. Seidel. Drift-free tracking of rigid and articulated objects. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008b.

C. Galletti and P. Fattori. Neuronal mechanisms for detection of motion in the field of view. *Neuropsychologia*, 41(13):1717–1727, 2003. ISSN 0028-3932.

D. Gallup, J.M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1418–1425, 2010.

L. Goshen and I. Shimshoni. Guided sampling via weak motion models and outlier sample generation for epipolar geometry estimation. *International Journal of Computer Vision*, 80(2):275–288, 2008. ISSN 0920-5691.

H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: learning where the object might be. *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

K. Grauman and T. Darrell. The pyramid match kernel: discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*, volume 2, pages 1458–1465, 2005.

K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:19–25, 2006.

M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2141–2148, 2010.

J.J. Guerrero and C. Sagues. Robust line matching and estimate of homographies simultaneously. *Pattern Recognition and Image Analysis*, pages 297–307, 2003.

V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3129–3136, 2010.

M. Han, W. Xu, and Y. Gong. Video object segmentation by motion-based sequential feature clustering. In *Proceedings of the 14th annual ACM International Conference on Multimedia*, page 782, 2006.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

E. Hayman and J.O. Eklundh. Statistical background subtraction for a mobile observer. *IEEE International Conference on Computer Vision*, 1:67–74, 2003.

J.E. Hummel. Where view-based theories break down: The role of structure in shape perception and object recognition. *Cognitive dynamics: Conceptual change in humans and machines*, pages 157–185, 2000.

M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

K.H. James, G.K. Humphrey, T. Vilis, B. Corrie, R. Baddour, and M.A. Goodale. Active and passive learning of three-dimensional object structure within an immersive virtual reality environment. *Behavior Research Methods Instruments and Computers*, 34(3): 383–390, 2002.

T. Joachims. Transductive learning via spectral graph partitioning. *International Conference on Machine Learning*, 20(1):290, 2003.

P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. *European Workshop on Advanced Video-based Surveillance Systems*, 1(3), 2001.

M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

H. Kim, E. Murphy-Chutorian, and J. Triesch. Semi-autonomous learning of objects. *Workshops in IEEE Conference on Computer Vision and Pattern Recognition*, pages 145–145, 2006.

G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, 2007.

J. Knight and I. Reid. Automated alignment of robotic pan-tilt camera units using vision. *International Journal of Computer Vision*, 68(3):219–237, 2006.

T. Ko, S. Soatto, and D. Estrin. Warping background subtraction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1331–1338, 2010.

B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1-2):191–233, 2000.

B. Kuipers. Drinking from the firehose of experience. *Artificial Intelligence in Medicine*, 44(2):155–170, 2008.

B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. *IEEE International Conference on Robotics and Automation*, 5:4845–4851, 2004.

A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3D object recognition. *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

J. Kwon and K.M. Lee. Monocular slam with locally planar landmarks via geometric rao-blackwellized particle filtering on lie groups. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1522–1529, 2010.

Y.J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. *IEEE International Conference on Computer Vision*, 2011.

N.K. Logothetis and D.L. Sheinberg. Visual object recognition. *Annual Review of Neuroscience*, 1996.

H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

D.G. Lowe. Local feature view clustering for 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, 2001.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Y. Ma. *An invitation to 3-D vision: From images to geometric models*. Springer Verlag, 2004.

D. Marr and HK Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. of the Royal Society-London B*, 1978.

M. Marszalek and C. Schmid. Semantic hierarchies for visual object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2007.

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

S.J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42–56, 2000.

A. Mittal and D. Huttenlocher. Scene modeling for wide area surveillance and image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, page 2160, 2000.

A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 2004.

J. Modayil and B. Kuipers. Bootstrap learning for object discovery. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 742–747, 2004.

J. Modayil and B. Kuipers. Autonomous shape model learning for object localization and recognition. *IEEE International Conference on Robotics and Automation*, pages 2991–2996, 2006.

J. Modayil and B. Kuipers. The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems*, 56(11):879–890, 2008.

N.D. Molton, A.J. Davison, and I.D. Reid. Locally planar patch features for real-time structure from motion. In *British Machine Vision Conference*, 2004.

A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. *IEEE International Conference on Computer Vision*, pages 1305–1312, 2008.

J. Mugan and B. Kuipers. Autonomously learning an action hierarchy using a learned qualitative state representation. *IJCAI*, 2009.

A. Murarka, M. Sridharan, and B. Kuipers. Detecting obstacles and drop-offs using stereo and motion cues for safe local motion. *IROS*, 2008.

