D I S S E R T A T I O N

From Images to Objects –
Vision for Cognitive Robotic Experimentation

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Institut für Automatisierungs- und Regelungstechnik (E376)

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl.-Ing. Johann Prankl
geb. am 9.3.1977
Matr. Nr.: 9825083
Krügling 7, 3250 Wieselburg

Wien, im September 2011

Abstract

In recent years, research has focused on the development of intelligent robots that are aware of their environment and begin to link perceptions to the meaning of objects attributed by humans. One way to bridge the gap between robotic and human understanding of their common environment is the concept of affordances, which links the perception of objects to the action of robots. Allowing a robot to learn object affordances from a human tutor or autonomously requires a representation of what it knows, so that it can reason about what it can learn, how to act so as to learn it, execute those actions and then learn to fill its knowledge gaps.

This work presents a perception system for a cognitive robot, which represents the structure of objects to link perception to robot actions. To manifest what objects afford to the robot, a model based on piecewise planar surface patches is proposed. Planar patches are detected from tracked interest points in image sequences. For this we formalize model selection with Minimal Description Length (MDL) in an incremental manner. In each iteration tracked planes and new planes computed from randomly sampled interest points are evaluated. The hypotheses that best explain the scene are retained and their supporting points are marked so that in the next iteration random sampling is guided to unexplained points. Hence, it is possible to represent the remaining finer details of the scene.

Planar patches are stored in a spatio-temporal graph and tracked to subsequent images. After reconstruction of the planes the 3D motion is analyzed and initial object hypotheses are created. These object hypotheses are verified from the robot by pushing them. In case planar patches start moving independently a split event is triggered, the spatio-temporal object graph is traced back and visible planes as well as occluded planes are assigned to the most probable split object.

Furthermore, we developed probabilistic measures for observed detection success, predicted detection success and the completeness of learned models where learning is incremental and online. This allows the robot to decide when to add a new keyframe to its view-based object model, and where to look next in order to complete the model, predicting the probability of successful object detection given the model trained so far as well as knowing when to stop learning.

We demonstrate that the proposed planar patches build basic *meaningful features*, where the robot can start to explore the scene. We further show that through interaction with planar parts the approximate structure of objects can be reconstructed and that the proposed measure for completeness explains what has been seen and where to continue exploration.

Acknowledgements

I would like to sincerely thank all people who supported me while working on this thesis. My foremost thanks go to my supervisor Markus Vincze. I am grateful for the freedom he provided me and for the continuing support. I would also like to thank my external reviewer Bastian Leibe from the RWTH Aachen for his time and for his encouragements. My gratitude additionally goes to Antonis Argyros for the fruitful discussions and ideas during my time at FORTH.

I am grateful to my office mates of the V4R group who continuously cared for an enjoyable working environment. Thanks for providing help in research related questions and for the various discussions during coffee, lunch, and tabletop soccer breaks. I am also indebted to the general public for funding my work via various EU and nationally funded projects.

Last but not least I would like to thank my brothers Peter and Andreas and my parents Maria and Hans for their unconditional support.

Contents

1 Introduction **1**
- 1.1 Cognitive Vision, Affordances and Object Modelling 4
- 1.2 Requirements 5
- 1.3 Putting it Together based on Planar Patches 6
- 1.4 Contributions and Outline of the Thesis 10

2 Related Work **13**
- 2.1 Situated Vision 13
- 2.2 Active Vision 15
- 2.3 Active Object Modelling 17
- 2.4 Affordance Learning and Recognition 20

3 Plane Detection **25**
- 3.1 State of the Art 25
- 3.2 Iterative Plane Hypotheses Selection 27
  - 3.2.1 Hypotheses Representation and Model Selection 29
  - 3.2.2 Hypotheses Generation and Efficiency 32
- 3.3 Tracking of Clusters 36
- 3.4 Post-Processing of the Planes 36
  - 3.4.1 Belief Propagation for Feature/Plane Assignment 36
  - 3.4.2 Splitting of Clusters 37
- 3.5 Tests and Evaluation 38
  - 3.5.1 Parameter Optimization 39
  - 3.5.2 Comparison of a Greedy and the Exact Brute Force Solution of the QBP 40
  - 3.5.3 Plane Detection 41
- 3.6 Discussion 45

4 Coherent Planes in Image Sequences **47**
- 4.1 State of the Art 48
- 4.2 Approach 49
  - 4.2.1 Object Representation 50
  - 4.2.2 Object Detection 51

4.3 Reasoning based on an Evaluation Function for the Object Hypotheses Graph . . . 52
4.4 Tightly Coupling Segmentation, Detection and Tracking in 2D using Model Selection . . . 54
4.4.1 Motion Segmentation . . . 54
4.4.2 Confidence Values for the Model Selection Framework . . . 56
4.4.3 Maintenance of the Object Models . . . 57
4.4.4 Direct Occlusion Detection . . . 58
4.4.5 Model Selection for Reasoning . . . 58
4.5 Tests and Evaluation . . . 60
4.5.1 Comparison with/without the Reasoning Component . . . 60
4.5.2 Tests with the Model Selection Framework for Reasoning . . . 62
4.6 Discussion . . . 65

5 From Planes to Objects **67**
5.1 State of the Art . . . 68
5.2 Basic Structure from Motion (SfM) . . . 69
5.2.1 Camera Geometry . . . 70
5.2.2 Reconstruction from Two Calibrated Views . . . 71
5.2.3 Feature Tracking and Pose Estimation . . . 73
5.2.4 Bundle Adjustment . . . 74
5.3 3D Object Modelling Based on Planes . . . 74
5.4 Homographies for 3D Reconstruction . . . 75
5.5 Pose Tracking . . . 76
5.6 Incremental Bundle Adjustment and Absolute Scale . . . 78
5.7 Merging and Splitting in a Probabilistic Formulation . . . 79
5.7.1 Merging of Planes with Consistent Motion . . . 79
5.7.2 Separating Planes in case of Different Motions . . . 80
5.8 Tests and Evaluation . . . 81
5.8.1 Evaluation of Plane Assignment using Color and Interest Point Proximity . . . 81
5.8.2 Tests of the Integrated System . . . 83
5.9 Discussion . . . 85

6 Model Completeness and Object Recognition **87**
6.1 State of the Art . . . 89
6.2 Online Learning of Objects . . . 90
6.3 Codebook Structure . . . 92
6.3.1 Vocabulary Tree . . . 93
6.3.2 Codebook Extension with Mean Shift Update . . . 94
6.4 Object Recognition . . . 95
6.4.1 Voting for Object Hypotheses . . . 95

6.4.2 Locally Optimized RANSAC for Object Hypotheses Verification and 3D Pose Registration 96
6.5 Self-Evaluation and Prediction 97
6.5.1 Probability of Detection 98
6.5.2 Representing Completeness 99
6.5.3 Taking Action 102
6.6 Tests and Evaluation 103
6.7 Discussion 106

7 Summary and Discussion **107**
7.1 Discussion 109

Bibliography **113**

List of Figures

1.1 Human tutor driven learning by showing, where a person shows his favorite cookies to the robot. 4

1.2 System overview 7

1.3 Example scenario we used to test our system, where a camera moves around objects and pushes them. The image shows a stereo setup, from which we use only a single camera. 8

1.4 Vision Approach 9

1.5 Object Model including the approximate structure of two objects (center), representations for object (-instance) recognition (green arrows) and representations for the completeness of the object model (blue arrows) 10

2.1 *Variety of arches (Schlemmer et al. [SPV09])* – what is the common constituting substrate? 14

2.2 *Object Semantic Hierarchy (OSH, Xu et al. [XK10]).* The agent repeatedly identifies new invariants with the goal to reduce the noise and thus builds models for foreground objects and for the background within different abstraction levels. 19

3.1 Test image with ground truth overlay 33

3.2 Connected components filter for early pruning of plane hypotheses. A 2D Delaunay triangulation is used to connect interest points (white edges). Three points are sampled (red) and an affine homography is computed. The graph is traced and points supporting the affine mapping are clustered (green). A Hypothesis is accepted if all initial sampled points are connected within the cluster (right image). 34

3.3 Sampling strategies and pre-filtering of hypotheses 35

3.4 A plane (top surface of packaging) accidentally picks up interest points in the background. Therefore we split the interest point clusters (colored dots) using the mean and standard deviation of edges of a Delaunay graph (white edges). 38

3.5 Parameter optimization 40

3.6 Comparison of plane detection methods. Left graph shows the plane detection result for images with no background texture. The test images of the right graph have a highly textured background. 41
3.7 Results for all our images (left) and for the Oxford houses data set (right). 42
3.8 Results for the packaging data set 1 using *ItMoS*, the proposed Algorithm 2. 44
3.9 Results for the packaging data set 2 using *ItMoS*, the proposed Algorithm 2. 44
3.10 Examples of the Oxford Visual Geometry data set using *ItMoS*. . . 44
3.11 Indoor scenes using *ItMoS*. 45

4.1 System overview, including the hypotheses graph (left), the structure of the system and the communication between components (right). 50
4.2 Figure 4.2(a) shows detected interest points and in Figure 4.2(b) the star-shaped object representation is sketched. 51
4.3 System overview, including the hypothesis graph (left), the structure of the system and the communication between the different components (right) including motion segmentation for automatic initialization. 55
4.4 Grouped motion vectors of interest points are shown with identical colors; different colors mean different clusters. Each subfigure depicts the result of different processing steps. Figure 4.4(a) shows the result of grouping according to similar length and direction of the motion vectors using a 2-dimensional histogram. Figure 4.4(b) pictures the result after examination of the neighboring motion vectors using a Delaunay triangulation and after affine outlier detection. In Figure 4.4(c) the result of Figure 4.4(b) is used to initialize a merging algorithm which combines clusters depending on their affine motion. 56
4.5 Probability map for one specific object computed using the predicted location and the result of the occlusion analysis. 57
4.6 Occlusion event including correct depth ordering and an occluded object linked to the occluder 59
4.7 Two possible interpretations of an observed scene are shown. Figure 4.7(a) shows the best interpretation of the frame 290, which is the correct one. The confidence value of this hypothesis path is 0.9669, compared to the other interpretation in Figure 4.7(b), which is only 0.8913. The system explained the occlusion event correctly. . 60
4.8 The first box sequence showing the various occlusion events (Frames: 420, 495, 545, 570, 595, and 845). The relative depth between the objects is correct in every frame. 61

4.9 Selected frames of the video named *Sorting the Shopping Basket*, indicating the complex interactions. Bounding boxes of learned objects are shown with different colors and the corresponding IDs and confidence values of all currently available models are depicted in the upper left corner of each image. If an object is lost, the last position is depicted with a colored dot surrounded by a grey circle and the ID of the occluded object is displayed under the occluder ID within brackets. 63

4.10 Part of a 900 frames long video which indicates the learning of an object model including the history of the object. The model of object 13 is learned while rotating to completely different views. Then it is moved behind object 3 and 9. During full occlusion, the object is rotated and appears again with a view learned at the beginning. . . 64

4.11 Three images of a 2590 frames long video are depicted showing two possible errors. 64

5.1 A typical Structure from Motion (SfM) pipeline. 70

5.2 Epipolar geometry. 72

5.3 Four solutions of essential matrix decomposition. 73

5.4 The upper row shows three keyframes of sequence 1 (897 frames) with detected planes in green which are merged because of common 3D motion. The brightness of the interest points indicates the assignment to different planes. After the gripper (two black dots on the left image border) pushes the plane 43, the keyframe-graph is traced back and the object model (44) and the background object (43) are created (lower image row). Changing plane ids of the top surface of the hexagonal object indicate that planes represented by homographies are substituted with better explanations. 77

5.5 Initialization to evaluate the plane assignment using colour and point proximity. 81

5.6 Comparison of plane assignment using color (Figure 5.6(a)), interest point proximity (Figure 5.6(b)), the combination of color and proximity (Figure 5.6(c)) and the according ground truth image (Figure 5.6(d)) for sequence 1 frame 280. 82

5.7 Comparison of plane assignment sequence 1 frame 480. 82

5.8 Comparison of plane assignment sequence 1 frame 515. 82

5.9 Sequence 2 (715 frames). 84

5.10 Sequence 3 (870 frames). 84

5.11 Sequence 4 (543 frames). 84

5.12 Sequence 5 (811 frames). 84

5.13 Example image and reconstruction of a small, more complex sequence which shows the limits of our system. Planes of the three dominant objects at the front are reconstructed, while the object in the center of the image and the objects in the background are not detected because of low texture and too few features. 85

6.1 Learning an object model: The object in the scene (left) and bundles of features with their view vectors (in blue) after acquiring some views of the object. View sphere (right) with brighter shades of grey indicating that the object has been learned from the respective direction. 88
6.2 Learning of interest point models. The top image of the left column shows interest points detected within the region of interest (center) which is computed from the point cloud. The bottom image shows the point cloud where the table plane is already removed. Note that, although the images show a clean table plane because of the rgb-depth sensor, this approach handles both: highly cluttered scenes and clean untextured tables. The center column indicates the alignment of interest points and 3D structure, where the right part shows 3D locations of the interest points projected to the object. The right column of the image depicts a possible camera trajectory, where keyframes are added and hence, a more complete object model is created. 91
6.3 Interest points (red circles) vote for object centers (coloured dots) which results in the correct pose indicated with the green overlay (left image). 96
6.4 Typical scenario to test the object recognizer. The colored overlays are computed from the convex hull of the 3D points of the recognized object views, which are projected into the image. 98
6.5 Rendered virtual views of an object for evaluating detection probability, with a true positive (left) and a true negative (right). 99
6.6 Histograms for true positives (blue) and true negatives (red) 100
6.7 Estimated Gaussian PDFs of confidence for true positives (blue) and true negatives (green) 100
6.8 *Observed detection success*: Posterior probability of detected object for given confidence 101
6.9 *Predicted detection success*: Posterior probability of detected object for given rotation, for a single learned view. 101
6.10 Comparison of *observed detection success* learned from virtual training data including various levels of noise (thin lines) and learned from real ground truth data (thick line). 104
6.11 Comparison of *observed detection success* for different objects. . . . 104

6.12 Comparison of *predicted detection success* learned from virtual training data including various levels of noise (thin lines) and learned from real ground truth data (thick line). 105
6.13 Accuracy of the pose of the recognized objects for real data. The evaluation is done in camera coordinates with the z-axis pointing towards the object. Hence, *error x* and *error y* represent the deviation from ground truth within the image plane and *error z* is the depth error. 105

List of Tables

3.1 Comparison of feature assignment methods. 37
3.2 Comparison of a greedy solution and the exact brute force computed solution of the QBP. 40
3.3 Results for the packaging data with a weakly textured background. 42
3.4 Results for the packaging data with a highly textured background. . 42
3.5 Results for the packaging data set. 42
3.6 Results for the Oxford houses data set. 43

4.1 Comparison of using a static model versus a dynamic model. 62

5.1 Comparison of plane assignment for sequence 1 82
5.2 Comparison of plane assignment for sequence 2 83
5.3 Comparison of plane assignment for sequence 3 83
5.4 Comparison of plane assignment for all sequences 83

Chapter 1

Introduction

Since people started using stones as hammers and learned to forge knives from metal, their intention is to disburden human lives. A major turning point in human history has been the industrial revolution which started at the end of the 18^{th} century. At this time there began a transition from manual labor to machine based manufacturing and machines that perform simple tasks by their own. This led to current robots that can be defined as stationary or mobile machines designed for specific tasks. Robots are usually programmed by experts and repeatedly execute tasks in a controlled (mostly) industrial environment.

In recent years, research has focused on the development of *intelligent* robots, which are able to operate side by side and cooperate with humans. Hence, robots operating in home environments as well as robots designed for industry need to be aware of their environments and perceive their meaning in accordance with humans. The ability to reason, perceive and act in humans environments leads to a *cognitive, i.e., intelligent* robot. In [oC10] *cognitive robot* is defined as being

> [...] concerned with integrating reasoning, perception, and action within a uniform theoretical and implementation framework (using methods drawn from logic, probability and decision theory, reinforcement learning, game theory, etc.). It is quite a young field of research. [...] Complex applications and the need for effective interaction with humans are increasing the demand for robots that are capable of deliberation and other high-level cognitive functions. Models from cognitive science and techniques from machine learning are being used to enable robots to extend their knowledge and skills. Combining results from mainstream robotics and computer vision with those from knowledge representation and reasoning, machine learning, and cognitive science has been central to research in cognitive robotics.

As can be seen, building a cognitive robot involves a lot of different disciplines. We want to attach special importance to the embodiment, the robot itself. It is convincing that a robot can only perform actions according to its embodiment. But

beside the action capabilities, all other sub-fields including the perception strongly depend on the embodiment and often could benefit if the system were treated from a global point of view. This is what we set out to do in this work: how to design visual perception given a cognitive robot system. To know what to require from visual perception, we first need to know what we expect the robot to do.

Basic skills for a cognitive robotic system are learning and interaction. Certainly, if robots move into daily life it must be possible to interact with them in a human-like manner. A robot must be able to expand its knowledge in a human tutor driven way or even learn on its own. Hence, the real challenge is to understand and build cognitive systems which are able to handle situations unforeseen by their designers. The aim is to create a theory – grounded and evaluated in robots – of how a cognitive system can model its own knowledge, use this uncertain knowledge, extend its abilities and knowledge, and extend its understanding of those abilities. This should lead to[1]

> ... a cognitive system that models not only the environment, but its own understanding of the environment and the way this understanding changes under action. It identifies gaps in its own understanding and then plans how to fill those gaps so as to deal with novelty and uncertainty in task execution, gather information necessary to complete its tasks, and to extend its abilities and knowledge so as to perform future tasks more efficiently.

Therefore, the system must be capable of self-extension, that is it must be able to learn, represent what it does not know, reason about what it can learn and how to act so as to learn it, to execute those actions and then learn from the resulting experience. We argue, that for a cognitive robot those abilities, namely *explore, explain and extend* are not only manifested in a high level reasoning component – the brain of a robot – but need to be implemented at each level of the system. At the lowest level, finally leading to an intelligent perception system tightly coupled with actuators.

This thesis focuses on a concept for visual perception that enables a cognitive robot to interact in human environments. The robot should act as a companion for humans and it should be able to perform *fetch and carry* tasks for everyday household objects.

Let's start with a specific example, which includes human tutor driven learning by showing (see Figure 1.1) as well as learning by experimentation by the robot on its own. Imagine a person, who wants to teach *my cup* to the robot:

> When Mrs. T got her new robot, she guided it through the apartment and since James already knows most of the furniture, it also knows the table in the kitchen. Mrs. T guides the robot James to the kitchen

[1]From the description of work for CogX, a project of the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181

cupboard, takes her cup and puts it on the kitchen table. Then she says:

- Mrs. T: James, that is my cup!

James looks to the table and sees Mrs. T pointing to the yellow cup with a picture of Eiffel Tower on it. Some time before, Mrs. T made tea and left a yellow tea box on the table. Because of the known geometry of the table, James immediately identifies that there is a second object which is also yellow. James analyzes these two objects and recognizes that these objects have a common modality, the color. So James tries to generalize the cup of Mrs. T in order to learn the object category *cup*, and asks Mrs. T:

- James: Mrs. T, there is a second object on the table, which is also yellow. Is it *a cup* too?
- Mrs. T: No, it is a yellow tea box.

Then she takes several cups out of the cupboard, puts them on the table and says:

- These are cups!

The table is quite small, so the cups are located close together. To be sure that there are several cups on the table James pushes one by one and separates them. Now, James is able to identify the common modality shape for the object class cup and it knows that the yellow cup with a picture of Eiffel Tower is Mrs. T's cup.

This scenario points out the tasks given to vision and perception as part of a robot system. Important questions emerging from the scenario are:

- What does my cup look like?
- How do cups look in general?
- Where is my cup typically found?
- Where are cups typically found within a house?
- How to grasp my cup?
- How to grasp cups in general?
- What are cups used for?

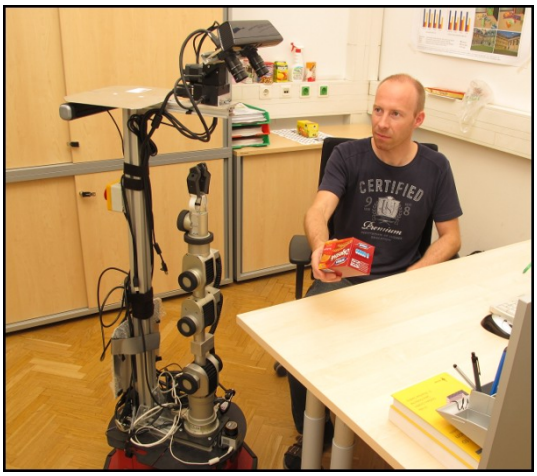


Figure 1.1: Human tutor driven learning by showing, where a person shows his favorite cookies to the robot.

Of course detecting the object instance *my cup* including 3D pose alignment is solved and can be done with high reliability [RBSF09]. It is more challenging to detect *a cup*, but recently developed approaches to detect generic object classes are promising (e.g. [LLS08, RBTH10]). In our opinion, the real challenge is to find a way to empower the robot with capabilities to gather information in a task driven way to enable it to learn by experimentation and accordingly provide meaningful features, explain them and extend the models.

1.1 Cognitive Vision, Affordances and Object Modelling

In essence, we try to provide vision for a cognitive robot. In the last few years, *cognitive vision* has become a popular buzzword within the computer vision community. Often the term *cognitive vision* is used to indicate a cognitively inspired method. We use the term cognitive vision for a vision system designed to bridge the gap between the robot and its environment (Vernon [Ver06]). In this sense meaningful features are parts of objects which describe what objects afford to the robot. Backed by the cognitive robot and the scenario outlined in the last section, our cognitive vision system needs to be an active vision system which links object models with affordances. Hence, vision has to be embedded into the cognition of the robot and thus it has to be adaptable to the current situation of the robot and its environment. Within this context, cognitive vision is equivalent to situated vision (Pylyshyn [Pyl01]) and tries to give a comprehensive approach to vision for a cognitive robot.

One concept that helps to link perception to action is the concept of affordances. The American psychologist J.J. Gibson states that the world is perceived not only

in terms of object shapes and spatial relations but also in terms of actions, which an agent is able to execute with an object. In the context of ecological perception, visual perception would enable agents to experience the opportunities for action in a direct way. In [Gib86] Gibson defines affordances of the environment:

> ...are what it offers the animal, what it provides or furnishes, either for good or ill something that refers both to the environment and the animal in a way that no existing term does. It implies the complementarities of the animal and the environment.

Examples for affordances would be: buttons for pushing, knobs for turning, handles for pulling, but also cups for grasping or to fill them with coffee. According to Gibson's theory, perception of the environment inevitably leads to some course of action.

Affordance based visual object representations are function based representations. Stark and Bowyer [SB94] and Rivlin et al. [RDR95] use a set of primitives, such as relative orientation, stability or proximity and face and vertex information to define specific functional properties. According to the scenario described in the last section, the simplest action is touching an object. Furthermore, the robot should perform simple object manipulations that aim at grasping the object and putting it on top of another object. Affordances necessary for this are for example 'pushable', 'graspable' and 'supportable'.

1.2 Requirements

Considering the cognitive robot and our scenario, the following requirements to link affordances to an object model can be positioned:

Meaningful parts: bridge the gap between what is seen in images and what objects afford to an embodiment. We propose piecewise *planar surface patches* detected in image sequences as meaningful parts.

Clusters of parts: are created from basic meaningful parts and bridge the gap to complex object affordances (e.g. grasping). For the description of clusters, we propose to use histograms of the relations of angle and the scale between reconstructed parts.

Description of the completeness of models and of knowledge gaps: Before the robot can formalize what to explore, it must describe what it knows. We propose a probabilistic completeness model which can be visualized with a spherical histogram showing what has been seen up to now.

Recognition of object instances: is one of the basic abilities of a cognitive robot which has to operate side by side and cooperate with humans. For this we combine state-of-the-art interest points and color histograms.

Recognition of object classes and categories: One of the most important abilities of a cognitive robot is the ability to generalize from instances. Depending on the object and its affordances, 2D image features, such as color and texture, as well as, 3D shape features are necessary to describe object categories. We propose to combine colour, texture and the description of clusters of parts to recognize object classes and categories.

Interaction such as pushing and grasping: Beside the visual appearance in static images, motion in image sequences is the strongest cue to cluster parts which belong together. The proposed meaningful parts can be directly used to compute approaching vectors for pushing and grasping.

Action selection and planning: To close the circle from perception to the exploring robot, action selection and planning need to be supported. The prediction *what to do next* can directly be inferred from the proposed description of completeness. Planning the action itself is supported by the reconstructed clusters of meaningful parts.

1.3 Putting it Together based on Planar Patches

State-of-the-art object recognition approaches learn appearances of object parts directly in images. Normally, for this interest point detectors are used to gather salient regions, which are then encoded with image descriptors. For learning "meaningful" parts, similar interest points are clustered and significant ones are assigned to the corresponding object class (Leibe et al. [LLS08]). These approaches implicitly encode the projective object shape. To some extent they are able to cope with the large variety within object classes. Current affordance learning approaches use interest points (Fritz et al. [FPK+06]) or approximate objects with basic features such as color, circleness or squareness (Montesano et al. [MLBSV08]). What all of these approaches have in common is that they use interest points or basic shape features and color to create a vaguely construed link from affordances to "meaningful" parts.

Our hypothesis is that clusters of planar patches bridge the gap between images and objects affordances. Hence, in contrast to state-of-the-art approaches, which implicitly encode affordances with image features, we propose to detect planar patches in pairs of images, cluster and reconstruct them depending on common motion in image sequences and use these features to explicitly encode affordances.

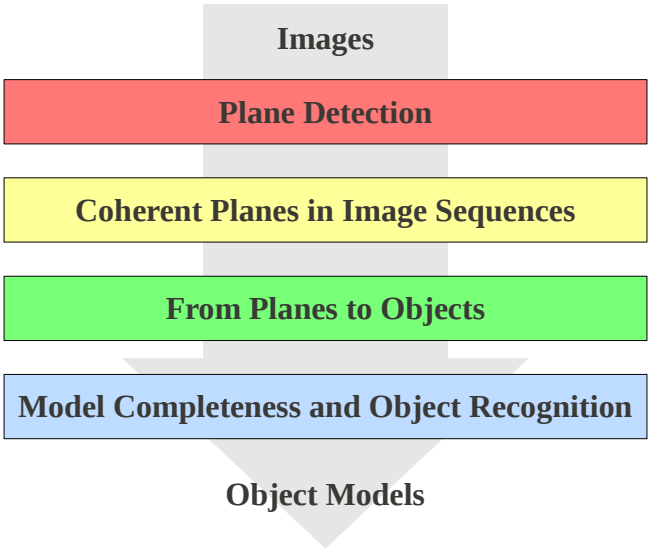


Figure 1.2: System overview

The diagram shown in Figure 1.2 indicates the tasks necessary to form meaningful parts from images and accordingly create the object model and their completeness description. The first step is to detect planes in pairs of images (red). For this we rely on interest points to compute plane hypotheses. To provide reliable planar patches, coherent planes are detected and tracked in image sequences (yellow). Subsequently, planar patches are reconstructed and clusters with common motion build initial object hypotheses. Contact points, computed from parts, are used to push them and if parts start moving separately they are split to separate items (green). Finally, features for recognition are detected and the probabilistic completeness model is computed (blue).

A specific experiment is shown in Figure 1.3. Imagine, a robot which moves around and at one point sees an unknown structure popping out from the floor or from a table. It moves around the structure and builds an object hypothesis of it, followed by simple interaction to verify the hypothesis. The individual steps for this are shown in Figure 1.4. The system is based on interest points which are tracked in an image sequence. To detect planar patches, we developed an incremental model selection scheme, where planar patches once detected are tracked and serve as prior in subsequent images (red). In each iteration, tracked planar patches and new planes computed from randomly selected interest points are evaluated. If planar patches accumulate enough disparity, they are reconstructed and build the initial object hypotheses (green). Incremental bundle adjustment in combination with the estimated motion (encoder/odometry) by the robot ensures an optimal reconstruction in an absolute coordinate system. The incremental approach adds new planar patches if new viewpoints are visited (Figure 1.4 second row). If patches start moving separately, a split event is triggered and the accumulated information,

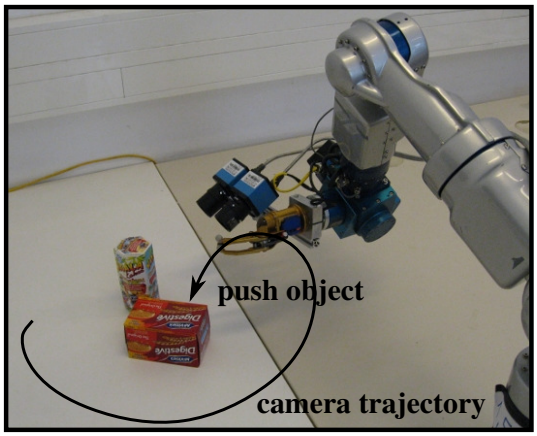


Figure 1.3: Example scenario we used to test our system, where a camera moves around objects and pushes them. The image shows a stereo setup, from which we use only a single camera.

which is stored in a keyframe based graph structure, is evaluated. Therefore the graph is traced back and depending on color and structure information, the patches are assigned to the most likely split object hypothesis. Accompanying the modelling process the object description parameters stated in the last section are stored within the graph structure. Hence, at each point in time, the system is able to provide the necessary information to plan trajectories to visit new view points in order to get a more complete object model or to interact with the object.

As already indicated, the final object model is based on an approximated 3D structure using planar surface patches. The motivation behind using planar patches as basic parts is, on the one hand, that planes can directly be detected in image pairs and, on the other hand, that the relation of planar patches in the 3D Euclidean space can be efficiently used for interaction. State-of-the-art interest points in combination with local descriptors such as SIFT [Low04] or SURF [BETG08] are used for reliable recognition of object instances and for pose alignment. To account for object classes, each individual planar patch holds a normalized color histogram and the relative location of pairs of patches is described by a histogram of angle and scale.

So far, we end up with a representation of physical properties derived from images. Still, there is one issue that remains open, namely, how to support planning with the goal to improve the object model? Necessarily, to provide information about these knowledge gaps, we need a description of the completeness of the object model. Therefore, we developed a probabilistic representation, which indicates the likelihood to successfully recognize objects given a specific viewpoint. The main idea is to carry out tests about stability while learning the object and to maintain a spherical viewpoint histogram. Hence, at each point in time the approach is able to predict the success rate of recognition from a specified view point, as well, as to

Figure 1.4: Vision Approach

evaluate the recognition result.

To recapitulate, the object is built-on the the following cues (Figure 1.5):

- 3D shape description for interaction using piecewise planar surfaces (center),
- Color histogram for object(-instance) recognition (green),
- Relative angle and scale (RAS) description of planar patches for object class recognition (green),
- Interest points including 2D image location and 3D location for object instance recognition and pose alignment (green), and

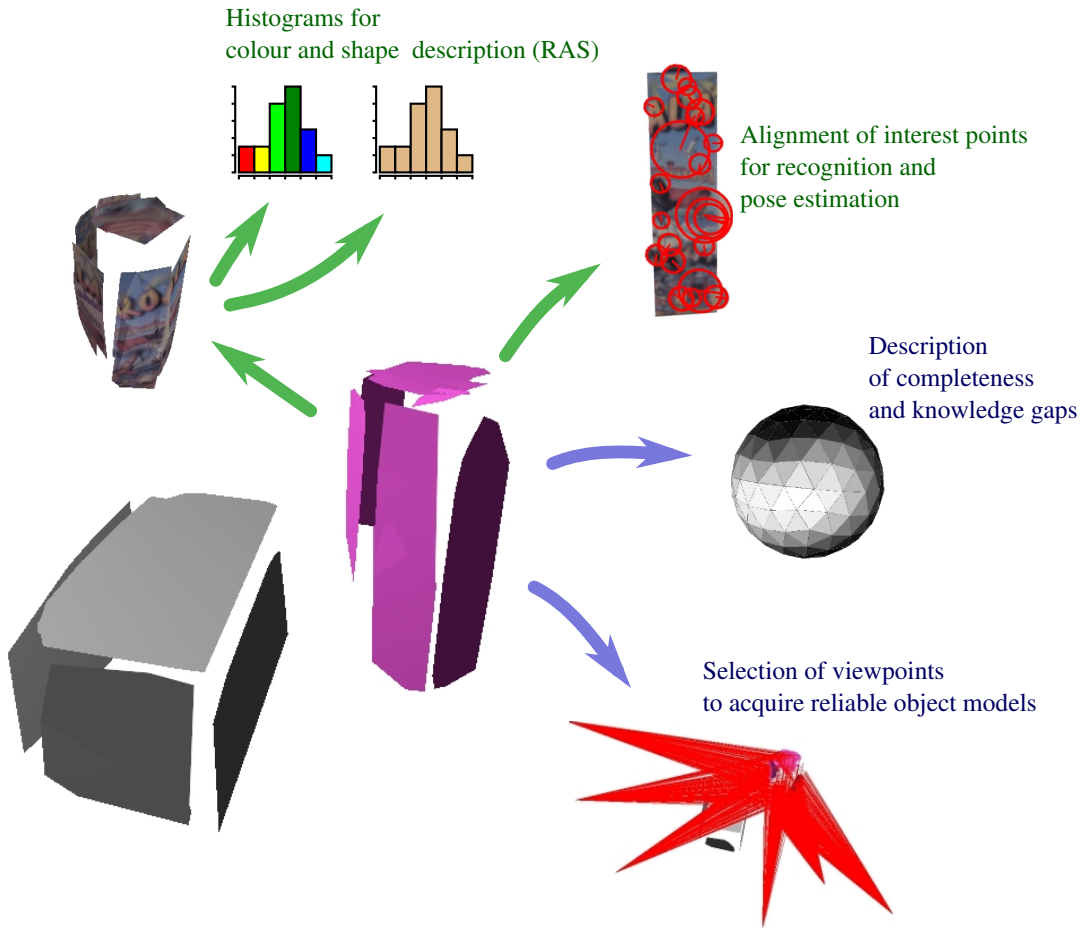


Figure 1.5: Object Model including the approximate structure of two objects (center), representations for object (-instance) recognition (green arrows) and representations for the completeness of the object model (blue arrows)

- Spherical viewpoint histogram to represent the completeness and knowledge gaps (blue).

1.4 Contributions and Outline of the Thesis

This thesis presents a system to interactively detect, model and recognize objects for a cognitive robot. The first contribution is to provide a concept for an object model to enable a robot to interact with its environment. We propose an object model based on planar patches to manifest affordances for the robot. The following paragraphs present an outline of this thesis and describe the contributions of each chapter. The thesis is structured as follows:

Chapter 2 – Related Work gives an overview of related work in situated vision and active vision, on which our work is concentrated. Next, we review the state-of-the-art in related vision areas which we tackle in this thesis, namely active object modelling and learning of object affordance and affordance recognition.

Chapter 3 – Plane Detection introduces the detection of planar surface patches. Planar patches are the main building block for the subsequent 3D reconstruction and for the later affordance recognition. We developed a new method to detect multiple planes in image pairs. The theoretical framework for this is grounded in model selection and Minimal Description Length (MDL). We embed model selection in an iterative scheme, where existing planar patches have to compete with newly generated hypotheses. Planes are represented with the 2D projective transformation (homography), which is computed from four sampled interest point pairs. This method allows to create plane hypotheses mainly in unexplained regions and in this way, it limits the search space which leads to a faster explanation of the entire image. Furthermore, the iterative model selection scheme invites to incorporate knowledge about tracked planar patches from the last frame and information of new detected planes in the current image. The framework allows to avoid a confidence value, which otherwise would be necessary for pruning wrong tracking results. Detection of planes is also published in [PZLV10].

Chapter 4 – Coherent Planes in Image Sequences introduces a consistency check and provides a method for reasoning in image sequences. While model selection, described in the previous chapters is formalized to select the best planes of the current frame, here we investigate how to keep track of multiple hypotheses. Therefore, a graph based representation of multiple hypotheses, which describe the content of a particular frame, is proposed and an evaluation criterion is used to calculate the most likely hypothesis path that best explains the image sequence. The ideas of occlusion reasoning and reasoning in image sequences are published in [PAAV09, PZV09, APVA09].

Chapter 5 – From Planes to Objects investigates the reconstruction and the merging of planar patches to individual objects. Tracked planar patches are directly used to reconstruct multiple objects. For this purpose, consistent with the MDL-formulation of the previous chapter, a pseudo-likelihood is developed which combines motion, color and the spatial arrangement of planar patches. The approach presented here is related to the problem of Multi-body Structure-and-Motion (MSaM). Instead of directly reconstructing object models from interest points, we first cluster the latter to planes using 2D projective transformation and then reconstruct planar patches with consistent motion. Furthermore, a new strategy is presented, which enables an agent to interactively learn object models. This work is also published in [PZV11b].

Chapter 6 – Model Completeness and Object Recognition investigates the recognition of objects. To represent the completeness of an object model, confidence values are learned and a probabilistic view representation is proposed. This representation is used to predict the success rate of recognition from a specified view point, as well, as to describe the overall model completeness and to evaluate the reliability of object recognition. Object instance recognition, itself is based on a hierarchical approach with different levels of abstraction, starting with a codebook representation and a fast indexing of object views and finally leading to a 3D pose alignment of the object. The generative object model and the provided description of completeness allow a robot to decide whether it should continue learning or stop and make use of the models learned so far. The proposed recognizer and the probabilistic model for object completeness are also published in [ZPMV11].

Chapter 7 – Summary and Discussion concludes with a discussion of the relevance in cognitive robotics. Subsequently, further directions and the perspective of this thesis are given.

Chapter 2

Related Work

Our system combines different methods and thus contributes to different areas in computer vision. It is based on the paradigm of situated robot vision and the overall outcome can be attributed to the field of active vision. Hence, we start with a review of related work in situated vision, active vision and active object modelling. This leads us to the aspects learning of affordances and affordance recognition. At the beginning of each chapter, we review the state-of-the-art for specific vision algorithms used in this work.

2.1 Situated Vision

We use the term *situated vision* in the sense of [Sch09] and [SPV09] who refer to the dual goal of situating vision inside a broader cognitive framework and to emphasize the situatedness of the agent – and hence of its vision – in its environment. In this sense, it is related to the use of this notion by [Pyl01], which claims that a theory of situated vision needs to know two distinct routes – a pre-conceptual, unmediated connection between visual elements and elements in the world, as well as a conceptual, constructive representation of the stimuli.

In the last years, there has been an inflationary use of the notion "cognitive vision" to emphasize the need to include "cognition" into vision research. However, very often it either resulted in *explaining* some algorithm as if it were cognitive or in sticking cognitive reasoning on top of vision. Schlemmer et al. [SPV09] claim that true "cognitive vision" is in its best sense pleonastic: Vision needs to be seen as *part of* cognition. They say that *vision* in its best sense – as *situated vision* – needs to be recognized as being multi-dimensional, not just two- or three-dimensional. This means that dimensions, such as prediction, priming, or intentionality must be accounted for. Schlemmer et al. claim that this would indeed allow to solve tasks that are not only hard at the moment but in fact (due to a wrong approach) in principle not solvable. Consider Figure 2.1: When confronted with the task of recognizing arches, a classical computer vision learning algorithm would rely on a huge training set of arch-images in order to re-detect one (or a *very similar* instance)

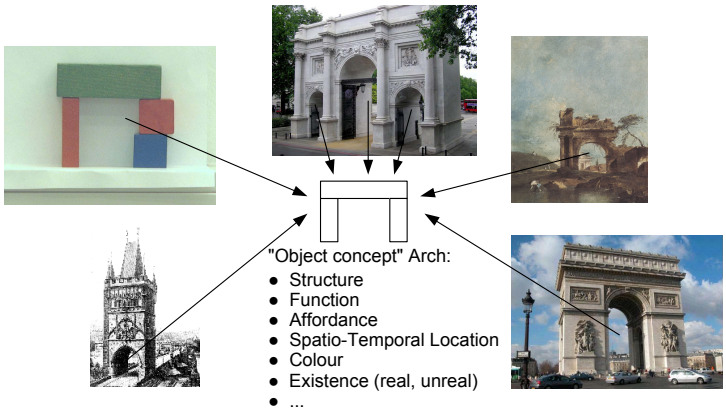


Figure 2.1: *Variety of arches (Schlemmer et al. [SPV09])* – what is the common constituting substrate?

of those when confronted with it in a test situation. Humans, however, even if they have never seen one of those arches before, would immediately recognize that all images of Figure 2.1 indeed show instances of arches. Schlemmer et al. argue that one crucial "cognitive function" that is at work here is *abstraction*, i.e., seeing the *concept* behind the *instance*.

In [Sch09] Schlemmer defines the following stances of "situated vision" for a cognitive robot companion:

- Incorporates various non-visual cues,
- Relies on a powerful ontology of high-level scene knowledge, observations, empirical generalizations, etc,
- Uses semantic information about what is seen by linking observations to previous knowledge,
- Is not only *feeding* other cognitive subtasks with "adequate" information, but is *itself fed* by those routines,
- Includes for the special case of a robot companion the possibility to see things "with another agent's eyes" (due to the shared working environment), and
- Is a problem definition that demands an interdisciplinary approach.

Computer vision, as outlined above, never quite reaches a "semantic level". This is not a big surprise, as it usually proceeds in a bottom-up fashion. "Meaning" can *only* get into the system by connecting observations with previous knowledge (the sources of which might be very diverse – reaching from innate stuff to empirical generalizations). A simple illustration: Detecting a cylinder is a typical computer vision task. Knowing that this cylinder is the favorite coffee mug of

someone goes far beyond that. However, a pre-given and fixed 1:1 matching of observation to higher-level concepts would not allow for the context-dependence claimed before, and robot companions are inherently confronted with dynamically changing environments. Hence, according to Schlemmer [Sch09], the approach is even more abstract, namely *functional*. He proposes to search for necessary functionalities as deployment of what he calls *cognitive functions*. The functional layer allows us to connect various disciplines that – on a more concrete layer – might diverge too much (cf. Mark Solms' conception of the right discussion layer between neuro-psychoanalysts and engineers in [Sol08]).

For situated vision of a robot companion, Schlemmer [Sch09] proposes that *intentionality*, *prediction*, *abstraction*, *generalization*, and *symbol binding* seem especially important. The first one is the overall principle of task-guidance which can already be found in older philosophical works (such as [Pup13]), and which introduces the directedness of consciousness towards an object (be it "internal" as a kind of representation or "external" as object of focus of attention). The second function is more concretely concerned with anticipating what will be seen next and can be loosely associated with the unfolding of situation-understanding and the use of previous knowledge. Some approaches believe this is the most important capability of the human brain, e.g. [HB04]. Abstraction then is concerned with mapping actually observed data to higher-level conceptual knowledge. This is the really hard part of seeing the "concept behind the instance" (cf. Figure 2.1). Generalization is the other way round: The building up of a concept from instances seen. In [Sch09], Schlemmer proposes that this is *not* thought to be appearance- and brute-force-based but rather in the sense of extracting the "thing-in-itself" as philosophers call it: the constituting substrate of what makes an object an object. Symbol binding, finally, is the capability to link additional (of course, again situation-dependent) information to the observations made.

As can be seen, some of the mentioned cognitive functions are more vision-related (prediction, abstraction, and generalization) than others (intentionality and symbol binding) which can rather be seen as superordinate principles that govern the whole processing of the cognitive agent. Nevertheless, they need to be incorporated into a thorough vision analysis of what a robot companion needs.

2.2 Active Vision

Early attempts on Active Vision go back to [Baj88], [AWB88], [Bal91]. In [Baj88], Bajcsy states that perception is not passive, but active. Bajcsy strengthens this statement with a comparison to humans, where the pupils adjust to the level of illumination, the eyes bring the world into sharp focus, the eyes converge and diverge and the human head moves to get a better view. Compared to an "active sensor", which generally refers to a sensor which transmits to the environment, such as a time-of-flight sensor, Bajcsy uses the term 'Active Vision' for a passive sensor

operated in an active fashion. Here, the sensor's state parameters are changed according to the sensing strategies. Thus, according to Bajcsy, Active Vision is an application of intelligent control theory which includes reasoning, decision making and control. This implies three facts: First of all, an Active Vision system needs the model of the physics of the sensors as well as the noise of the sensors. Then the model of the signal processing and the data reduction mechanisms are applied to the acquired sensor data. Together, these two models build the "local models". The system is modular and interactive, thus a further model is required which models the whole process, including feedback. Bajcsy refers to this model as "global models". Thirdly, it is necessary to explicitly specify the initial and the final stage/goal. Bajcsy's emphasis lies in the study of modelling and control strategies for perception.

This differs from Aloimonos et al. [AWB88] who introduce a general methodology, in which they hold the view that low-level vision problems should be addressed. They investigate several basic problems in vision under the assumption that the observer is active and provides a continuous sequence of images with a known viewing transformation. They argue that "problems that are ill-posed, nonlinear or unstable for a passive observer become well-posed, linear and stable for an active observer." They investigate four algorithms in computer vision, namely shape from shading, shape from contour, shape from texture and structure from motion. The proposed solutions address problems regarding stability and linearity. Furthermore, Aloimonos et al. propose unique solutions which require no assumptions.