R.A. Newcombe and A.J. Davison. Live dense reconstruction with a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1498–1505, 2010.

D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6), 2004.

D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

D. Parikh and T. Chen. Unsupervised learning of hierarchical semantics of objects (hSOs). In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

D. Pierce and B.J. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92(1-2):169–227, 1997.

M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59 (3):207–232, 2004.

A.R. Pope and D.G. Lowe. Probabilistic models of appearance for 3-D object recognition. *International Journal of Computer Vision*, 40(2):149–167, 2000.

M. Pressigout and E. Marchand. Real-time 3D model-based tracking: Combining edge and texture information. *IEEE International Conference on Robotics and Automation*, pages 2726–2731, 2006.

X. Ren and C. Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3137–3144, 2010.

X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, 2006.

S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. *International Conference on Computer Vision*, pages 1–8, 2007.

H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. *IEEE Conference on Computer Vision and Pattern Recognition*, page 1746, 2000.

S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528, 2006.

Y. Sheikh, O. Javed, and T. Kanade. Background subtraction for freely moving cameras. In *International Conference on Computer Vision*, pages 1219–1225, 2009.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

J. Shi and C. Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision*, volume 1, pages 503–510, 2005.

G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics Science and Systems Conference*, 2009.

J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *International Journal of Computer Vision*, 67(2):189–210, 2006.

J. Sivic, B.C. Russell, A. Zisserman, W.T. Freeman, and A.A. Efros. Unsupervised discovery of visual object class hierarchies. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

N. Snavely, S.M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. ISSN 0920-5691.

E.S. Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:246–252, 1999.

C. Stauffer and WEL Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

P. Sturm. Algorithms for plane-based pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 706–711, 2000.

E.B. Sudderth, A. Torralba, W.T. Freeman, and A.S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *International Conference on Computer Vision*, volume 2, pages 1331–1338, 2005.

R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 2:104, 2006.

PHS Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

S. Tran and L. Davis. Robust object tracking with regional affine invariant features. *International Conference on Computer Vision*, 2007.

B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *Vision algorithms: theory and practice*, pages 153–177, 2000.

G. Tsai, C. Xu, J. Liu, and B. Kuipers. Real-time indoor scene understanding using bayesian filtering with motion cues. *International Conference on Computer Vision*, 2011.

H. Uemura, S. Ishikawa, and K. Mikolajczyk. Feature tracking and motion compensation for action recognition. In *British Machine Vision Conference*, 2008.

S. Ullman. Three-dimensional object recognition based on the combination of views. *Cognition*, 67(1-2):21–44, 1998.

L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. *The 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2004.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:I–511, 2001.

J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. *European Conference on Computer Vision*, 1842:18–32, 2000.

C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (7):780–785, 1997.

J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1644–1659, 2005. ISSN 0162-8828.

C. Xu and B. Kuipers. Construction of the Object Semantic Hierarchy. *Fifth International Cognitive Vision Workshop*, 2009.

C. Xu and B. Kuipers. Towards the Object Semantic Hierarchy. *International Conference On Development and Learning*, pages 39–45, 2010.

C. Xu and B. Kuipers. Object detection using principal contour fragments. *Canadian Conference on Computer and Robot Vision*, pages 363–370, 2011.

C. Xu, Y.J. Lee, and B. Kuipers. Ray-based color image segmentation. *Canadian Conference on Computer and Robot Vision*, pages 79–86, 2008.

C. Xu, B. Kuipers, and A. Murarka. 3D pose estimation for planes. *ICCV Workshop on 3D Representation for Recognition (3dRR-09)*, pages 673–680, 2009.

C. Xu, J. Liu, and B. Kuipers. Motion segmentation by learning homography matrices from motor signals. *Canadian Conference on Computer and Robot Vision*, pages 316–323, 2011.

A. Yilmaz, O. Javed, and M. Shah. Object tracking: a survey. *ACM Computing Surveys*, 38 (4):13, 2006.

Z. Yin and R.T. Collins. Shape constrained figure-ground segmentation and tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–738, 2009.

Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

J. Zhong and S. Sclaroff. Segmenting foreground objects from a dynamic textured background via a robust Kalman Filter. In *IEEE International Conference on Computer Vision*, volume 2, page 44, 2003.

T. Zinsser, C. Grassl, and H. Niemann. Efficient feature tracking for long video sequences. *Pattern Recognition: 26th DAGM Symposium*, pages 326–333, 2004.