Following these ideas, researchers have contributed in different areas. Krotkov [Kro88] studies automatic focusing of a servo-controlled video camera. He manually selects a target and tries to find the position of the servo motor with the sharpest focus on the pre-specified object point and then computes the distance to the point. Krotkov proposes to measure the sharpness of focus based on maximizing the magnitude of the intensity gradient since this solution has the advantage of being unimodal, monotonic about the mode, and robust in the presence of noise. For this purpose, the Fibonacci search technique is used to optimally locate the mode of the criterion function. To determine the distance of the selected point, the thick-lens law is used. For objects within 3m this allows to compute the location with a precision of 2.5cm.

According to the Active Vision paradigms of Aloimonos, Hamker [Ham06] proposes feature-based attention as an active top-down inference process. This approach aims to select relevant information within the scene and the computation of an appropriate representation. In the sense of a visual selection device, this method is able to acquire the necessary information on demand by focusing on the relevant areas within the visual scene while taking different viewpoints of the same object. Therefore, an approach is developed in which feature-based attention acts on the object representation itself. The computed top-down expectations meet the bottom-up processed stimulus features in the ventral pathway. Then a competitive interaction mechanism filters out the information that is inconsistent with the

high-level goal description.

The work of Mishra [MA09] et al. follow up ideas of the original ideas of Active Vision by Aloimonos with respect to segmentation. The motivation is that the human visual system observes and understands a scene by fixation of points and thus humans are able to see small parts around those points (in the fovea) in high resolution. Hence, in this work the basic segmentation problem is defined as segmenting a region containing the fixation point. The proposed algorithm combines monocular cues (color/intensity/texture) with stereo or motion in a cue independent manner. The core cue integration relies on a log-polar transformation of the images around the fixation point and graph-cut to separate the edge-image into foreground and background. The cue integration itself is performed at edge level, where edges are weighted depending on depth cues or motion. Hence, this work incorporates the Active Vision paradigm with respect to focusing on fixation points and integration of image sequences.

We tackle the Active Vision paradigm with respect to several points. In contrast to Aloimonos et al. [AWB88], who focuses on linearizing vision problems with the help of continuous image sequences, we rely on state-of-the-art nonlinear optimization techniques, but in case an estimate of the camera pose is available, this information is involved at reconstruction level. We aim to model objects within a natural environment. With respect to segmentation of the object, our approach goes beyond the traditional Active Vision paradigm. Our integrated system actively interacts with objects to group features and form objects and in this way segmentation relies, beside geometry and color, on motion.

2.3 Active Object Modelling

In the spirit of Active Vision we are concerned with active object modelling. We now give a brief review of systems which model the environment and especially objects within the environment. Most related literature belongs to robotic navigation and assumes a laser range sensor. The motivation is to model and detect objects and thereby to get rid of the static world assumption. In [BLST02] Biswas et al. propose an object mapping in non-stationary environments where the objects change their location over time. The approach builds on the well-known occupancy grid map. Each map captures a snapshot of the environment at a specific point in time. Moved objects are detected using standard change detection algorithms. Then the approach learns models of the objects using an expectation maximization (EM) algorithm, where the E-step establishes correspondences between different object views at a different point in time and the M-step refines the object models represented by occupancy grids. The assumption is that the environment changes slowly and the objects can be assumed to be static for the time of mapping.

Modayil et al. [MK04] also use mapping of laser range data to an occupancy grid. The main emphasis of the approach of Modayil et al. is to learn an ontol-

ogy of objects to explain aspects of the sensor input from an unknown dynamic world. The occupancy grid representation for local space does not include the concept of an object. Hence, the assumption is that the environment is static and that the space can be divided into empty and occupied. Then the shape of the occupied space is detected. In [MK06], they improved the creation of the object shape model. They propose an algorithm that defines angular constraints between multiple sensor scans of an object. Due to these constraints the system is able to align the scans and to create maximally coherent object shape models. In [MK07] and [MK08], Modayil et al. extend the model and propose a system with multiple integrated representations, including spatio-temporal clusters of data, percepts which represent properties for the tracked objects, generalizations from past experience, and actions which change object percepts. Thus, the robot is able to use the observed shape of a tracked object to generate shape classes, which are used to generalize from past experiences. Hence, the robot is able to interact with an individual object using learned actions that modify percepts of the objects.

Modayil closes the loop from simple laser range sensor data to higher level objects and object interaction. Xu et al. [XK10] state that in order to understand and manipulate in the world, the agent must be able to learn high level concepts. Hence, they propose multiple representations with different ontologies within a joint representation which they call Object Semantic Hierarchy (OSH, see Figure 2.2). This representation separates the problems of object perception into intermediate stages. Each layer consists of knowledge about an object with the natural representation and thus with relatively simple transitions from less structured to more structured. From the bottom to the top, each layer creates new invariants on top of the previous layer. The approach starts from a constant background model, where the foreground object is treated as noise. Then the foreground objects are individuated from the background and the object model is created while they are tracked. The idea of the hierarchical model is, that lower layers can provide simpler properties that are more robust, while higher layers benefit from lower layers and get simpler too. If higher layer, fail processing at a lower level can still be successful and can accumulate information for a more sophisticated analysis.

While the above literature is concerned with modelling the object (shape) itself, Hart et al. [HGJ05] and Stoytchev [Sto05] propose representations for task knowledge and affordances. According to Mandler [Man04], task knowledge can be decomposed into declarative and procedural components. In [HGJ05], Hart explains how to learn procedural knowledge when the declarative structure is given. They propose a relational model to find the statistical dependencies between sensorimotor variables and task success. Mandler claims that relational models are useful because they provide a framework for learning from experience. Furthermore, if the structure of a task recommends an action, the model can be used to observe the sensorimotor variables and to determine which resources should be employed. Stoytchev [Sto05] proposes a behavior-grounded representation of tool affordances. The idea is to ground tool affordances in a behavioral repertoire of the

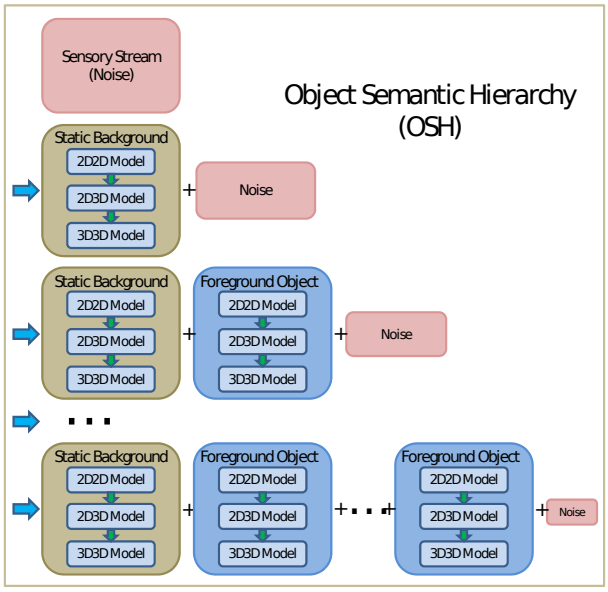


Figure 2.2: *Object Semantic Hierarchy (OSH, Xu et al. [XK10]).* The agent repeatedly identifies new invariants with the goal to reduce the noise and thus builds models for foreground objects and for the background within different abstraction levels.

robot. The representation is learned in a training phase, where the robot randomly chooses different behaviors, applies them to the tool, and observes their effects in the environment.

Metta et al. [MF03] develop an active strategy for a robot to acquire visual experience through simple experimental manipulation. The experiments are oriented towards determining what parts of the environment are physically coherent, that is, which parts will move together, and which are more or less independent. For testing their approach, an upper torso humanoid, called *Cog*, is used, which is not designed to enact trajectories with high fidelity. It is rather designed for interaction with a poorly characterized environment, where collisions are frequent and informative events. The system includes an attentional system consisting of a pre-attentive filter sensitive to motion, color, and binocular disparity. These filters are implemented on a space-variant imaging system, which mimics the distribution of photoreceptors in the human retina. A voting mechanism is used to decide what to attend and to track next. Several experiments are performed: Beside studies about direct effects of actions, where time-correlation of optical flow is used as signature to identify parts of the scene that are influenced by the robot's motion, indirect effects including a refinement of the segmentation and the development of mirror neurons are studied.

Our experiments are similar, but in contrast to Metta, who studies the causal chains of events, we focus on learning a 3D piecewise planar object model trig-

gered by motion events. With respect to the conceptual object modelling system, the work by Modayil et al. [MK08] and Xu et al. [XK10] is closely related to our approach. They rely on a laser ranger, which is only able to sample points in a plane with constant height to the floor. Accordingly, the representation of objects is simplified to a contour in 2D. Instead, we propose a system with a monocular camera and build a complete 3D model up to a certain generalization which is necessary for robot interaction. The work by Stoytchev [Sto05], who already proposes learning of affordances, directly leads us to the next section.

2.4 Affordance Learning and Recognition

The world is perceived not only in terms of object shapes and spatial relations but also in terms of actions, which an agent is able to execute with an object (Gibbson [Gib86]). For cognitive robotics, this becomes essential if we want to enable the robot to interact in an unknown environment. In the following, we provide an overview of approaches that demonstrate the learning of causal relationships between visual cues and predictable interactions.

Fritz et al. [FPK+06] show the importance of learning in perceptual cueing for anticipation of opportunities for interaction of robotic agents and provide a refined concept of affordance perception. They propose an interaction component to recognize relevant events in interaction via perceptual entities and a predictive aspect to predict interaction via perceptual entities. The outcome of the affordance cueing system, given a multimodal feature vector, is a probability distribution over affordance hypotheses. Among other cues, such as color, shape and 3D information, they investigate the benefit of using visual 2D patterns, especially Scale Invariant Feature Transform (SIFT [Low04]) features for their use in affordance cueing. The proof of concept is shown with experiments using a simulated environment. There, predictive 2D affordance cues to characterize affordance recognition processes have successfully been learned.

In [MMFF03], Metta et al. extend the experiments studying the causal chain of events, described in [MF03], with object affordances. They stated that affordances are not only a property of the mechanics of an object, but rather a combination of visual appearance, of the object's physical composition, and the ability of an actor. For the visual component of the affordance, they select the principal axis of the object directly measured from the segmentation. For the experiments, they select four simple objects (bottle, cube, car, ball) and train about 100 actions per object, where the motor vocabulary includes four possible directions to poke. A clustering schema is used to cluster the resulting behaviors. Using these four objects the robot made about 15 false behavior predictions out of 100.

The work of Sinapov et al. [SS07] is based on the framework for behavior-grounded representation of tools by Stoytchev [Sto08]. They extend the model with an approach that enables the robot to learn the effects of its action with a

tool and therefore to learn to detect features in its sensory stream which are useful to predict the effects. The final model consists of a compact representation of the action possibilities that a tool affords the robot. This representation is learned by exploring the space of actions with the tool and observing their effects. Hence, the predictive model is grounded in the robot's own perceptual and behavioral repertoire. For learning of the compact model a classical decision tree [Qui93] approach is used. Experiments have shown that the decision tree outperforms the k-NN algorithm and that a reference frame centered on the attractor object contains the most predictive information. An application towards detection of functional components is proposed by Sukhoy et al.[SS10]. They use active exploration and a multimodal correlation of the input data that is a visual feature vector together with auditory feedback, to train a visual model for detecting the functional components of a doorbell button. The system acts like an affordance detector as it is able to label patches with, e.g., "pushable there". Experiments have shown that for novel buttons, the system is still able to find a decent approximation of the functional components.

Affordances encode relationships between actions, objects and effects. Montesano et al. [MLBSV08] present a general model for learning object affordances using Bayesian networks integrated within a general developmental architecture for social robots. Furthermore, one of the motivations of Montesano's system is imitation, where affordances also play an important role for interaction with another agent. Like newborns have a series of reflexes and responses, Montesano considers that the robot starts with a predefined set of core motor actions which must be adjusted by self-experience. Experiments have been conducted in a simple playground environment to get along with the built-in segmentation and category formation capabilities of the system, which provides region based measurements such as convexity, eccentricity circleness and squareness. For learning, Montesano et al. propose a Bayesian Networks to encode the dependencies between actions, object features, and the effects of those actions. This representation permits to take into account the uncertainty of the real world, to encode some notion of causality and to provide a unified framework for learning and using affordances.

In [RSL10], Ridge et al. propose a self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In the real world it is important that a robot builds its own internal representation of object affordances. Ridge et al. refer to the input modality of features for features taken prior to arm-object interaction. For this purpose, they first fit a quadratic surface to segmented range data. Secondly, use the intensity image segmentation to compute 10 shape features. After an action has been initiated on an object in motion, the output modality is formed. Therefore the motion is calculated with the help of a color-based particle filter and 9 features are calculated. To estimate how the appearances of the objects change during motion the average difference of color and edge histograms is computed. For co-occurrence clustering, Ridge et al. develop an online classifier training algorithm based on Kohonen's learning vector

quantization (LVQ) which does not require labels during training. Thus, given a series of interactive episodes, the learning task is to find clusters in the output modality feature space which correspond to affordance classes. These clusters are then used to train a classifier for the input modality space which is used to predict the affordance classes.

The above approaches either start with a pre-defined set of action capabilities, which are improved by self-experience (e.g. [MLBSV08]), or the action behavior is simply given [RSL10]. Given a (multi-)modal input vector, the systems learn to predict effects according to an action and an object. Stoytchev [Sto08] propose a decision tree for learning to predict affordances. While all of the described approaches concentrate on learning affordances from 2D features in simplified environments. Instead, we propose to use relations of piecewise planar surface patches in 3D Euclidean space which are directly represented within our object model.

In summary, we reviewed different areas which are related to computer vision within the context of cognitive robotics, namely *situated vision*, *active vision*, *active object modelling* and *affordance learning and recognition*. In this work, the term situated vision is used in the sense of [Sch09] and [SPV09] who refer to the dual goal of situating vision inside a broader cognitive framework and to emphasize the situatedness of the agent – and hence of its vision – in its environment.

The proposed approach is about the perception of an active robot which explores its environments. Early work on active vision goes back to Bajcsy [Baj88], Aloimonos et al. [AWB88] and Ballard [Bal91]. Bajcsy states that perception is not passive, but active. All of them investigate different aspects related to computer vision. Whereas Aloimonos et al. investigate how low level vision benefits from image sequences captured from an active sensor, according to Bajcsy active vision is an application of intelligent control theory which includes reasoning, decision making and control.

Our experiments within the context of active object modelling are similar to the work of Metta et al. [MF03], but in contrast to Metta, who studies the causal chains of events, we focus on learning a 3D piecewise planar object model triggered by motion events. With respect to the conceptual object modelling system, the work by Modayil et al. [MK08] and Xu et al. [XK10] is closely related to our approach. They rely on a laser ranger, which is only able to sample points in a plane with constant height to the floor. Accordingly, the representation of objects is simplified to a contour in 2D. Instead, we propose a system with a monocular camera and build a complete 3D model up to a certain generalization which is necessary for robot interaction. The work by Stoytchev [Sto05], who already proposes learning of affordances, directly leads us to affordances and the interaction of a robot with the environment.

State-of-the-art approaches for learning affordances either start with a pre-defined set of action capabilities, which are improved by self-experience (e.g. [MLBSV08]), or the action behavior is simply given [RSL10]. Given a

(multi-)modal input vector, the systems learn to predict effects according to an action and an object. Stoytchev [Sto08] propose a decision tree for learning to predict affordances. While all of the related approaches concentrate on learning affordances from 2D features in simplified environment. In contrast, we propose to use relations of piecewise planar surface patches in 3D Euclidean space which are directly represented within our object model.

Chapter 3

Plane Detection

For proposing planes as basic features to learn dense object affordances and their relations, we need to develop a method to detect multiple planes in image sequences. In [PZLV10] we have shown that our method competes with state-of-the-art algorithms while it is more flexible to incorporate prior knowledge from tracking. In the following, we first review the state of the art 3.1 and then describe the approach including model selection based plane detection (Sections 3.2 and 3.2.1), a comparison of different sampling strategies (Section 3.2.2) and the tracking of plane hypotheses (Section 3.3). Finally, results of the experiments are shown in Section 3.5.

3.1 State of the Art

There exist various approaches for plane detection in uncalibrated image pairs exist. Most of them use a hypothesize-and-test framework. A popular method for detecting multiple models is to use the robust estimation method RANSAC [FB81], to sequentially fit the model to a data set and then to remove inliers. To generate plane hypotheses, Vincent et al. [VL01] studied different methods to select groups of four points which are likely to be coplanar. Beside a totally random selection, they propose to check the interest point configuration to avoid degenerate cases. Therefore, they compare the area of the four triangles defined by the four points. Another possibility to increase the likelihood that the points belong to the same plane is to select points which lay on two different edges in an image. In contrast, Kanazawa et al. [KK04] define a probability for feature points to belong to the same plane. The idea is to use a uniform distribution to sample an initial set of points and then to compute a conditional probability using the Euclidean distance between the initially selected points and the remaining points. According to these probabilities, for each initially selected point three additional points are sampled and thus the chosen groups of points are located in a local area. Both approaches use a RANSAC scheme, iteratively detect the dominant plane, remove the inliers and proceed with the remaining interest points.

The success of the plane computation depends on the coplanarity of four matched points. Nicolas et al. [LNGPS05] propose to use the knowledge of the first homography to guide the computation of further homographies and thus reduce the set of points/lines to three pairs. After the first iteration, they use the geometric constraint of the first homography and three points to generate further plane hypotheses. The selection scheme is also iterative and RANSAC based. Piazzi et al. [PP06] also need only three corresponding points. They propose to use two cameras aligned to the same orientation to compute the normal vector to a plane. The normal vector is then used to cluster triangles. This approach does not use RANSAC, but instead clusters sequentially similar neighboring normal vectors of Delaunay triangles. Lourakis et al. [LAO02] use results from projective geometry to automatically detect planes. They exploit the fact that every pair of a 3D line and a 3D point defines a plane. First, they estimate the fundamental matrix and the epipoles. Then the homography is computed for each set of point and line feature and a voting scheme is applied, where each homography transfers all features from the first image to the second. The homography with the highest number of inliers is selected and refined using Least Median of Squares. This approach also detects a plane and removes the inliers in an iterative scheme.

Fraundorfer et al. [FSB06] start with a set of sparse affine invariant corresponding seed regions and iteratively expand and refine the plane-induced homographies. This image-driven method simultaneously solves the region segmentation and the matching problem for planar parts of a scene containing an unknown number of planar regions. The idea is to use affine invariant regions for initialization. In an iterative manner Harris corners of the regions are detected to compute a homography, the region is expanded based on correlation of grey-values and the homography is re-estimated. The iterative algorithm is stopped if no new points can be added. This approach leads to a pixel-wise image segmentation in contrast to sparse clusters of features.

More recent approaches concentrate on robust estimation of multiple structures. Toldo et al. [TF08] propose j-linkage, an approach based on random sampling and conceptual data representation. Each point is represented with the characteristic function of the set of random models that fit the point. The method starts with random sampling. M hypotheses are generated by sampling a minimal set of data points necessary to estimate the model. Then the consensus set for each model is computed, i.e., the set of points with a distance from the models smaller than a threshold ϵ. Agglomerative clustering is used to group points belonging to the same model. In [FSB10] this method is used for the robust detection and matching of multiple planes. For hypothesis generation random sampling is used. Then Fouhey et al. use j-linkage for clustering homographies followed by a global merging. Since most of the initial hypotheses are generated from groups of nearby points, globally visible aspects of the perspective transformation are underdetermined. Thus agglomerative clustering is continued using a different error function which measures the average error for the model. Finally, a spatial analysis is used

to split overlapping hypotheses. In [CWS09], Chin et al. propose a novel Mercer kernel for the robust estimation problem which elicits the potential of two points to have emerged from the same underlying structure and thus can cope with more than 90% outliers. This approach consists of two steps: First they propose to remove gross outliers, followed by a robust fitting of multiple structures. To remove gross outliers, a mercer kernel is used for robust fitting before the data is analyzed using a Singular Value Decomposition (SVD). To fit multiple structures to the remaining inliers they propose a Principle Component Analysis (PCA) and spectral clustering step followed by a structure merging scheme where clusters are merged if the estimated model still satisfactorily "explains" the resulting structures. While random sampling is used to generate hypotheses, principal component analysis and spectral clustering are applied for robust fitting. The methods of Toldo et al. [TF08], Fouhey et al. [FSB10] as well as Chin et al. [CWS09] use clustering schemes and avoid removing inliers and iterative detection of planes.

Given a fixed threshold to detect inliers, incremental methods favor planes detected first over subsequent planes by greedily consuming features. Recently developed approaches overcome this drawback by treating hypotheses equally, but plane hypotheses have to be created independently of each other and thus it is not possible to restrict the search space, which leads to higher computational complexity. We propose model selection based on the MDL principle: Instead of creating all hypotheses at once, pruning models and then using model selection, we propose to embed model selection in an incremental scheme and thus guide randomized selection of interest points to compute more likely plane hypotheses. This allows us to avoid an additional hypotheses pruning step without decrease of performance. Finally, this formulation allows us to explicitly introduce priors, hence we can detect and track planes in one scheme which is not possible in any of the approaches described above.

3.2 Iterative Plane Hypotheses Selection

We develop a method to detect multiple planes in image pairs. Typical approaches (cf. Vincent et al. [VL01], Kanazawa et al. [KK04] or Lourakis et al. [LAO02]) use interest points to find the corresponding point in image pairs. Groups of four points are used to generate plane hypotheses which are represented by the two dimensional projective transformation (homography). A popular method to achieve robustness against noise is RANSAC proposed by Fischler et al. [FB81]. Algorithm 1 shows a typical implementation. An important role plays the inlier ratio $\epsilon = I_{max}/N$ which is the ratio of the inliers of the correct solution and all available correspondences. If this ratio is known, the probability that a plane with $m = 4$ point pairs is correct is ϵ^m, and that in k iterations no correct solution is found is

$$\eta = (1 - \epsilon^m)^k. \tag{3.1}$$

Algorithm 1 Dominant plane detection with RANSAC

```
k ← 0, ϵ ← m/N, I_max ← 0
while η = (1 − ϵ^m)^k ≥ η_0 do
  Sample m point pairs
  Compute the plane P′
  Count number of explained interest points (inliers) I for P′
  if I > I_max then
    I_max ← I
    ϵ ← I_max/N
    P ← P′
  end if
  k ← k + 1
end while
```

Typically the true inlier ratio is unknown, thus a well known strategy is to use the best estimate available up to now. To detect multiple planes, this algorithm can be embedded in a sequential loop. In each iteration, the explained interest points are removed and the algorithm starts again with the remaining point pairs.

Given a fixed threshold to detect inliers, sequential methods favor planes detected first over subsequent planes by greedily consuming features. If all hypotheses are created first and interest points have to compete, this drawback is overcome. But generating hypotheses first does not allow us to restrict the search space. Hence, in complex environments the number of random hypotheses grows, finding the ideal solution gets intractable and finer scene details are missed. Additionally, there is no possibility to estimate the inlier ratio.

We therefore propose to embed Minimal Description Length (MDL) based model selection in an iterative scheme. Existing planes compete with newly created hypotheses to ensure that interest points are assigned to the best currently available hypothesis. Additionally hypothesis generation can be guided to unexplained regions. This method avoids the bias towards dominant planes that is typical for iterative methods and it limits the search space, which leads to a faster explanation of the entire image in terms of piecewise planar surfaces.

Algorithm 2 shows our proposed method for plane detection. In each iteration, a small number Z of new plane hypotheses P' are computed which have to compete with the selected hypotheses P of the last iteration. The termination criterion is based on the true inlier ratio ϵ and the number of samples M which are necessary to compute the homographies. As long as we do not know these values, we use the best estimate available up to now. For ϵ, that is, the ratio of the number of explained interest points I_{max} of the current best plane hypotheses and the number of matched interest points N to explain.

A critical question is how to set M? From a global point of view, i.e., if hypotheses are distributed uniformly and there is no bias or guiding, it would be

Algorithm 2 Plane detection using model selection

```
P ← 0, P' ← 0
k ← 0, ϵ ← M/N, I_max ← 0
while η = (1 − ϵ^M)^k ≥ η_0 do
  P' ← P
  Add Z random plane hypotheses to P' (Section 3.2.2)
  Select plane hypotheses from P' and store in P (Section 3.2.1)
  Count number of explained interest points (inliers) I for P
  if I > I_max then
    I_max ← I
    ϵ ← I_max/N
  end if
  k ← k + 1
end while
```

feasible to estimate M with the number of plane hypotheses currently selected multiplied with the minimal set of interest points $m = 4$ to compute one plane homography. In case there are just a few dominant planes in the scene this would be possible, but, because of Equation 3.1, with a growing number of planes this does not converge within the desired "realtime" framerate. In Section 3.2.2 we show that in case sampling is biased to unexplained regions M can be set to four without a loss of performance. The reason for this could be that in each iteration only one additional new hypothesis from unexplained points is computed and added to the model selection. We will use this value for all experiments. Furthermore, in Algorithm 2 k is the number of iterations, η stands for the probability that no correct set of hypotheses is found and the parameter η_0 is the desired failure rate. Due to the incremental scheme, it is possible to guide the computation of new hypotheses to unexplained regions.

3.2.1 Hypotheses Representation and Model Selection

In each iteration, selected homographies of the last iteration have to compete with newly sampled hypotheses. For the selection, the idea is that the same feature cannot belong to more than one plane and that the model cannot be fitted sequentially. Thus an over-complete set of homographies is generated and the best subset in terms of a Minimum Description Length criterion is chosen.

Minimum Description Length (MDL)

The core of the problem is to find a general mechanism to optimally describe the data with respect to an objective function and to reduce the number of redundant models. Hence, the objective function has to formalize a trade-off between the complexity of the representation and a general notion of simplicity. The notion

of simplicity has a long history in Gestalt psychology, which has found its way in different vision systems (e.g. Lowe [Low87], Zillich [Zil07]). In the field of information theory, this led to the principle of Minimum Description Length (MDL) (Rissanen [Ris84]). The purpose of MDL is to discover regularities in the observed data and hence to describe the data with the shortest possible encoding. In image processing and computer vision, this principle can be used to select a subset of all recovered models and thus to describe parts of the image or possibly the whole image.

The basic mathematical tool for the purpose of range image segmentation is introduced by Leonardis et al. [LGB95] and adapted in [LLS08]. In the following, we briefly describe the ideas, which are then reformulated for plane detection in the next section.

According to Leonardis et al. [LGB95], the length of encoding the image

$$L_{image}(\mathbf{n}) = L_{pointwise}(\mathbf{n}) + L_{models}(\mathbf{n}) \tag{3.2}$$

can be formalized as the sum of the length of encoding of individual data points that are not described by the model $L_{pointwise}(\mathbf{n})$ and the length of encoding of data described by the selected models $L_{models}(\mathbf{n})$. The vector $\mathbf{n}^T = [n_1, n_2, ..., n_M]$ stands for a set of models, with $n_i = 1$ if a model is selected and $n_i = 0$ otherwise. If a model is represented by an image region with $S_{data,i}$ elements and the error measure $S_{error,i}$ Equation 3.2 leads to

$$L_{image} = K_1(S_{all} - S_{data}(\mathbf{n})) + K_2 S_{error}(\mathbf{n}) + K_3 S_{model}(\mathbf{n}) \tag{3.3}$$

where S_{all} denotes the number of all data points in the image and S_{data} refers to the number of data points explained by the selected models. S_{model} stands for the parameters which are needed to describe the models and S_{error} is the error, added by the models. The constants K_1, K_2 and K_3 are weights which can be determined on a purely information-theoretical basis (in terms of bits), or they can be adjusted in order to express the preference for a particular type of description.

Now, the task is to minimize the cost of encoding the data L_{image}. Since S_{all} is constant this is equivalent to maximizing the expression

$$S_H(\mathbf{n}) = K_1 S_{data}(\mathbf{n}) - K_2 S_{error}(\mathbf{n}) - K_3 S_{model}(\mathbf{n}) \tag{3.4}$$

Intuitively, this formulation shows that an encoding is efficient if the number of data points described by a model is large, the contributed error is low, and the number of parameters is small.

MDL for Plane Hypotheses Selection

In practice, the weights K_1, K_2 and K_3 of Equation 3.4 are related to the average cost of the data points, the model and the error, and we only need to consider the

relative savings between different combinations of hypotheses. Hence, to select the best model, the savings for each individual hypothesis H can be expressed as

$$S_H = S_{data} - \kappa_1 S_{model} - \kappa_2 S_{error} \tag{3.5}$$

where $\kappa_1 = \frac{K_3}{K_1}$ and $\kappa_2 = \frac{K_2}{K_1}$. In our case S_{data} is the number of interest points N explained by H and S_{model} stands for the cost of coding the model itself. We use one model (the homography of a plane) and thus $S_{model} = 1$. S_{error} describes the cost for the error added, which we express with the log-likelihood over all interest points f_i of the plane hypothesis H. Experiments indicate that the Gaussian error model

$$p(f_i|H) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\epsilon_i^2}{2\sigma^2}\right) \tag{3.6}$$

in conjunction with an approximation of the log-likelihood comply with our expectations. Thus the cost of the error results in

$$S_{error} = -\log \prod_{i=1}^{N} p(f_i|H) = -\sum_{i=1}^{N} \log(p(f_i|H)) \tag{3.7}$$

$$= -\sum_{i=1}^{N}\sum_{n=1}^{\infty} \frac{1}{n}(1 - p(f_i|H))^n \approx N - \sum_{i=1}^{N} p(f_i|H) \tag{3.8}$$

where $\log(p(f_i|H))$ is the log-likelihood that an interest point belongs to the plane. For ϵ_i we use the Euclidean distance of inliers to the estimated homography. Substitution of Equation 3.8 into Equation 3.5 yields the merit of a model

$$S_H = -\kappa_1 + \sum_{k=1}^{N} ((1 - \kappa_2) + \kappa_2 p(f_k|H)) \tag{3.9}$$

An interest point can only be assigned to one model. Hence, overlapping models compete for interest points which can be represented by interaction costs

$$s_{ij} = -\frac{1}{2} \sum_{f_k \in H_i \cap H_j} ((1 - \kappa_2) + \kappa_2 \min\{p(f_k|H_i), p(f_k|H_j)\}) \,. \tag{3.10}$$

Finding the optimal possible set of homographies for the current iteration leads to a Quadratic Boolean Problem (QBP)[1]

$$\max_n \mathbf{n}^T S \mathbf{n} \;,\; S = \begin{bmatrix} s_{11} & \cdots & s_{1N} \\ \vdots & \ddots & \vdots \\ s_{N1} & \cdots & s_{NN} \end{bmatrix} \tag{3.11}$$

[1] QBP assumes pairwise interaction, which in our case can be violated. But this is still a good approximation because interaction always increases cost, yielding a desirable bias against weak hypotheses.

where $\mathbf{n}= [n_1, n_2, \cdots, n_N]$ stands for the indicator vector with $n_i = 1$ if a plane hypothesis is selected and $n_i = 0$ otherwise and $s_{ii} = S_{H,i}$ represents a merit term of a plane hypothesis. The time to solve the QBP grows exponentially with the number of hypotheses. Several methods have been proposed to solve the problem with an approximate solution. Our results indicate that for our specific problem a greedy approximation gives good results (cf. Section 3.5.2). But still, what is most important is to keep the number of hypotheses tractable. We addressed this by embedding the model selection in an iterative algorithm and hence, the solution can be found very fast.

3.2.2 Hypotheses Generation and Efficiency

One of the key issues of approaches that use random samples is to select *good* features. Our method addresses this issue in different ways. Following Myatt et al. [MTN+02], sampling is biased to features that are most likely located on the same plane. The second strategy is to sequentially guide sampling towards unexplained regions. Furthermore, we use a pre-filter which selects good hypotheses and adds them to the iterative model selection. In the following sections, we describe the different sampling strategies, the proposed filter to select good plane hypotheses and finally compare the impact to uniformly distributed sampling.

Uniformly Distributed Sampling

One possibility to compute plane hypotheses is to sample features uniformly. This method is often used for robust object detection or pose estimation, where the percentage of outliers is known to be lower than 50%. A typical setting is to estimate the affine homography of image points matched with a model. Then the number of attempts to select outlier free samples is

$$k = \frac{\log(1-p)}{\log(1-\epsilon^m)} \tag{3.12}$$

If the desired confidence to obtain outlier free samples is $p = 99\%$ and if the data is contaminated with 50% noise, for an affine model ($m = 3$) 35 trials are necessary. In our case, the setting is much more complex. In one of the test images shown in Figure 3.1 we marked 10 ground truth planes. To compute the plane homography, we need $m = 4$ point pairs. If we assume that all 10 planes have equal size, and we also want a desired confidence of $p = 99\%$ and the data consists of only 20% noise, there are 112429 trials necessary to compute one plane. Hence, uniformly distributed sampling does not lead to satisfying results within a given timeframe.

Sampling Biased Towards Near Adjacent Points

Instead of uniformly distributed sampling, Myatt et al. [MTN+02] propose to bias random selection towards clusters in a multi-dimensional space. If a selected point

Figure 3.1: Test image with ground truth overlay

A is an inlier, then there will be an increased probability of a point adjacent to A also being an inlier. Following this approach we first select an interest point A randomly. Then all other points are ordered by increasing Euclidean distance from A and three additional nearby points are randomly selected, with a sampling probability depending on their position in the sorted list using a Gaussian distribution.

Sampling Biased to Features with a Similar Motion Vector

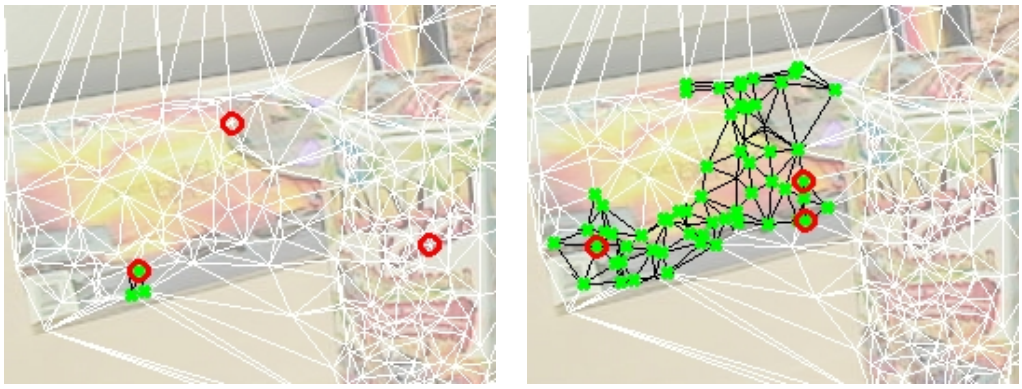
Another heuristic, which significantly improves the performance, is to sort the points depending on the motion vector. The motion vector describes the shift of a specific point between two images. The method proposed in the last section sorts points depending on the Euclidean distance of an initial selected point. Here we propose to select a point A and sort all other points depending on the similarity of the motion vector to the first point. The selection is also biased to similar motion with a Gaussian distribution.

Sampling Biased to Unexplained Regions

The last three methods focus on increasing the probability of selecting three points which lie on the same plane as a point A, selected first. The overall approach is concerned with describing the whole scene with planes. Thus, if a plane is found it seems to be plausible to bias sampling towards unexplained regions. Our iterative model selection schema perfectly supports this. In each iteration explained interest points are marked and points are sampled in unexplained regions. As described in Section 3.2, newly generated plane hypotheses have to compete with previously selected ones.

Pre-Filter Good Hypotheses

Although we propose a sequential algorithm, there are many samples necessary to reliably detect planes in a complex scene. Hence, it is important to prune plane



(a) Bad hypothesis, growing stops (b) Good hypothesis

Figure 3.2: Connected components filter for early pruning of plane hypotheses. A 2D Delaunay triangulation is used to connect interest points (white edges). Three points are sampled (red) and an affine homography is computed. The graph is traced and points supporting the affine mapping are clustered (green). A Hypothesis is accepted if all initial sampled points are connected within the cluster (right image).

hypotheses as early as possible. We propose a connected components analysis of points supporting a hypothesis. Therefore, neighboring interest points are connected using a 2D Delaunay triangulation. To further reduce the complexity of the initial model, only three points are sampled and an affine homography is computed (6 degrees of freedom). Starting from one of the sampled points, the graph is expanded and interest points which support the affine transformation are clustered until a cut-off threshold is reached. If all three initial sampled points are visited during clustering, this hypothesis is considered as good. Otherwise the algorithm starts again with a new set of three points. In case a good hypothesis is found, a least squares homography (8 parameters) is computed using the Direct Linear Transform (DLT) algorithm proposed by Hartley [Har08] (see Appendix A), the graph is further expanded and this plane hypothesis is considered for the subsequent model selection. Algorithm 3 summarizes the proposed plane hypothesis generation. In Figure 3.2(a) a typical bad hypothesis is shown. Three initial pairs of interest points are shown in red. The clustering of interest points (green dots) immediately stops because one of the three points lies on a totally different surface. In contrast, in Figure 3.2(b) the cluster of interest points (colored green) also contains the initial sampled points (red). In our test images, only about 3% of the plane hypothesis are refined and passed on to the subsequent model selection.

The comparison in Figure 3.3 shows the improvement when the sampling is biased towards unexplained interest points. While in the test scenario shown in Figure 3.1, uniform sampling does not exceed a tp-rate of 0.3, a bias towards near adjacent points improves the tp-rate to 0.6. It is interesting to note that if a bias to

Algorithm 3 Connected component filter for early pruning of plane hypotheses

```
Create 2D neighborhood graph using the Delaunay triangulation
while No good plane hypothesis found do
  Sample 3 interest point pairs
  Compute affine transformation A (6 parameters)
  Trace graph and cluster interest points which support A
  if Cluster contains initial 3 sampled points then
    Good hypothesis found
    break
  else
    No hypothesis found
    continue
  end if
end while
Compute least-squares homography (8 parameters) using the DLT algorithm
Continue clustering and store plane hypothesis for iterative model selection
```

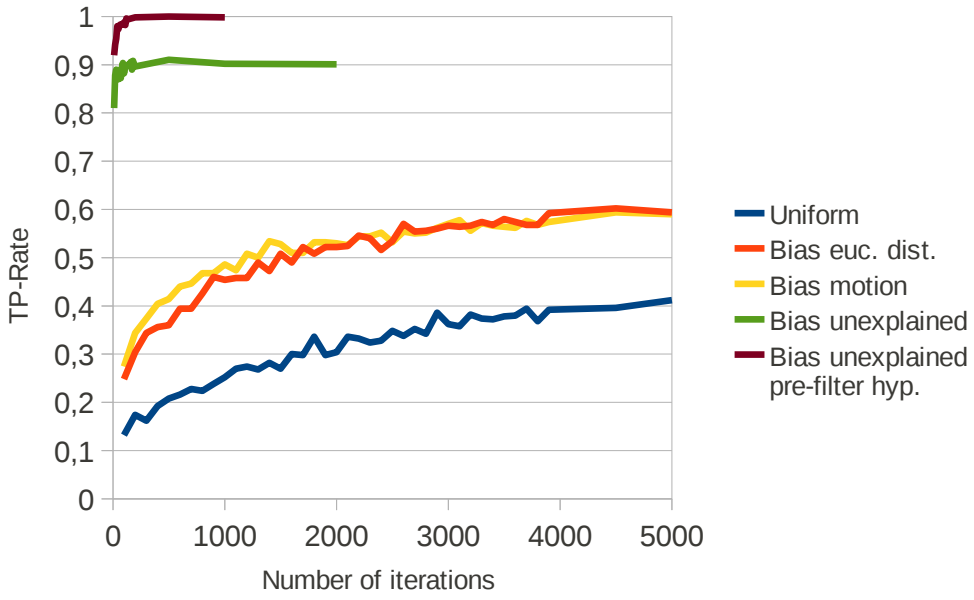


Figure 3.3: Sampling strategies and pre-filtering of hypotheses

a similar motion vector is used, the tp-rate is slightly higher for a lower number of iterations. The reason for this might be that for big planes, which are detected first, the interest points are distributed more uniformly over the plane, while in contrast the results are more unstable if near adjacent points are selected. As expected, incrementally adding hypotheses in unexplained regions drastically improves the quality. In combination with the connected component analysis, this method has a tp-rate higher than 0.99 with a low number of only 120 filtered hypotheses.

3.3 Tracking of Clusters

One of the key benefits of our algorithm is that prior knowledge can be introduced easily. We exploit this in image sequences where detected planes are propagated to subsequent frames. For this, the interest points of planes detected in the previous image pair are matched with interest points of the current frame, followed by a robust homography estimation using least median of squares (LMedS)[2]. Thus Algorithm 2 is extended with tracked planes $P_{tracked}$, which are used to initialize P. Following $P_{tracked}$, the initialization value of the inlier ratio ϵ increases to the number of accumulated interest points of the tracked planes divided by the total number of matched interest points. Hence, plane detection already starts with an initial guess of planes which have to survive the following hypothesis selection stage.

3.4 Post-Processing of the Planes

Plane hypotheses often capture interest points that match the underlying homography by chance. To account for this, we introduce a post-processing step with the goal to assign features which support more than one homography to an individual plane and additionally split planes because of an interest point neighborhood constraint, using the Delaunay triangulation.

3.4.1 Belief Propagation for Feature/Plane Assignment

An straight-forward solution to assign interest point pairs that support more than one homography to an individual plane is to select the point with the minimal error to the transferred model point. In cases where this assignment is ambiguous, i.e., the error in several planes is similar, the assignment of the adjacent points can be considered. This can be formalized with a Markov random field (MRF) framework. Felzenszwalb et al. [FH06] use a MRF framework to solve early vision problems, such as image restoration and stereo image registration. We use this method to assign interest points to individual planar patches. Because of the irregular spatial distribution of the interest points, we use the Delaunay triangulation to define the neighborhood graph, instead of the grid graph proposed by Felzenszwalb et al. [FH06]. Thus the general goal is to assign plane labels $l \in L$ to interest points $f \in F$ with respect to an energy function

$$E(l) = \sum_{f_1 \in F} D_{f_1}(l_{f_1}) + \sum_{(f_1,f_2)\in N} W(l_{f_1}, l_{f_2}) \tag{3.13}$$

[2]For robust estimation of homographies, we use an implementation by Manolis Lourakis [Lou06].

	belief prop.	min. error	no post-processing
precision (p_{pr})	0.980	0.970	0.946
wrong assignments per image ($n_{f,fp}$)	5.53	8.435	16.7

Table 3.1: Comparison of feature assignment methods.

where N are the undirected edges of the interest point neighborhood graph, $D_{f_1}(l_{f_1})$ is the data cost of assigning plane label l_{f_1} to interest point f_1 and $W(l_{f_1}, l_{f_2})$ measures the cost of assigning the labels l_{f_1} and l_{f_2} to two neighboring interest points. To approximate the MAP solution to MRF problems, a max-product algorithm can be used (see [WF01]). The reason why the algorithm only approximates the MAP solution is that the graph is cyclic and therefore Loopy Belief Propagation is used. For a non-cyclic graph, Belief Propagation would return the exact result. We use the negative log probability of Equation 3.6, thus the max-product becomes a min-sum. Felzenszwalb et al. [FH06] have shown that the minimization problem can be solved by iteratively passing messages

$$m^t_{f_1 \to f_2}(l_{f_2}) = \min_{l_{f_1}} \left(W(l_{f_1}, l_{f_2}) + D_{f_1}(l_{f_1}) + \sum_{s \in N(f_1) \setminus f_2} m^{t-1}_{s \to f_1}(l_{f_1}) \right) \tag{3.14}$$

around the graph. $m^t_{f_1 \to f_2}(l_{f_2})$ is a vector whose dimension is given by the number of available planes and stands for a message that a feature f_1 sends to a neighboring feature f_2 at iteration t. For first experiments we set $W(l_{f_1}, l_{f_2}) = 0$ if the features f_1 and f_2 have the same label and in case they have different labels we set $W(l_{f_1}, l_{f_2}) = c$ to a constant cost value c. After T iterations the final assignment of interest points and plane labels is the minimum of the individual label elements of the vector

$$b_{f_2}(l_{f_2}) = D_{f_2}(l_{f_2}) + \sum_{f_1 \in N(f_2)} m^T_{f_1 \to f_2}(l_{f_2}). \tag{3.15}$$

In Table 3.1 belief propagation and best fit (min. error) assignment are compared with results without post-processing. It can be seen that because of the normalization the difference in precision is rather small but the absolute number of incorrect feature assignments is significantly reduced.

3.4.2 Splitting of Clusters

Another method to reduce the number of falsely assigned features is to analyze the distribution of the interest points. The assumption is that interest points are uniformly distributed on the surface of objects and thus large distances between neighboring interest points are a hint for incorrect assignment. Hence, we build a neighborhood graph of the interest points of a plane using the Delaunay triangulation. Then the mean and the standard deviation of the distance between connected

Figure 3.4: A plane (top surface of packaging) accidentally picks up interest points in the background. Therefore we split the interest point clusters (colored dots) using the mean and standard deviation of edges of a Delaunay graph (white edges).

interest points in image coordinates are computed and edges longer than s times the standard deviation are removed. For each split plane hypothesis, we verify that the support is large enough, i.e. the merit still surpasses κ_1. We found that for our scenarios a factor of $s = 1$ works best and thus keep it fixed for all following experiments. Figure 3.4 shows the edges of the Delaunay neighborhood graph in white and the plane hypothesis split into two groups of interest points. The dark cluster is accepted and the weaker white group is rejected, since it does not surpass the base cost κ_1.

3.5 Tests and Evaluation

To test our method, we use two completely different sets of images. Motivated by our cognitive robotic scenarios, the first set of images show packaging of arbitrary shapes typically found in a supermarket (see Figures 3.8). We placed each object in front of a weakly textured background as well as in a highly cluttered scene. For comparison, we additionally test the system with the houses data sets published by the Visual Geometry Group at the University of Oxford [Oxh] (see Figures 3.10). To get ground truth, we manually marked all planes in the foreground and the dominant ones of the background, resulting in 231 planes in 56 images. To test the tracking capability of our method, the packaging data set consists of 8 sequences with 4 subsequent images, whereas we use 6 sets from Oxford with also 4 images, but these images are not ordered in a sequence.

For all experiments we use SIFT, the well known interest point detector/descriptor proposed by Lowe [Low04]. SIFT features are matched in image pairs using the Euclidean distance of the descriptors and matches are accepted if the NNDR (nearest/next distance ratio) d is below 0.8. To compute the homography, we follow Hartley [Har08], i.e., points are normalized to zero mean and scaled to get an average distance of $\sqrt{2}$ from the origin. Then the homography is estimated using the Direct Linear Transform (DLT) algorithm.

Three numbers are computed to compare the methods. The first is the feature based precision

$$p_{pr} = \frac{n_{f,tp}}{n_{f,tp} + n_{f,fp}} \tag{3.16}$$

which is the ratio of the number of inliers $n_{f,tp}$ correctly located on a ground truth plane and the total number of features per detected plane $n_{f,tp} + n_{f,fp}$. The second number is the oversegmentation rate

$$p_{ov} = \frac{n_{p,fp}}{n_{p,tp} + n_{p,fp}} \tag{3.17}$$

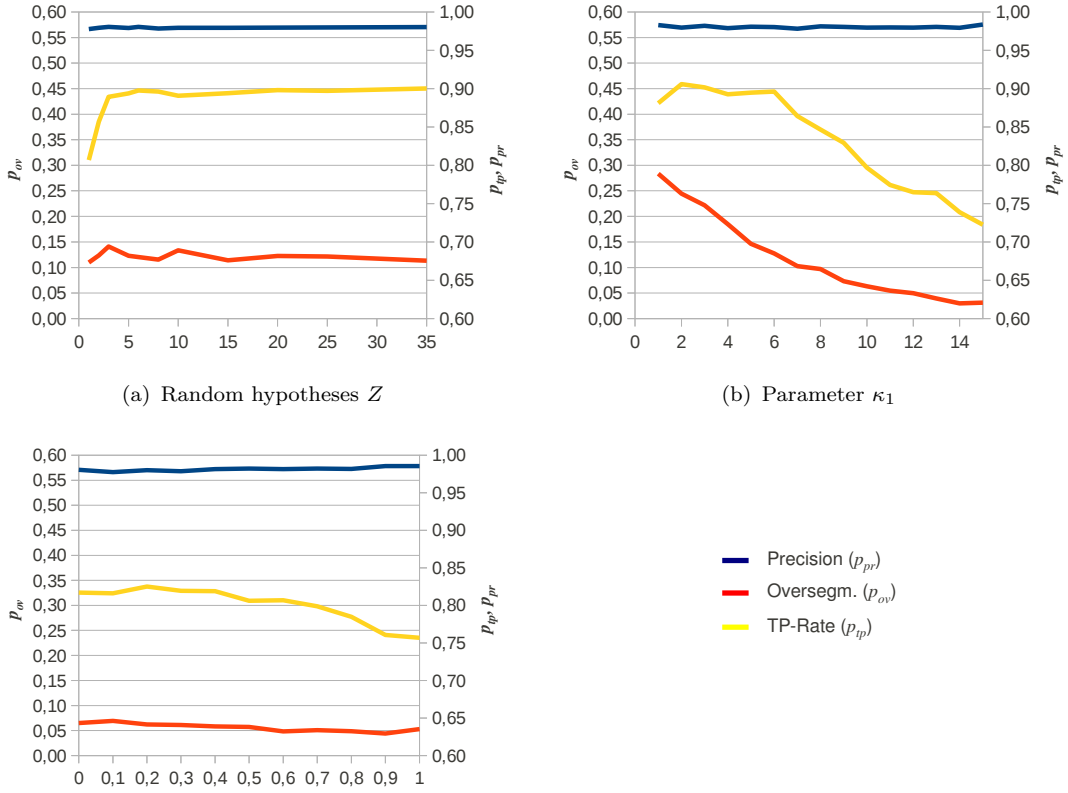
per plane which indicates whether an algorithm splits a plane into several parts. $n_{p,fp}$ the number of false positives is the number of detected planes minus the number of correctly detected planes $n_{p,tp}$. Furthermore, we compute the plane based true-positive rate (tp-rate)

$$p_{tp} = \frac{n_{p,tp}}{n_{p,tp} + n_{p,fn}} \tag{3.18}$$

which describes the ratio of the correctly detected planes $n_{p,tp}$ and the total number of planes $n_{p,tp} + n_{p,fn}$.

3.5.1 Parameter Optimization

To analyze the influence of the parameters of the proposed method, we tested it with the first half of the packaging data set. We vary the parameters: number of random hypotheses $Z = [1...35]$, $\kappa_1 = [1...15]$ and $\kappa_2 = [0...1.]$ and plot the performance measures. Figures 3.5(a), 3.5(b) and 3.5(c) show that our algorithm, in particular with respect to precision, is robust against variation of the parameters. While the over-segmentation-rate in Figure 3.5(a) is almost constant, the precision slightly increases at the beginning and the tp-rate reaches a peak at $Z = 3$. The Parameter κ_1 mostly influences the over-segmentation-rate and the tp-rate. We set $\kappa_1 = 6$ to the maximum of p_{tp}, where p_{ov} is already low. In Figure 3.5(c) it can be seen that the Parameter κ_2 is stable in a wide range. We set $\kappa_2 = 0.4$, where the tp-rate has a maximum.



(a) Random hypotheses Z

(b) Parameter κ_1

(c) Parameter κ_2

Figure 3.5: Parameter optimization

method	p_{pr}	p_{ov}	p_{tp}	savings
greedy	0.978	0,021	0.944	323.6
brute force	0.978	0,004	0.966	323.8

Table 3.2: Comparison of a greedy solution and the exact brute force computed solution of the QBP.

3.5.2 Comparison of a Greedy and the Exact Brute Force Solution of the QBP

To evaluate the performance of a greedy approximate solution of the Quadratic Boolean Problem (QBP) from Section 3.2.1, we compute the feature based precision p_{pr}, the over-segmentation-rate p_{ov}, the true-positive rate p_{tp} and the total savings S (see Equation 3.11) for our algorithm. Table 3.2 shows that there is a very small decrease in performance for the approximate solution.

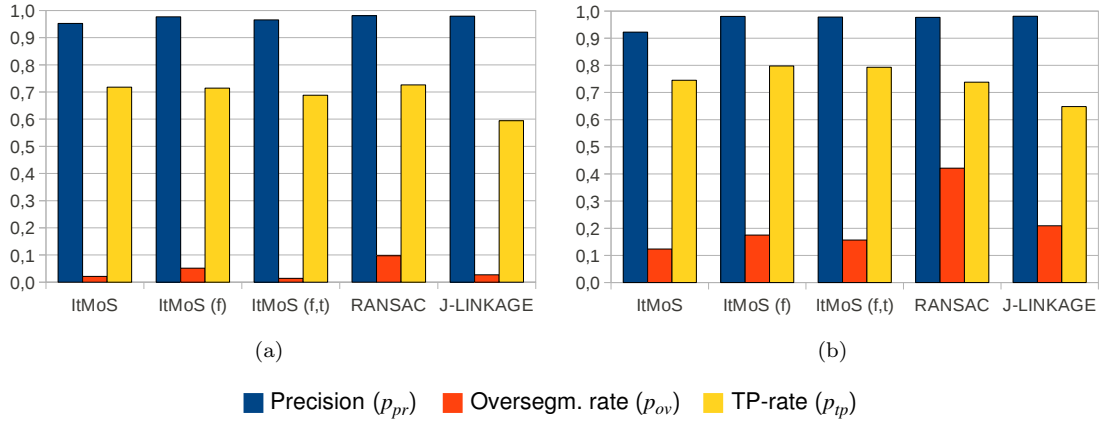


Figure 3.6: Comparison of plane detection methods. Left graph shows the plane detection result for images with no background texture. The test images of the right graph have a highly textured background.

3.5.3 Plane Detection

For the experiments all images of our packaging data set and the Oxford houses data set [Oxh] are used. We compare the proposed methods with a sequential *RANSAC* and *J-LINKAGE*. The *RANSAC* based method detects a dominant plane and marks supporting features, which are then excluded in the following iterations. *J-LINKAGE* is an implementation according to Fouhey et al. [FSB10]. With *ItMoS* we refer to the proposed iterative model selection algorithm (see Section 3.2). For the tests, sampling of interest points is biased to near adjacent points and to unexplained regions (see Section 3.2.2). Additionally, *ItMoS (f)* stands for our method including the connected components based pre-filter and *ItMoS (f,t)* refers to our method including tracking of planes in image sequences.

The experimental evaluation shows that our model selection method outperforms the other methods in terms of tp-rate and lower over-segmentation, especially for complex scenes. Although it is not optimized for outdoor environments of the Oxford houses, it competes with the other methods. The incremental RANSAC approach has a slightly higher tp-rate at the cost of over-segmentation. If one compares Figure 3.6(a) with the Figures 3.6(b) and 3.7(b), an interesting detail can be seen. Although we use the same post-processing step for all of the methods (see Section 3.4), in the case of highly cluttered images over-segmentation increases especially with the RANSAC based method, while it remains low for the ItMoS methods. Comparing Figure 3.6(a) and 3.6(b) it seems that all methods have a higher tp-rate in case of a cluttered background. For foreground objects, some of the marked ground truth planes are very small and thus easily missed, while the background planes of the cluttered scenes are generally rather large and thus more easily detected, which explains the higher overall tp-rate for these scenes.

The Tables 3.3, 3.4, 3.5, 3.6 show the numerical results depicted in Figures 3.6

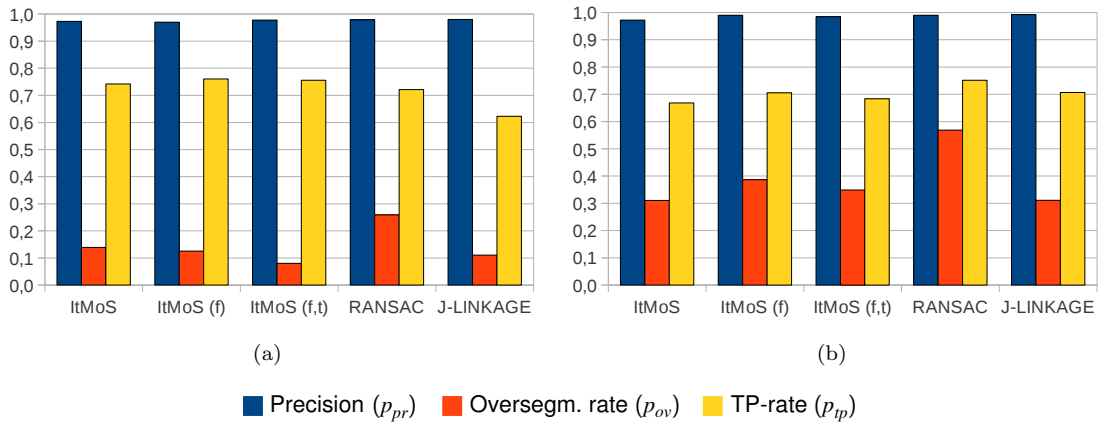


Figure 3.7: Results for all our images (left) and for the Oxford houses data set (right).

method	p_{pr}	p_{ov}	p_{tp}	$t_{processing}$ [s]	$n_{samples}$ [1/image]
ItMoS	0.952	0.021	0.718	1.4	2814
ItMoS (f)	0.977	0.051	0.714	1.6	1486 (24318)
ItMoS (f.t)	0.966	0.014	0.688	1.0	858 (14476)
RANSAC	0.981	0.097	0.726	0.7	1793
J-LINKAGE	0.980	0.030	0.590	5.7	5000

Table 3.3: Results for the packaging data with a weakly textured background.

method	p_{pr}	p_{ov}	p_{tp}	$t_{processing}$ [s]	$n_{samples}$ [1/image]
ItMoS	0.923	0.123	0.745	7.4	6629
ItMoS (f)	0.980	0.175	0.798	3.8	672 (30904)
ItMoS (f.t)	0.978	0.156	0.793	2.5	481 (18903)
RANSAC	0.977	0.421	0.738	5.4	16886
J-LINKAGE	0.980	0.210	0.650	27.8	5000

Table 3.4: Results for the packaging data with a highly textured background.

method	p_{pr}	p_{ov}	p_{tp}	$t_{processing}$ [s]	$n_{samples}$ [1/image]
ItMoS	0.973	0.139	0.742	8.7	8950
ItMoS (f)	0.970	0.126	0.761	2.5	1017 (26528)
ItMoS (f,t)	0.978	0.080	0.756	1.7	639 (17897)
RANSAC	0.979	0.259	0.722	3.0	9320
J-LINKAGE	0.980	0.110	0.620	16.0	5000

Table 3.5: Results for the packaging data set.

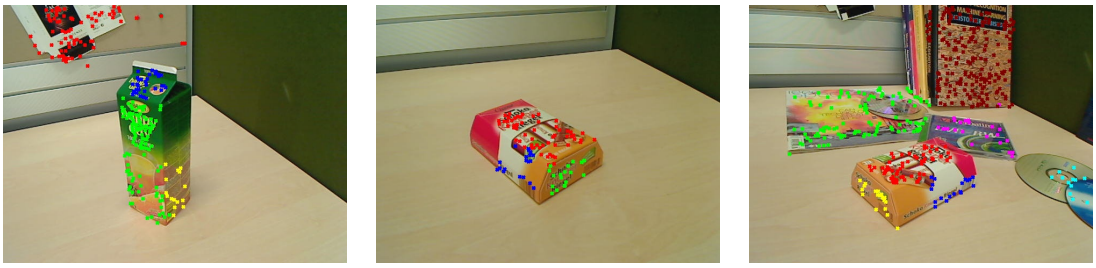
method	p_{pr}	p_{ov}	p_{tp}	$t_{processing}$ [s]	$n_{samples}$ [1/image]
ItMoS	0.97	0.310	0.668	13.4	5967
ItMoS (f)	0.99	0.39	0.71	17.7	1856 (75593)
ItMoS (f.t)	0.98	0.35	0.68	13.6	1341 (62932)
RANSAC	0.990	0.568	0.751	88.8	292995
J-LINKAGE	0.992	0.311	0.707	67.1	5000

Table 3.6: Results for the Oxford houses data set.

and 3.7. In these tables, $t_{processing}$ stands for the mean processing time per image without the time needed for computation of the interest points. The experiments have been performed with a laptop with an Intel(R) Core(TM)2 Duo CPU T7500 (2.20GHz, bogomips 4388.83). Furthermore, $n_{samples}$ is the mean number of samples generated per image. For the methods with a pre-filter (ItMoS (f), ItMoS (f,t)), the first number is the number of samples after filtering and the number within brackets is the total number of generated samples. It can be seen that much more hypotheses can be analyzed within a shorter time and only about 3% are passed on to the model selection stage. Comparing ItMoS and sequential RANSAC, it can be seen that although ItMoS converges faster and the mean number of random samples per image is lower the tp-rate is higher. One reason for this is that the incremental filtering out of interest points which support planes detected first by the RANSAC method leads to a decreasing inlier ratio and thus to an increasing number of samples for planes detected later. In contrast, ItMoS treats all planes simultaneously and thus the number of samples has an appropriate lower value.

In Figures 3.8, 3.9, 3.10 and 3.11, the detected planes are depicted in different colors. A critical point in images with a highly cluttered background is the inlier threshold. Especially interest points of the CD's on the table are often clustered with parts of the CD cover. In Figures 3.8 and 3.9, they are correctly separated, but interest points on the CD's lying on the magazine are clustered together with the magazine. The inlier threshold is also responsible for the approximation of the cylinder with piecewise planar surfaces in Figure 3.9(c). If the inlier threshold were lower and if the cylindrical object were less textured, the approximation would be less accurate.

Figure 3.10 and Table 3.6 show results of the Merton college and the Wadham college from the Oxford data set. In these images, the camera motion between the frames is much larger than it is in our data set. Furthermore the size of the images is larger (1024x768). In general this leads to a higher processing time. While our methods converge after $15s...20s$, RANSAC and J-LINKAGE need more than one minute. Furthermore, we tried some indoor examples with satisfying results. Figure 3.11(a) shows a rather crowded living room where planes of the pillow break in different pieces. In contrast Figure 3.11(b) shows a sparsely textured room where very small planes on the tile stove are merged to one bigger plane and because of



(a) (b) (c)

Figure 3.8: Results for the packaging data set 1 using *ItMoS*, the proposed Algorithm 2.

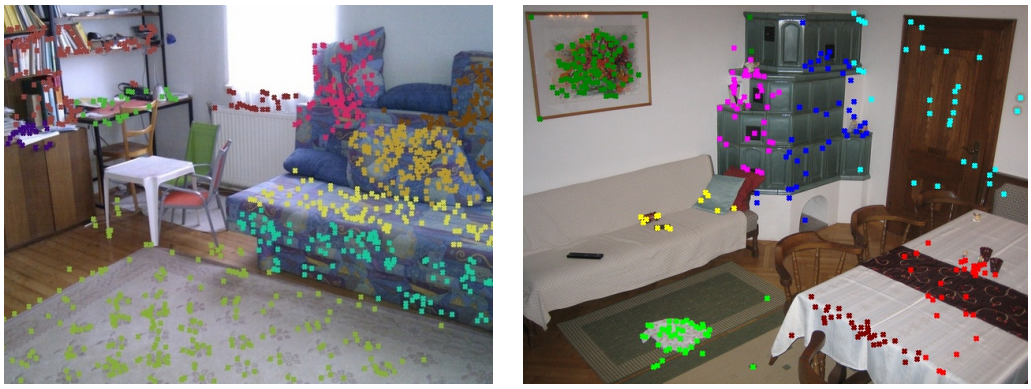
(a) (b) (c)

Figure 3.9: Results for the packaging data set 2 using *ItMoS*, the proposed Algorithm 2.

(a) MertonCollege1 (b) WadhamCollege

Figure 3.10: Examples of the Oxford Visual Geometry data set using *ItMoS*.

low texture, the couch is hardly visible for the system.



(a) (b)

Figure 3.11: Indoor scenes using *ItMoS*.

3.6 Discussion

This chapter presented a method for modelling a scene in terms of piecewise planar surfaces which are represented by the 2D projective transformation homography. For this, we track interest points in image pairs and use randomly selected points to generate plane hypotheses.

To select the subset of planes that best explains the image pair we reformulated the model selection based on Minimum Description Length (MDL) proposed by Leonardis et al. [LGB95] and lateron by Leibe [LLS08]. Finding the optimal set leads to a Quadratic Boolean Problem (QBP). The time to solve the QBP grows exponentially with the number of hypotheses. The goal to achieve real time performance makes it necessary to adopt an approximate solution. We have shown that for this formulation, a greedy approximation gives nearly as good results as a brute force computed exact solution. But still, the problem space to compute the interaction matrix for the plane hypotheses is of size $O(n^2)$ and hence, we propose to embed model selection in an incremental algorithm to keep the interaction matrix of the QBP as small as possible.

Nevertheless, the piecewise planar scene model can only be as good as the generated plane hypotheses. Besides a uniformly distributed sampling, we explored two options to increase the probability of choosing point pairs which are located on the same plane. Experiments have shown that the selection of near adjacent points to the first sampled point, using the Euclidean distance, gives nearly as good results as the selection using a sorted list depending on the motion vector of each point pair. It seems to be most important to keep the search space as small as possible. Our incremental model selection perfectly supports this because it is possible to distribute samples in unexplained regions.

There are two more details which dramatically improve the performance. First, it is important to provide a small number of good hypotheses. Superior performance is achieved with a connected component based pre-filter. This filter is able to select

the important 3% of hypotheses which are then passed on to the least-squares refinement and to the model selection. Secondly, the performance is improved if three points are sampled first, a six parameter affine transformation is used for the connected component analysis and after that the full 8 parameter least-squares homography is computed.

Up to now we did not aim at providing a pixel-wise segmentation, but it seems to be feasible to extend the approach with a post-processing to get a dense piecewise planar object model. One possibility would be to introduce a multi-label segmentation using a Markov Random Field (MRF) optimization and graph-cuts, e.g., using the method proposed by Sudipta et al. [SSS09] and Micusik et al. [MK09].

Chapter 4

Coherent Planes in Image Sequences

Typical vision systems integrate low-level visual cues in a hierarchical fashion and extract relevant output from this bottom-up processing. More recent approaches try to establish feedback loops and combine different vision methods at different levels, but these methods also reach their limitations if dynamical scenes get crowded up and objects get partially or even totally occluded. This chapter explores how to benefit from long image sequences. In this context, it is an excursus to modelling objects in 2D without a reconstruction of the Euclidean 3D structure. Instead, we try to handle the inaccuracy of the object model and the uncertainty of the vision component by means of reasoning over the image sequence. Planar patches from the previous chapter can directly be used for reasoning in long image sequences. But to show the benefit of our reasoning approach, we use a simpler motion clustering of interest points for segmentation and a star-shaped representation for recognition.

The work is inspired by findings of Gredebäck [Gre04] and Spelke [SvH01], who have shown that even infants at the age of 4 months build a spatio-temporal representation of objects and accurately predict the reappearance after full occlusion. To incorporate such ability, bottom-up visual processing is fused with top-down reasoning and inference to keep track of occluded objects and to learn appearances of objects that continuously change due to rotation or lighting.

We investigate two methods, namely **reasoning based on an evaluation function for an object hypotheses graph** and **tight coupling of segmentation, detection and tracking using model selection**. The first one builds on a reasoning component originally developed by Antenreiter et al. [AA06] for template based object tracking. The approach reasons about occlusion and hiding events and maintains a hypotheses graph that is updated according to the visual input. In case of a plausible hypothesis, a learning event is triggered and the vision component updates the interest point based object model. If the object is moving, interest points adjacent to the assumed object boundary are tested for consistent

motion. Furthermore, an interest point statistic is maintained that allows to delete interest points with low information content. Hence, the approach is able to keep an object model smart and manageable. In [PAAV09] and [APVA09], we have shown the increase of performance of an already very robust interest point based object detector in case our reasoning component keeps track of the objects.

Secondly, in Section 4.4 we test a tight coupling of interest point based object segmentation, detection and tracking within a model selection framework [PZV09]. This approach adds hypotheses depending on simple motion segmentation and weights the results with priors from the last frames. Additionally, direct occlusion detection based on interest point statistics is proposed, which triggers hiding events.

We test the methods with a scenario where a human moves different objects which interact several times, i.e., get occluded and reappear again. The goal is that the system tracks the objects even under full occlusion while enhancing the object model with never seen interest points and removing unreliable interest points.

After a review of the related work, we describe the overall approach (Section 4.2) and the common parts of both methods, namely the object model (Section 4.2.1) and the object detection (Section 4.2.2). Subsequently, each reasoning approach is described in detail in Section 4.3 and Section 4.4. Finally, results are given in Section 4.5.

4.1 State of the Art

Approaches can be split into occlusion reasoning systems for tracking and segmenting objects, mostly for traffic scenes or for following persons.

Elgammal and Davis [ED01] use a probabilistic framework for segmenting people under occlusion. Their system operates on a pixel level, whereas our system performs the occlusion reasoning on a more abstract object level.

Huang and Essa [HE05] present an approach for tracking a varying number of objects through temporally and spatially significant occlusions. The method is built on the idea of object permanence. The system can track objects in presence of long periods of full occlusions. They assume that a simple color model is sufficient to describe each object in a video sequence. Therefore they do not have to update their object models.

The approaches in [BT98] and [MJD+00] use image regions for occlusion reasoning. A region may consist of one or more objects, the relative depth between objects is not considered. If occlusions occur, the system merges the affected regions into a new region. On the other hand, a region is split, if the system is able to discriminate objects within this region. Thus, these approaches handle occlusion not at the object level.

Bennett et al. [BMCH04] enhance tracking results of moving objects by reasoning about spatio-temporal constraints. The reasoning engine resolves errors, ambiguities, and occlusions to produce a most likely hypothesis, which is consis-

tent with global spatio-temporal continuity constraints. However, the whole system performs only bottom-up processing.

One way to incorporate knowledge into vision systems is to use a knowledge-based approach, e.g. [MH90] for an aerial image understanding system, and [CPM00] for traffic monitoring applications. Matsuyama and Hwang [MH90] identified two types of knowledge in an image understanding system, namely, knowledge about objects and about analysis tools, and they built a modular framework which integrates top-down and bottom-up reasoning. The system extracts various scene descriptions, and an evaluation function selects the most complex scene description from the database. The evaluation function is simple and trusts the low-level vision output more than the reasoning output. While reasoning in [CPM00] is based on image regions using trajectories and velocity information, our reasoning is based on more abstract object behavior to achieve a consistent scene interpretation even if objects are totally occluded.

4.2 Approach

Computer vision deals with very noisy sensor data and a huge amount of data. Thus, robust object detection in single images is still challenging. Reasoning in image sequences addresses this in two ways: First, with the prior knowledge from previous images the location of objects can be predicted, which accumulates to higher certainty of object detection. The second advantage when dealing with image sequences is the reduction of the search space. If we keep track of objects, we do not need to re-detect them again. Even if objects get totally occluded, the location of the re-appearance can be predicted accurately around the occluder. In the following sections, we describe the overall approach and the common components of both reasoning methods proposed.

Both approaches consist of three main parts (Figure 4.1, right): the object detector, the reasoning component, and the knowledge-base used by the reasoning component. The reasoning component maintains a graph of hypotheses (Figure 4.1, left). Each hypothesis describes the content of a particular frame of the video, and it is linked to plausible hypotheses for the previous as well as for the next frame. A hypothesis is an assumption about the states of all relevant objects. It is calculated from the vision inputs for the current frame, from one of the hypotheses for the previous frame, and from the rules in the knowledge-base. Our current knowledge base includes the rules about the appearance of interacting physical objects. Examples for such rules are given in Section 4.3. Communication between the object detector and the reasoning component is not just bottom-up, but also includes top-down instructions from the reasoning component to the object detector, such as a request to detect a specific object at a certain location, or the command to update the object model with interest points at a specified position. The following list describes the main steps:

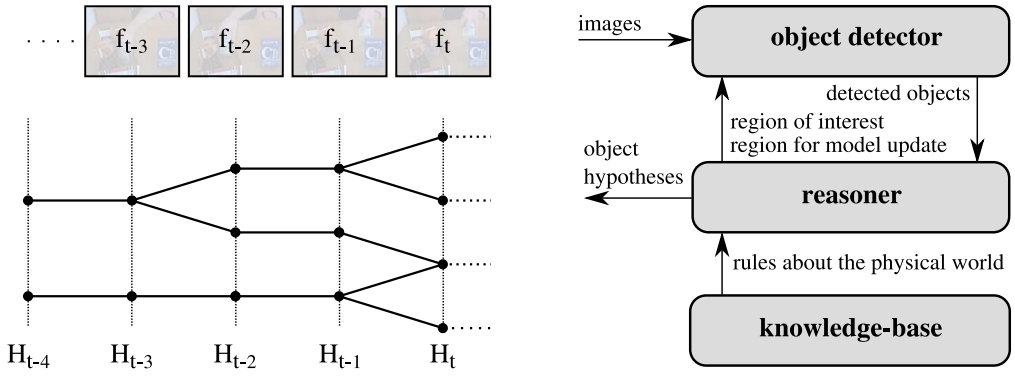


Figure 4.1: System overview, including the hypotheses graph (left), the structure of the system and the communication between components (right).

1. Top-down processing: Feed information from the hypotheses about the previous frame to the vision components.

2. Bottom-up processing: Create new *pre-hypotheses* from the output of the vision components. These pre-hypotheses give possible object positions and the confidence of the vision components for these positions.

3. Reasoning: Apply the *construction rules* of the knowledge base to the pre-hypotheses. The construction rules construct all plausible hypotheses.

4. Hypotheses selection: Prune the hypotheses graph from unlikely hypotheses. This pruning is based on an evaluation function for sequences of hypotheses corresponding to the sequence of frames.

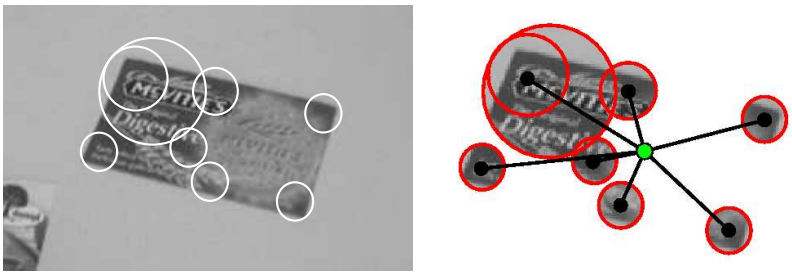
5. Object model update: Provide the boundary of the visible object area to update the object model if the actual hypothesis is reliable.

The next sections in detail describe the involved components and the main steps, starting with the object detector.

4.2.1 Object Representation

In Section 4.2, we described our reasoning approach which is designed to process image sequences. The first step is to learn the initial object models. To this end, the user has to mark the object boundaries in the first frame. In case of an existing initial object model or in case we use some kind of segmentation (e.g. plane detection proposed in Chapter 3 or the motion segmentation from the tested implementation described in Section 4.4.1), this interaction can be omitted.

As object representation, we use a part based model proposed by Leibe et al. [LLS08] for the task of object class recognition. The idea is to build up a vocabulary (in the following termed a *codebook*) of interest points and to compose



(a) (b)

Figure 4.2: Figure 4.2(a) shows detected interest points and in Figure 4.2(b) the star-shaped object representation is sketched.

the geometric structure out of this vocabulary. The initial set of codebook entries is generated from the first image. As proposed by Lowe [Low04], the Difference-of-Gaussian (DoG) operator and SIFT descriptor are used for detection and description of interest points. These interest points are clustered and each codebook entry represents a cluster of interest points with a similar SIFT descriptor. The codebook is then used to build up a structural model of the object. Therefore detected interest points on the object are matched with the codebook and the relative locations of interest points with respect to the center of the object are assigned to codebook entries.

In summary, each codebook entry represents a part of the object and stores the possible locations where this part can be found relative to the object center. The result is a star-shaped structural model of the object that is shown in Figure 4.2.

4.2.2 Object Detection

As described in Section 4.2.1 for generating the codebook, interest points are detected using the DoG-operator and the SIFT-descriptor for object recognition. The detected interest points are matched with the codebook, and activated codebook entries vote for an object center.

Consistent votes are accumulated in the Hough accumulator array. We use a four dimensional space where occurrences of activated codebook entries vote for an object location $\mathbf{x}_v = (x_v, y_v)$, a scale s_v and an object model m:

$$s_v = \frac{s_i}{s_{occ}}, \tag{4.1}$$

$$\mathbf{x}_v = \mathbf{R}\mathbf{x}_{occ}s_v + \mathbf{x}_i. \tag{4.2}$$

In Equation 4.1 s_i is the scale of the detected interest point in the current image and s_{occ} denotes respectively the scale of the occurrence in the learning image. In Equation 4.2 $\mathbf{x}_i$ is the location of the detected interest point in the current image, $\mathbf{x}_{occ}$ denotes the location of the object center with respect to an occurrence

of the model and **R** stands for the matrix that describes the rotation (in image coordinates) from model to image orientation of the interest point. Likewise, as proposed by Leibe et al. [LLS08] we weight votes depending on the cluster size of the according codebook entries.

Once all matched interest points have voted, the Hough accumulator array is used to find the most promising object hypotheses. The votes in each Hough bin i are summed up and – starting with the best hypothesis, i.e., the maximal bin – the object location is refined. This is done in a mean shift like procedure, for which the neighboring bins are examined for contributing votes. This handles the typical boundary effect of Hough voting schemas.

The result of the mean shift refinement is a cluster of interest points that consistently vote for an object location. This cluster is used to compute an affine homography H_{aff}, for which we use a Least Median of Squares implementation, publicly available at FORTH [Lou06].

We define the object confidence

$$detected(o|m, f_t) = \frac{n_{matched}}{n_{detected}} \tag{4.3}$$

of an object o for a given frame f_t and an object model m as the ratio between the matched interest points $n_{matched}$ and the number of the detected interest points $n_{detected}$ located within the boundary projected to the current frame.

4.3 Reasoning based on an Evaluation Function for the Object Hypotheses Graph

The hypotheses graph described in Section 4.2 stores different explanations for the scene, i.e., object hypotheses and the corresponding confidence values for each image of a video. The challenge for the reasoning component is to consider the possible object locations detected in the image sequence and to select the best hypothesis available up to now.

The main idea is to split a hypothesis into two or more plausible hypotheses, if the correct hypothesis cannot be inferred. What can be inferred mainly depends on what can be observed. In the setting for the following experiments, we use one static camera. Thus the distance of objects from the camera cannot be observed. If two boundaries intersect, it cannot be decided ad hoc which object is in front of the other object. Hence, the current hypothesis is split in two and each possibility is represented by one of the hypotheses. Examples for hypotheses construction rules:

1. If the boundaries of two objects intersect, then split the hypothesis into two. One hypothesis states that the first object is in front, the other hypothesis states that the second object is in front.

2. If an object is partially occluded by one other object (occluder), then we generate two hypotheses. One hypothesis states that the object does not move and reappears at the old position. The other hypothesis states that the occluded object can reappear along the trajectory of the occluder.

3. If the object detector returns two or more possible object positions with similar confidence for an object, then generate one hypothesis for each plausible object position.

After running the object detector and constructing the hypotheses graph, an evaluation criterion is used to calculate the most likely hypothesis path which explains the video sequence. The following summarizes the reasoning approach originally developed by Antenreiter et al. [AA06] for template based object tracking. The evaluation function $Q(\mathcal{H}|\mathcal{F})$ evaluates a hypothesis path $\mathcal{H} = \langle h_1, \ldots, h_m \rangle$ from the root hypothesis at frame 1 to a hypothesis for the current frame where $\mathcal{F} = \langle f_1, \ldots f_m \rangle$ is the corresponding sequence of frames. It is a quality measure of a hypothesis path compared to other paths. We sum the quality measures q for each hypothesis on the path to calculate the overall evaluation measure Q,

$$Q(\mathcal{H}|\mathcal{F}) = \frac{1}{m} \sum_{i=1}^{m} q(h_i|f_i). \tag{4.4}$$

The quality measure for a hypothesis of frame f_t then is the sum of all normalized plausibility values,

$$q(h_i|f_t) = \frac{1}{\#\{o : o \in h_i\}} \sum_{o \in h_i} \frac{p(o|h_i, f_t)}{\max_{h \in H_t} p(o|h, f_t)} \tag{4.5}$$

where $H_t = \{h : h \text{ is a hypothesis for frame } f_t\}$ is the set of all hypotheses for frame f_t. The plausibility value $p(o|h_i, f_t)$ is a mapping of the detector state and the current state of the hypothesis h_i.

The inconsistency function ι combines the confidence value $detected(o|h_i, f_t)$ of the detector component with the relative occluded area $occludedarea(o|h_i)$, and should detect inconsistency between these values:

$$\iota(o|h_i, f_t) \quad = \quad \left| -\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\, detected(o|h_i, f_t) + \frac{\sqrt{2}}{2}\, occludedarea(o|h_i) \right| \tag{4.6}$$

Equation 4.6 describes the normal distance from a line defined by $(1, 0)$ and $(0, 1)$ and thus the values are between 0 and $\sqrt{(}2)/2$. The function *occludedarea* returns the normalized area of the occluded object area, depending on the state of the given hypothesis, ranging from 0 to 1. We assume that if the detector confidence is small, the occluded object area should be large. The inconsistency ι is small if the low-level vision confidence is small and the occluded object area is large, or vice versa.

The plausibility function p maps the inconsistency value to a value between one and zero. For objects which are not totally occluded, we define

$$p(o|h_i, f_t) := -\frac{2}{\sqrt{2}}\,\iota(o|h_i, f_t) + 1. \tag{4.7}$$

If an object is totally occluded, a different definition of plausibility is needed. It would be misleading if a plausibility value of 1 were assigned to totally occluded objects, since a visible object will not typically receive a plausibility value of 1 due to noise, illumination changes, etc. Then a hypothesis with a totally occluded object would have higher plausibility than a hypothesis where the same object is partially visible. Therefore, we bound the plausibility of a total occluded object by its maximal plausibility when it is visible:

$$p(o|h_i, f_t) = \max_{\substack{h \in H_t \\ o \text{ is visible}|h}} p(o|h, f_t) \tag{4.8}$$

If there does not exist a hypothesis $h \in H_t$ so that object o is visible, then set $p(o|h_i, f_t) = 1.0$. This is the case where all hypotheses assume that the object o is totally occluded.

4.4 Tightly Coupling Segmentation, Detection and Tracking in 2D using Model Selection

The approach described in the preceding sections provides a general framework to integrate object detection and reasoning with the goal to improve the availability of object locations even if visual detection fails. To create the initial object model, manual interaction is necessary. A necessary basic ability for such an artificial cognitive system (be it an ambient intelligent system or an autonomous robot) is to focus on the foreground and perceive objects in contrast to the background. In the following, we introduce a simple clustering depending on common motion of interest points and a model selection framework to learn 2D appearances of objects and to keep track of them. Hence, in comparison to the system described above, we now formalize reasoning for a model selection framework and in addition. the initialization is not done manually anymore, but by motion segmentation. Thus the system from Figure 4.1 is enhanced with a motion segmentation which is shown in Figure 4.3.

4.4.1 Motion Segmentation

The whole system is triggered by the motion segmentation component. We do not rely on a perfect segmentation of moving objects, but rather take care to achieve

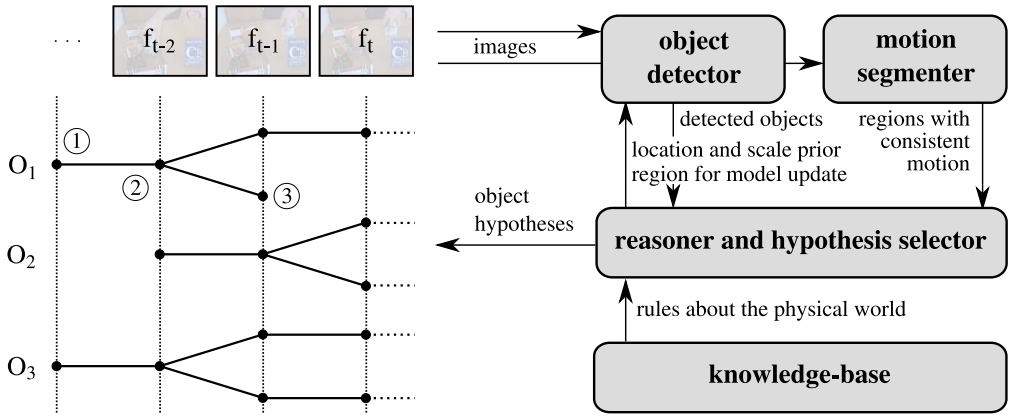


Figure 4.3: System overview, including the hypothesis graph (left), the structure of the system and the communication between the different components (right) including motion segmentation for automatic initialization.

robustness later due to the reasoning component described in Section 4.4.5. Consequently, we just use a fast clustering of interest points based on their affine motion. Our approach is inspired by the work of Pundlik et al. [PB08], who presented a real-time incremental approach to motion segmentation, operating on sparse feature points. In contrast to Pundlik et al., who randomly select interest points and uses an incremental growing algorithm, we use a 2-dimensional histogram of the length of the motion vectors and the motion direction to obtain good initial preclusters. Then a splitting algorithm and an outlier detection step follow and the clusters are merged based on similar affine motion.

In detail: The first step is to examine the 2D-motion histogram. For this, we search for all local maxima, that is we look for histogram bins which are surrounded by bins with a lower number of entries. Starting from the local maxima, all neighboring bins are clustered until a saddle bin is found. The result can be seen in Figure 4.4(a). The next step is to split large clusters in case of intersecting convex hulls of other clusters. For this purpose, we use a Delaunay tree to create a location neighborhood graph of all interest points detected in the image. The splitting criterion prohibits intersections of two clusters and thus substitutes a cluster with two new ones if they have no connection within the Delaunay tree. After an affine outlier detection using a Least Median of Squares implementation, publicly available at FORTH [Lou06] (see Figure 4.4(b)), the clusters are again merged if the affine error is lower than the maximal error would be if they stayed separated. Thus the merging criterion results in

$$C_m = C_i \cup C_j \text{ for } e_m < \max(e_i, e_j) \tag{4.9}$$

In Equation 4.9, C_i and C_j are two clusters, which are tested for similar affine motion and C_m denotes the merged cluster. e stands for the affine errors of the clusters. Additionally, we can adjust a chaining parameter to cluster only features

(a) (b) (c)

Figure 4.4: Grouped motion vectors of interest points are shown with identical colors; different colors mean different clusters. Each subfigure depicts the result of different processing steps. Figure 4.4(a) shows the result of grouping according to similar length and direction of the motion vectors using a 2-dimensional histogram. Figure 4.4(b) pictures the result after examination of the neighboring motion vectors using a Delaunay triangulation and after affine outlier detection. In Figure 4.4(c) the result of Figure 4.4(b) is used to initialize a merging algorithm which combines clusters depending on their affine motion.

which are tracked for more than two frames and are thus considered as more stable.

Figure 4.4 shows the results of all three main steps. It can be seen that the three outliers on the hand are filtered out as well as the mismatch on the keypad of the telephone. The final motion clusters are handed over to the reasoning component which initializes a new object hypothesis or adds the features to an existing object. This is explained in detail in Section 4.4.5.

4.4.2 Confidence Values for the Model Selection Framework

Finally, the object detector described in Section 4.2.2 incrementally consumes supporting interest points of neighboring histogram bins with a mean shift algorithm. The result of the mean shift refinement is a cluster of interest points that consistently vote for an object location. Subsequently, an affine homography H_{aff} of the cluster is computed, which is used to project the model boundary to the current frame. The projected boundary is on the one hand used for visualization, but also for interest point statistics and for computation of the confidence value

$$c(o|m, f_t) = -\kappa_1 + (1 - \kappa_2) \cdot \frac{n_{matched}}{n_{detected}} + \kappa_2 \cdot \frac{s_{matched}}{n_{detected}} \tag{4.10}$$

of an object o for a given frame f_t and an object model m. $n_{matched}$ is the number of matched interest points and $n_{detected}$ is the number of the detected interest points located within the boundary projected to the current frame. $s_{matched}$ is the sum of

Figure 4.5: Probability map for one specific object computed using the predicted location and the result of the occlusion analysis.

the weights

$$w = p_m p_{occ} = \frac{1}{n_m \cdot n_{occ}} \tag{4.11}$$

of all matched interest points with p_m and p_{occ} denoting the probabilities of the match and the occurrence in the model, respectively. n_{occ} is the number of occurrences of the specific object model of the activated codebook entry and n_m is the number of activated entries of the interest point. κ_1 and κ_2 are two constants which weight the different factors.

For tracking of the objects, we use a constant velocity assumption. Therefore, the affine homography H_{inc} between two frames is computed for each object. Then the assumed location and scale is computed for objects of the current frame and the confidence value is extended to

$$c_{track}(o|m, f_t) = c(o|m, f_t) + \kappa_3 \cdot \log\, p(o_{f_t}|o_{f_{t-x}}) + \kappa_4 \cdot \log\, s(o_{f_t}|o_{f_{t-x}}) \tag{4.12}$$

where $p(o_{f_t}|o_{f_{t-x}})$ and $s(o_{f_t}|o_{f_{t-x}})$ stand for the location and the scale prior. The log-probabilities result in negative values. This is reflected in the values of κ_3 and κ_4 which are used to weight the priors. . We model the priors using a Gaussian around the predicted location and the last scale. In case of occlusion, the location prior is extended and surrounds the whole boundary of the occluder. Thus reappearing objects are accepted near the occluder and at the last seen location (see Figure 4.5). Then the objects are sorted according to the tracking confidence value and added to the hypothesis tree.

4.4.3 Maintenance of the Object Models

The next step is to update existing object hypotheses. For this, we compute an overlap matrix which describes the support of segmented regions and detected object hypotheses. We define the support

$$support_{i,j} = \frac{A_{seg} \cap A_{det}}{A_{seg} \cup A_{det}}. \tag{4.13}$$

where the support of a segmented region r_{seg} for a detected object hypothesis o_{det} is the ratio of the intersection and the union of the segmented area A_{seg} and the area of the detected object hypothesis A_{det}. For our experiments we used a *winner takes all* updating strategy, which means that the detection result with the highest tracking confidence value is updated if the support is larger than a threshold t_{update}. Additionally we use a second threshold t_{new} to create a new object hypothesis. If a segmented region does not support any detection result more than t_{new}, a new hypothesis is created. Depending on the detection results and these two thresholds an over-complete set of object hypotheses is created, from which hypotheses explaining the scene in a consistent way are selected.

4.4.4 Direct Occlusion Detection

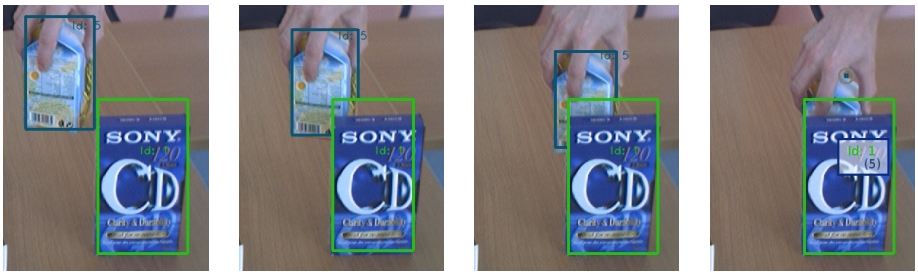
The reasoner plays the central role in our framework. It predicts object locations both for the case of tracking and for the case of total occlusion. It creates new object models or updates existing models depending on coherent segmentation and detection results and it selects objects from an over-complete set to obtain a consistent scene interpretation. This section describes the occlusion analysis, which forms one of the important cues of the reasoning and selection component described in the next section.

We aim to get a consistent interpretation of an image sequence. Thus it is necessary to predict objects even if they are totally occluded. Therefore we, developed an event based occlusion analysis schema. If an object gets lost, the past, the current and the predicted object locations in the future are examined for possible occluders. Hence, for each location the overlap of the projected object boundary with the other visible object hypotheses is computed and if they overlap the visible object gets an occlusion vote. The voting is performed for all past and future object locations which are within a maximum distance of half the object size. After the visible objects have accumulated the votes, the ID of the occluded object is assigned to the visible one with the most votes and to all other ones which got more than 80% of the maximum. It turned out that using this voting schema is more reliable than only looking at the position of disappearance, because in case of partial occlusion, our model updating algorithm tends to shrink the estimated object boundary to the visible part of the object.

Figure 4.6 shows an occlusion event, the correct depth ordering which is estimated from the confidence values and the link of an occluded object to the occluder (indicated by an object ID within brackets).

4.4.5 Model Selection for Reasoning

As an alternative to the reasoning component described in Section 4.3, we propose a tight coupling of segmentation, detection and tracking within a model selection framework. The motivation for model selection has already been introduced in



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

Figure 4.6: Occlusion event including correct depth ordering and an occluded object linked to the occluder

Chapter 3 for plane detection. The idea is that the same data points cannot explain more than one object and that the models cannot be fitted sequentially. Thus, an over-complete set of hypotheses is generated and the best subset is chosen.

As for plane detection, the data set consists of the interest points and each interest point can only be assigned to one object model. Hence, overlapping models compete for interest points. This competition is expressed by the interaction costs q_{ij}, while q_{ii} represents the merit term of an object hypothesis. Finding the optimal set of models leads to a Quadratic Boolean Problem (QBP)

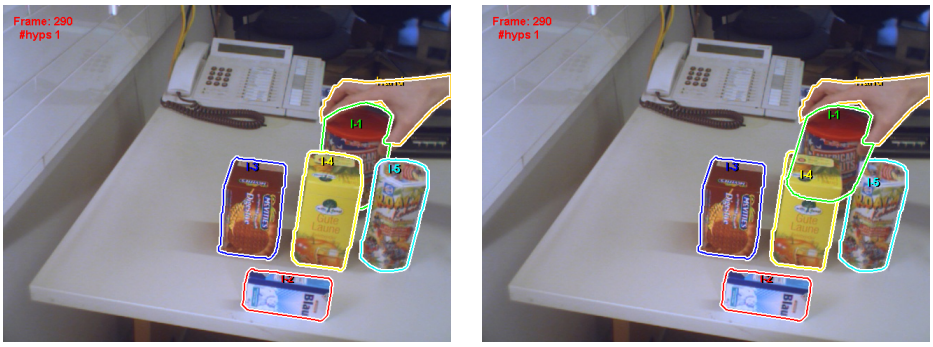
$$\max_{n} \mathbf{n}^T Q \mathbf{n} \ , \ Q = \begin{bmatrix} q_{11} & \cdots & q_{1N} \\ \vdots & \ddots & \vdots \\ q_{N1} & \cdots & q_{NN} \end{bmatrix} \tag{4.14}$$

where $\mathbf{n}$= $[n_1, n_2, \cdots, n_N]$ stands for the indicator vector with $n_i = 1$ if an object hypothesis is selected and $n_i = 0$ otherwise. Q is the interaction matrix with the diagonal elements $q_{ii} = c_{track}(o|m, f_t)$ and the off-diagonal elements

$$q_{ij} = -\frac{1}{n_{o,weak}} \cdot ((1 - \kappa_2) \cdot n_{overlap} + \kappa_2 \cdot s_{overlap}) \tag{4.15}$$

where $n_{o,weak}$ is the number of interest points within the projected boundary to the current frame of the weaker hypothesis, i.e. with the lower confidence value, $n_{overlap}$ stands for the number of interest points which are shared by both objects and $s_{overlap}$ is the sum of the weights of all shared interest points (cf. Equation 4.11). Equivalent to plane detection, the optimization problem described in Equation 4.14 is solved with a greedy approximation.

Hence, object hypotheses are generated with the results from object segmentation, detection and tracking. In addition, the QBP-framework selects a set of hypotheses that best explains the current frame.



(a) (b)

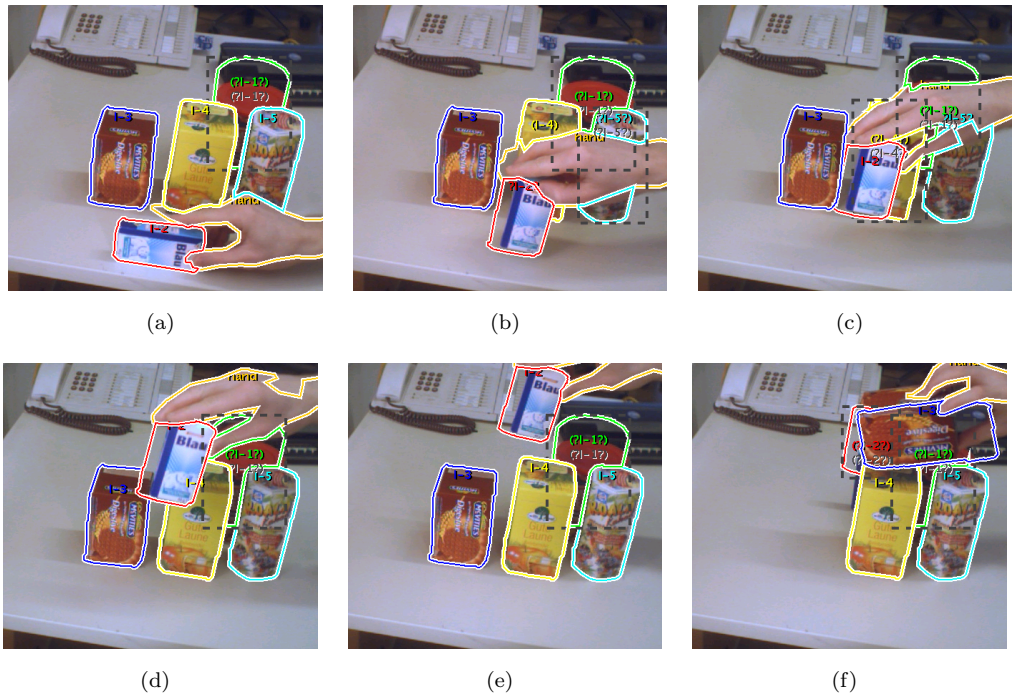
Figure 4.7: Two possible interpretations of an observed scene are shown. Figure 4.7(a) shows the best interpretation of the frame 290, which is the correct one. The confidence value of this hypothesis path is 0.9669, compared to the other interpretation in Figure 4.7(b), which is only 0.8913. The system explained the occlusion event correctly.

4.5 Tests and Evaluation

Both methods, the system including the reasoning component which evaluates the whole image sequence up to the current timestamp and the QBP-framework for scene interpretation, have been tested with scenarios where a human moves different objects which interact several times, i.e., get occluded and reappear again.

4.5.1 Comparison with/without the Reasoning Component

For a comparison with and without the reasoning component described in Section 4.3, we processed four video sequences. In the first video sequence we arranged the objects within three layers. A person moved the objects between the layers and the objects got partially or totally occluded. The system has to interpret the sequence correctly, so that it can update the object models accordingly. This is important, because wrong model updates can accumulate to a wrong model, which can lead to false detection results. On the other hand – without updating the object models –, the system will not be able to reliably detect the objects. Small objects are usually not reliably detected during motion and rotation. In our first test sequence, the object with id 'I-2' is an example of such a small object. Figure 4.7 shows an occlusion event and two possible interpretations from the observed detection results. In Figure 4.7(a) the system concludes that the object with id 'I-1' is occluded by the two objects 'I-4' and 'I-5'. Therefore it can accordingly update the model for 'I-1'. The system draws the visible boundary for the object 'I-1'. In Figure 4.7(b) the system tries to explain the scene with another object ordering. This leads to a different and wrong object model for object 'I-1'. The newly learnt model for object 'I-1' gives a good detection result, because the system



(a) (b) (c)

(d) (e) (f)

Figure 4.8: The first box sequence showing the various occlusion events (Frames: 420, 495, 545, 570, 595, and 845). The relative depth between the objects is correct in every frame.

learnt additional keypoints from the surrounding objects, but the overall hypothesis cannot explain the scene properly. The contradiction is that all detectors give good detection results even under occlusions. The evaluation function is able to detect such a contradiction. The corresponding hypothesis of Figure 4.7(a) has a confidence value of 0.9669, which is higher than the confidence value 0.8913 of the hypothesis from Figure 4.7(b). In Figure 4.8 we show a more complex arrangement of objects from the first sequence. Every object has an id label and a boundary. The system draws the estimated visible boundary. Therefore, the images show the estimated relative depth between objects. The id label (within the boundary) is bracketed if the reasoning does not use the detection result. Question marks indicate that the reasoning has assumed a detection failure. A dashed box around the last known object position is drawn if the reasoning has not enough evidence for an exact object position. In frame 420 (Figure 4.8(a)), the object with id 'I-1' is placed between the objects 'I-4' and 'I-5'. The relative depth is correctly estimated. The next frame (Figure 4.8(b)) shows a nearly total occlusion of object 'I-1', 'I-4', and 'I-5'. In Figure 4.8(c) detection results are used for objects 'I-2' and 'I-3', but not for object 'I-4' which is totally occluded by the hand, and not for object 'I-2'. In the frames 570 and 595 (Figures 4.8(d) and 4.8(e)), there is less occlusion and the reasoning uses the detection results (bottom-up processing). In the last

frame the object 'I-3' is rotated and placed on top of the object 'I-4'. The objects 'I-1' and 'I-2' are totally occluded by the other objects in the scene. In Table 4.1,

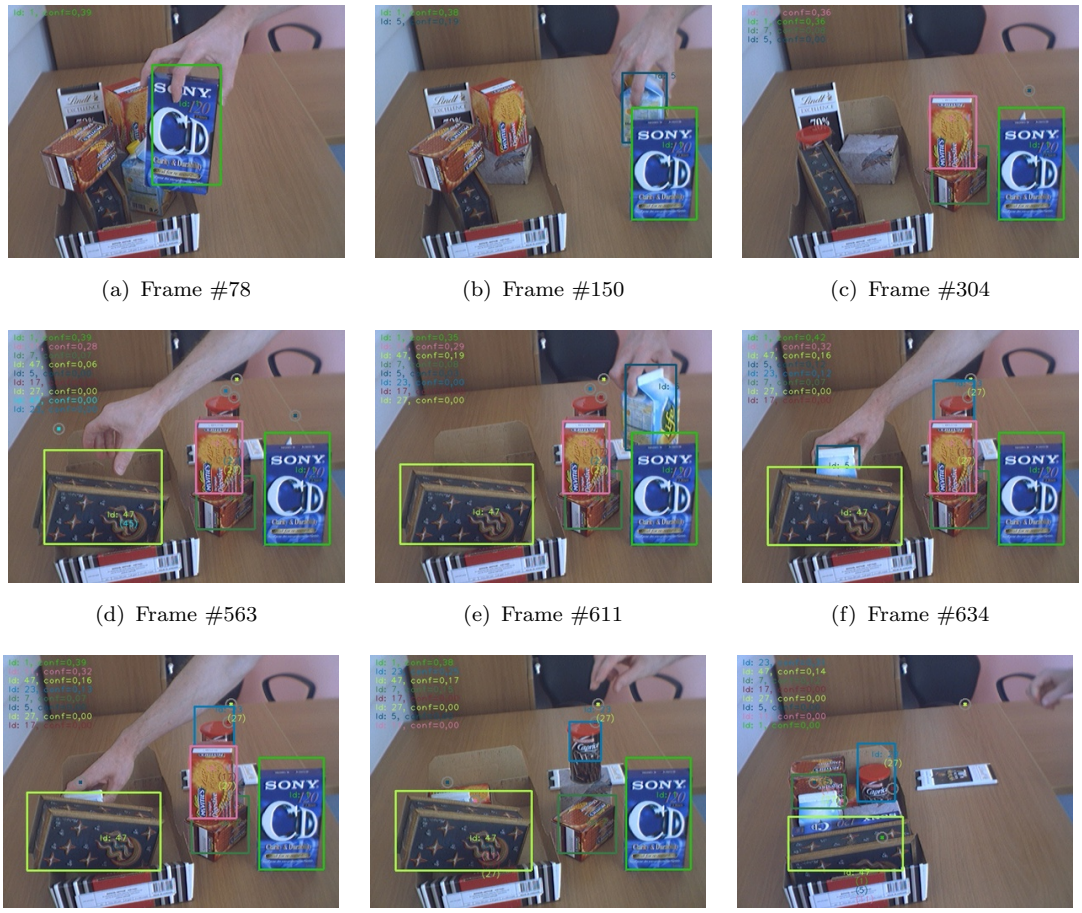
	static model		online learned model	
Video	detection rate	depth ordering	detection rate	depth ordering
box1	95.90%	88.86%	97.81%	85.05%
box2	74.31%	66.43%	98.71%	97.84%
box3	81.43%	74.80%	99.07%	91.24%
cap	32.86%	21.40%	99.95%	93.21%

Table 4.1: Comparison of using a static model versus a dynamic model.

we show a comparison with a system which learns the object model from the first image (static model). The static model is used with the reasoning component to explain the scene. The first column shows the detection rate for all visible objects. The second column indicates how well the reasoning can infer the correct depth ordering. As can be seen from Table 4.1, our online model improves the detection results compared to the static model. Additionally, the reasoning component can prevent the system from learning a wrong model even under occlusion events.

4.5.2 Tests with the Model Selection Framework for Reasoning

We processed six video sequences to test reasoning with the model selection framework described in Section 4.4. In the following, we present three sequences, which show the strengths as well as the weaknesses. The system has to detect object hypotheses because of consistent moving interest points, interpret the sequence correctly including hypotheses for totally occluded objects and build object models with all seen views. In our first video sequence, called *Sorting the Shopping Basket*, we arranged typical household articles in a box in a crowded manner. A person empties the box, resorts the articles and places them into the box again. The sequence shows many complex interactions and is taken at a low frame rate (objects move more than 40 pixels between two frames) to show that our system can handle motion blur and that it is not bound to a strict tracking assumption, but rather selects the best interpretation that is currently available. Figure 4.9 shows selected frames of the sequence. Currently available object models are depicted with bounding boxes and the corresponding IDs and confidence values are displayed in the upper left corner of each image. In Figure 4.9(a), the first object is grasped and because of the motion, an object hypothesis with ID 1 is generated. The next Figure 4.9(b) shows the second object (ID 5) which moves behind the first object. Correct occlusion assignment and the last detected location are indicated with the ID within brackets under the occluder ID and with a colored dot surrounded by a grey circle. During complex actions, sometimes "hallucinated"



(a) Frame #78 (b) Frame #150 (c) Frame #304

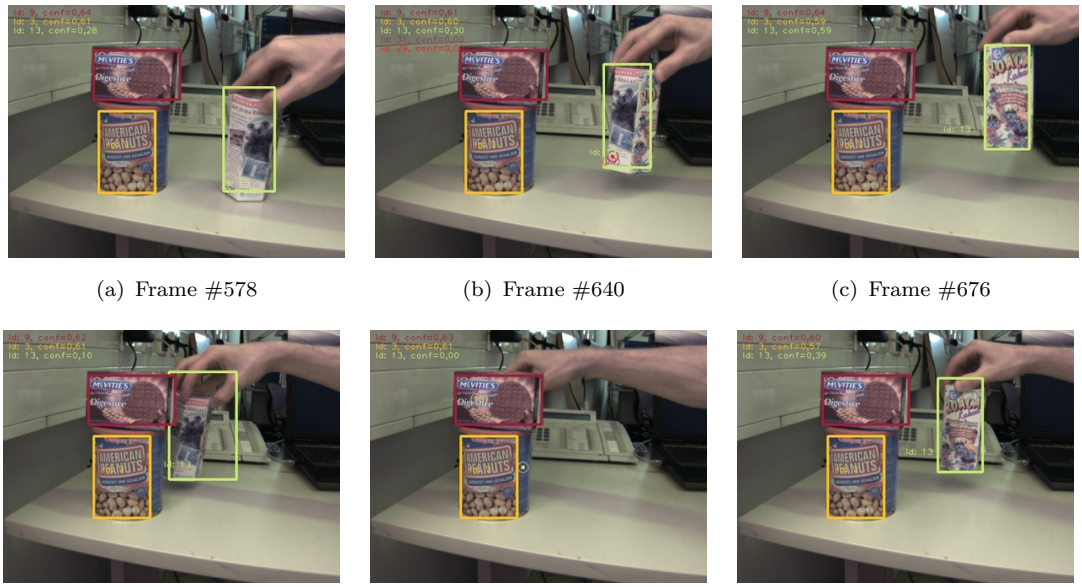
(d) Frame #563 (e) Frame #611 (f) Frame #634

(g) Frame #640 (h) Frame #707 (i) Frame #902

Figure 4.9: Selected frames of the video named *Sorting the Shopping Basket*, indicating the complex interactions. Bounding boxes of learned objects are shown with different colors and the corresponding IDs and confidence values of all currently available models are depicted in the upper left corner of each image. If an object is lost, the last position is depicted with a colored dot surrounded by a grey circle and the ID of the occluded object is displayed under the occluder ID within brackets.

object hypotheses are created (Figure 4.9(d) object ID 45) which are not confirmed and thus deleted in the following frames. In Figure 4.9(e), the object with ID 5 re-appears. In this frame, all eight correctly learned object models are listed in the upper left area of the image. After some interactions shown in Figure 4.9(f), 4.9(g) and 4.9(h), the sorted box with correct occlusion assignment is depicted in Figure 4.9(i). Only the chocolate bar (ID 27) is not recognized again, because of a too drastic change of the size and a too large rotation while it was occluded (i.e., no model was generated of this view before).

The second sequence depicted in Figure 4.10 contains three foreground objects. One of the objects (ID 13) is rotated to different views. Then this object moves

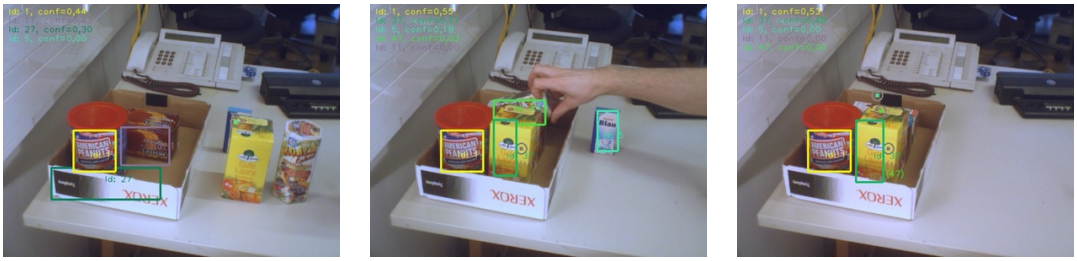


(a) Frame #578 (b) Frame #640 (c) Frame #676

(d) Frame #772 (e) Frame #810 (f) Frame #850

Figure 4.10: Part of a 900 frames long video which indicates the learning of an object model including the history of the object. The model of object 13 is learned while rotating to completely different views. Then it is moved behind object 3 and 9. During full occlusion, the object is rotated and appears again with a view learned at the beginning.

behind the other two and – triggered by the occlusion event – a model of all views, which have been shown before, is computed. During full occlusion, the object is rotated and re-appears with a view shown at the beginning. It is correctly recognized again in Frame #850 (Figure 4.10(f)).



(a) Frame #1820 (b) Frame #2175 (c) Frame #2460

Figure 4.11: Three images of a 2590 frames long video are depicted showing two possible errors.

In Figure 4.11, another sequence with household articles is shown. Two errors occurred despite the correctly learned object models. The first one is that the model of the xerox box (ID 27) has disappeared. This object hypothesis is not confirmed

often enough during tracking and thus it has been deleted due to our forgetting curve. The second error is that the occluded object with ID 5 is not linked to the occluder 37. Because of a rotation in depth during occlusion, the prediction was wrong and thus object 5 did not get in contact with the occluder 37.

4.6 Discussion

This chapter discussed modelling and detection of objects in image sequences using a simple two dimensional star-shaped object model. A reasoning system is proposed to handle the inaccuracy of the object model and to keep track of the objects even if they are fully occluded. For reasoning, two methods are tested. The first approach combines a decoupled reasoning component and the vision component with a general interface. The reasoning triggers detection and learning of a model and maintains a hypotheses graph based on an evaluation function. The second method is based on model selection to select hypotheses from an over-complete set, which currently best explain the image sequence. Hence, it tightly couples segmentation, detection and tracking.

We have shown that both methods are well suited to keep track of objects under full occlusion. Through maintaining different hypotheses it is possible to update the object model and to handle the drift typical for online update. Thus an image sequence with more than 2500 frames showing a large number of partially and totally occluded objects could be handled successfully.

However, two limitations arise from these methods. The evaluation of the hypotheses graph is NP-hard and thus early pruning is a crucial point to achieve real time performance. Another issue is that reasoning can handle the inaccuracy of the star-shaped model, but it is not possible to reconstruct the 3D Euclidean structure necessary for robotic applications. To solve this, the system could be adapted and a more accurate object model introduced, e.g., a model constructed from planar patches such as described in the previous chapter. Hence, the next chapter is concerned with the creation of 3D object models from planar patches.

Chapter 5

From Planes to Objects

In Chapters 3 and 4, we provided tools to detect meaningful structures – namely planar patches – in image pairs, which are then tracked and consistently interpreted during partial and total occlusions in long image sequences. Up to this point, planes are modelled with homographies which provide a lot of information necessary for robot manipulation. According to the requirements list in Chapter 1, Section 1.2, the piecewise planar object model already

- consists of meaningful parts
- forms clusters of parts according to to object hypotheses
- provides a description to recognize object instances
- enables object interaction
- contains elements to support a description for recognizing object classes and categories, and
- contains elements to support action selection and planning

Requirements that are not or only partially fulfilled – such as to recognize object classes, to detect affordances, to enable interaction and to accordingly do the required planning – highly depend on the 3D Euclidean structure. Hence, in this chapter we are concerned with the Euclidean reconstruction using the planar patches from Chapter 3. We further examine how to create reliable object hypotheses from image sequences.

In [PZV11b], we have shown that the planes modelled with homographies well support a subsequent robust Euclidean reconstruction. Hence, in the following chapter we investigate how to reconstruct the objects and how to acquire Euclidean models using image sequences. For this, we propose a Structure-from-Motion approach which focuses on already detected planes. This method is able to cope with multiple motions, and in combination with the pose provided by the robot kinematics, a full reconstruction without scale ambiguity is acquired.

5.1 State of the Art

The planes, represented by homographies, are the basic entities for 3D reconstruction and for merging/splitting to create the final object model. While the classical Structure-from-Motion problem of moving through a static scene is essentially solved in a coherent theory [FLP01, Har08] and several robust systems exist, in recent years, researchers have focused on dynamic scenes composed of rigidly moving objects. The solutions available so far can be broadly classified into algebraic methods, (e.g. [CK95, VM04, VH04]), which exploit algebraic constraints satisfied by all scene objects, even though they move relative to each other, and non-algebraic methods, (e.g. [Tor98, FZ00]), which essentially combine rigid SfM with segmentation.

Costeira et al. [CK95] propose an algebraic method. They developed a multibody factorization method to segment the motion and recover the shape of multiple moving objects from a set of features tracked in a sequence of images. They introduce a linear-algebraic construct of object shapes, called the shape interaction matrix. The entries of the matrix are directly computed from the feature trajectories without knowing their object identities. Then the transformation of the matrix into the canonical form results in a segmentation of the features as well as a recovery of the shape and motion of each object. Furthermore, it is shown that this approach works for different projection models, such as weak perspective (scaled orthography), paraperspective, or an affine camera. Vidal et al. [VM04] present a unified algebraic approach for 2D and 3D motion segmentation. The key contribution is to view the estimation of multiple motion models as the estimation of a single multibody motion model. Vidal et al. show that the image measurements can be fit with a real or complex polynomial function and that the parameters of the motion models can be obtained from the derivatives of the polynomial at the measurement. The approach is based on a polynomial differentiation instead of a polynomial factorization and thus, compared to former approaches, the efficiency, accuracy and robustness of the algorithm is improved. This approach studies the segmentation of multiple motions from two-view correspondences. In [VH04], Vidal et al. consider the problem of segmenting multiple rigid motions from point correspondences in multiple affine views. The problem is cast as a subspace clustering problem and thus the approach handles the whole spectrum of possible affine motions: from two dimensional and partially dependent to four-dimensional and fully independent. Point trajectories of all points are projected into a 5-dimensional space, and a PowerFactorization method is used to fill in missing data. Then GPCA is used to fit multiple linear subspaces representing independent motions. Thus, the approach achieves a misclassification error of less than 5% for sequences with up to 30% of missing data points.

The non-algebraic approach proposed by Torr [Tor98] uses mixture models in an optimal statistical framework to estimate the number and type of motion models and their parameters. For this, Torr developed a robust version of the AIC

information criterion, called GRIC, for robust model selection. This method simultaneously flushes outliers and selects the type of model that best fits the data. Fitzgibbon et al. [FZ00] extend the recovery of structure and motion to the case of image sequences with multiple unknown motions. The assumption that all cameras share the same unknown camera parameter allows Euclidean reconstruction in the multibody case, when it was underconstrained for a static scene. In [SSW08] Schindler et al. propose a framework for multibody-structure-and-motion based on model selection. First, in a recover-and-select procedure, a redundant set of scene motions is generated. Subsets of these redundant motion candidates are regarded as possible explanation of the tracked image features. In a second step, the most likely explanation is selected by model selection. For the selection criterion, Torr's BIC [Tor00] is adapted to estimate the coding length of the structure-and-motion. Schindler et al. generate motion hypotheses for the entire sequence and then prune an optimal set which is computationally intractable for long sequences. Based on this work, Ozden et al. [OSG10] adapt the idea and use model scoring only in a temporal window rather than as a global batch optimization.

Most closely related to our system are the methods proposed by Schindler [SSW08] and by Ozden [OSG10]. They use interleaved segmentation and 3D reconstruction of tracked features into independent objects. Instead of directly sampling features and generating 3D object hypotheses, we incrementally cluster features to planes in 2D using homographies and then reconstruct and merge/split planes to independently moving objects in 3D. Thus, in the first step we use a simpler model in order to more robustly cluster tracked features to planes, which is followed by a second step in which we, that is, to reconstruct, merge/split planes and create the final object model. Finally, instead of a sparse point cloud we get a dense representation with planes, which can be directly used for robotic manipulation.

5.2 Basic Structure from Motion (SfM)

The reconstruction is based on the typical SfM pipeline which is described by various authors (e.g. Beardsley et al. [BTZ96]), Pollefeys et al. [PVGV+04]). Depending on the setup, i.e., the type of the camera projection model, the type of correspondences between the images, the unknowns in the motion model (calibrated, semi-calibrated, uncalibrated internal parameters, wide or short base-line,..) or the type of optimization algorithms (linear, non-linear), numerous variants have been proposed. Figure 5.1 shows a typical SfM pipeline. In this section we provide an overview of the core techniques including multiple view geometry, calibrated reconstruction, pose tracking and bundle adjustment. More details about the individual methods can be found in [Har08].

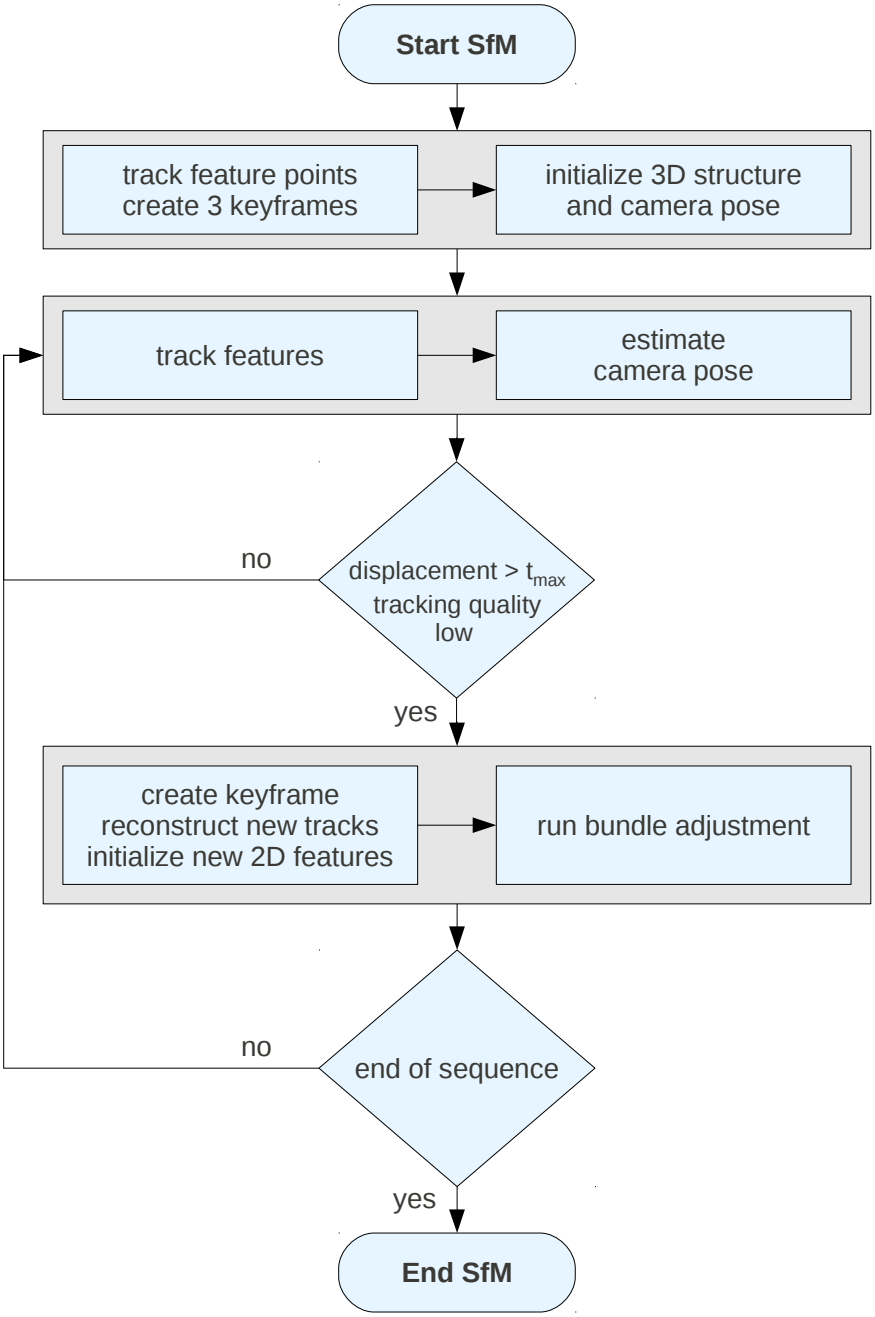


Figure 5.1: A typical Structure from Motion (SfM) pipeline.

5.2.1 Camera Geometry

Before we discuss the reconstruction, it is important to decide on the camera model. The most commonly used model in computer vision is the pin-hole perspective camera model. For most commercially available cameras, this model is a good compromise between simplicity and generality. For this type of camera a 3D world

point $\mathbf{X} = [\mathbf{X}\,\mathbf{Y}\,\mathbf{Z}\,\mathbf{1}]^{\mathbf{T}}$ can be projected to the 2D image point

$$\mathbf{x} = \mathbf{P}\,\mathbf{X}, \tag{5.1}$$

where $\mathbf{x} = [x\,y\,w]^T = [x/w\ y/w\ 1]^T$ is represented with homogeneous coordinates. The 3×4 matrix

$$\mathbf{P} = \mathbf{K}\ [\mathbf{R}|\mathbf{t}] \tag{5.2}$$

is called projection matrix and is derived from the extrinsic camera parameters, namely the camera rotation $\mathbf{R}$ and the camera translation $\mathbf{t}$, and the intrinsic camera matrix $\mathbf{K}$. The upper-triangular matrix

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

holds the camera internal parameters, where f_x and f_y stand for the focal length for the sensor in x and y direction expressed in pixel-related units, s encodes any possible skew between the sensor axes due to the sensor not being mounted perpendicular to the optical axis and c_x and c_y denote the optical center expressed in pixel coordinates. Thus, assuming an ideal camera without lens distortion and in case the external camera pose $(\mathbf{R}, \mathbf{t})$ is known, any 3D point can be transformed to the camera coordinate system and projected to the image using the camera matrix $\mathbf{K}$.

5.2.2 Reconstruction from Two Calibrated Views

In the last section, we assumed known 3D points and projected them to the image. Structure from motion considers the inverse problem, namely how to recover the 3D structure and the camera pose from image point correspondences. In the following, we assume that the camera parameters are available and fixed, hence 3D points are reconstructed from two calibrated views. One possibility to estimate the camera pose is based on the decomposition of the Essential matrix $\mathbf{E}$. Of course, in case there are only image measurements available, the recovered structure has an unknown scale factor and hence, the absolute distance is not available. In what follows, we briefly describe the basic reconstruction algorithm.

The images $\mathbf{x_{cam}}, \mathbf{x'_{cam}}$ of a 3D point $\mathbf{X}$ projected to the camera plane are related by the inner product

$$< \mathbf{x'_{cam}}, \mathbf{t} \times \mathbf{R}\mathbf{x_{cam}} > = 0, \tag{5.4}$$

$$\mathbf{x_{cam}}'^{T}\,\hat{\mathbf{T}}\mathbf{R}\,\mathbf{x_{cam}} = 0, \tag{5.5}$$

$$\mathbf{x_{cam}}'^{T}\,\mathbf{E}\,\mathbf{x_{cam}} = 0, \tag{5.6}$$

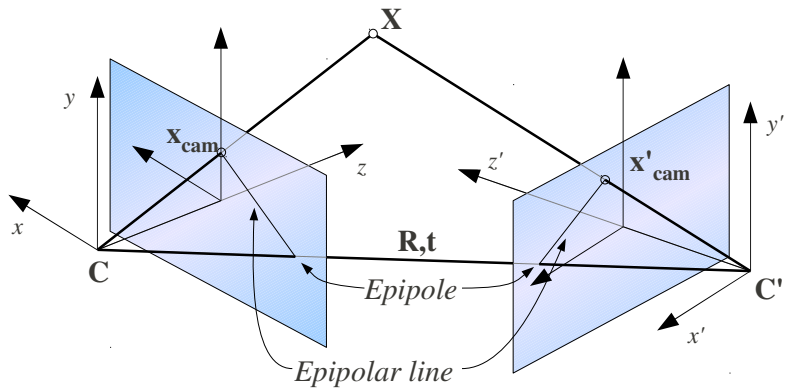


Figure 5.2: Epipolar geometry.

where the 3×3 matrix $\mathbf{E}$ is called essential matrix and defines the constraining line of two points in the camera planes. In combination with the camera matrix, this leads to the corresponding formulation

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0, \tag{5.7}$$

for two points $\mathbf{x}', \mathbf{x}$ in image coordinates, where $\mathbf{F}$ is called fundamental matrix and is computed from

$$\mathbf{E} = \mathbf{K}'^T \mathbf{F} \mathbf{K}, \tag{5.8}$$

with $K' = K$ if the images are from the same cameras. $\mathbf{F}$ can be estimated with the linear normalized 8-point algorithm if a minimum of 8 non-coplanar image point pairs are available (cf. Harley & Zisserman [Har08]). Figure 5.2 shows the geometric relationship of a 3D point and two cameras.

Once the essential matrix has been estimated, the singular value decomposition (SVD)

$$\mathbf{E} = \mathbf{U} \mathbf{S} \mathbf{V}^T \tag{5.9}$$

can be used to recover four solutions with

$$\mathbf{R} = \mathbf{U} \mathbf{W} \mathbf{V}^T \text{ or } \mathbf{U} \mathbf{W}^T \mathbf{V}^T \tag{5.10}$$

and

$$\mathbf{t} = \mathbf{u_3} \text{ or } -\mathbf{u_3}, \tag{5.11}$$

where $\mathbf{u_3}$ is the last column of $\mathbf{U}$ and

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.12}$$

Figure 5.3 shows, the geometric interpretation according to the four solutions. To select the correct solution, the depth of a 3D point must be positive in both

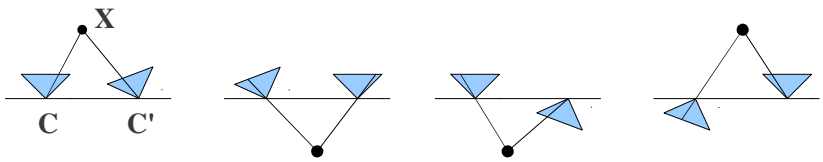


Figure 5.3: Four solutions of essential matrix decomposition.

cameras. Hence, for each combination of $\mathbf{R}$ and $\mathbf{t}$ a test point must be triangulated and transformed to both camera coordinate systems.

A general 3×3 matrix has 9 degrees of freedom, due to the scale invariance and the rank deficiency however, the fundamental matrix $\mathbf{F}$ has only 7 degrees of freedom. If the internal camera parameters are known, the essential matrix, which has five degrees of freedom, can directly be estimated using the 5-point algorithm proposed by Nister [Nis04]. Hence, for the case of known internal parameter, the pose transformation of a camera can be estimated up to scale, if five non-coplanar point pairs in the two images are available.

5.2.3 Feature Tracking and Pose Estimation

Establishing feature correspondences in image pairs is one of the fundamental challenges in computer vision. Depending on the camera viewpoint of the two images, two basic scenarios can be distinguished: wide baseline and narrow baseline applications. For the first one, where the viewpoints of the cameras differ significantly, features must be used which are robust to geometric transformations. As representatives for lots of variations, we pick out two popular methods. The first one, which we already mentioned in the Chapters 3 and 4, is the *Scale Invariant Feature Transform (SIFT)* proposed by Lowe [Low04]. SIFT interest points are based on salient scale invariant points in a *Difference of Gaussian (DoG)* image pyramid. To find the corresponding points in two images, histograms of oriented image gradients are computed and compared. Another established method is proposed by Bay et al. [BETG08], who use an integer approximation to the determinant of the Hessian blob detector and the sum of approximated 2D Haar wavelet responses for the description of the blob. In most visual Simultaneous Localization and Mapping (SLAM) and SfM applications a small baseline is assumed. Hence, in the majority of cases these approaches get along with simpler features. KLT is a successfully applied method proposed by Tomasi et al. [TK91]. This approach assumes brightness constancy and computes the displacement of local features by minimizing the intensity difference between the locations in two images supported by a small window. Tracked patches must have gradients in at least two different orientations. In addition to the original implementation various methods to detect corner-like structures are proposed (Harris et al. [HS88], Schmid et al. [SMB00],

Rosten et al. [RPD10]).

Once the initial structure is recovered and the point correspondences are established, the pose can be computed with a minimum of three points (Haralick et al. [HLON94]). To obtain a unique solution, this algorithm is embedded in the robust estimation scheme RANSAC (Fischler et al. [FB81]). The described method tracks each feature independently from the others, without considering any possible geometric relationship between their displacements. Hence, this method is quite general method and can be applied in many types of scenes including dynamic scenes. The drawback is that a good source of information, namely the predicted camera pose is not used. For visual SLAM applications, where subsequent image frames are guaranteed, tracking of features and pose can be jointly done within a prediction framework (e.g. extended Kalman filter (EKF) [TBF05]).

5.2.4 Bundle Adjustment

After the initial reconstruction, tracking of 3D points and reconstruction of points which newly appear goes hand in hand. That is, the current camera pose is used to reconstruct new tracks of 2D points, which are then used to track the camera pose in subsequent frames. The incremental scheme leads to an accumulation of error of the pose, as well as of the structure. Hence, a non-linear minimization is applied to all inliers to get a more suitable solution. If there are n images and z 3D points, $11\,n+3\,z$ parameters have to be optimized simultaneously in order to minimize the global error function, such as the 2D reprojection error. Typically, the minimization is carried out using the Levenberg-Marquardt (LM) algorithm. For the case of SfM, points appear in a small subset of camera frames. According to [Har08], this sparse block structure can be exploited in order to minimize the optimization cost. Additionally, in a typical SfM-pipeline, the points are not reconstructed and optimized in each frame, but only if the mean displacement exceeds a threshold.

5.3 3D Object Modelling Based on Planes

The final results of our system are 3D models of objects. Approaching this goal from object reconstruction, our system is strongly related to the dynamic SfM frameworks [SSW08, OSG10]. In [OSG10], Ozden et al. defined the following requirements:

1. Determine the number of independently moving objects at the beginning of a sequence, and
2. whenever that number changes, segment the feature tracks into different moving objects in each frame.
3. Compute the 3D structure of all features and the camera motion for the frames.

4. Resolve geometric ambiguities.

5. Robustness to short feature tracks due to occlusion, motion blur, etc.

6. Scale to realistic recording times.

To determine independent objects, they propose interleaved segmentation and 3D reconstruction of the feature tracks. Instead of directly sampling features and generating 3D object hypotheses, we incrementally cluster features to planes in 2D and track them. Thus, the first items are approached with a simpler model in 2D followed by reconstruction, clustering and splitting of planes to objects in 3D, which leads to more robust estimations.

To reconstruct planes, which are not assigned to a 3D motion model, we use a standard SfM pipeline similar to Nister et al. [NNB04] and Klein et al. [KM08]. Therefore, the nonlinear optimized homography is directly decomposed to initialize the first camera pose (cf. Ma et al. [Ma04]). In the following frames, the relative motion from C^{-1} to C is estimated using RANSAC [FB81] and a direct least squares solution between the two point sets (cf. Haralick et al. [HJL$^+$89]). A sparse bundle adjustment implementation by Lourakis [LA09] over the last N frames is used to refine camera poses and 3D points of the plane. Once a plane is reconstructed, our algorithm tries to incorporate planes greedily in case of consistent motion.

Algorithm 4 gives a detailed outline of the piecewise planar object modelling pipeline and Figure 5.4 depicts the events, including detection, tracking, merging and splitting of planes.

5.4 Homographies for 3D Reconstruction

In Section 5.2 we have shown how to use epipolar geometry for the initial pose reconstruction. The proposed object modelling approach is based on planes, represented by homographies, which can be used directly to compute the camera poses.

Additionally to the essential matrix which constrains a point in the left image to a line in the right image, the known homography constrains a point in the left image exactly to one point in the right image. This can be formalized with

$$\mathbf{x'_{cam}} = \mathbf{H_{cam}}\,\mathbf{x_{cam}}, \tag{5.13}$$

$$\mathbf{H_{cam}} = \mathbf{R} + \frac{1}{d}\,\mathbf{t}\,\mathbf{n}^T \tag{5.14}$$

where $\mathbf{H_{cam}}$ is the projective transformation, computed from four image points in camera coordinates which are in general form, i.e. no three of them are collinear and $\mathbf{n}$ stand for the unit normal vector to the plane in the first camera. According to [Ma04], the homography matrix can directly be decomposed to compute the rotation and the translation between the cameras. Especially if the scene contains

Algorithm 4 Piecewise planar object modelling pipeline

1. Instantiate new interest points (IPs)
2. Track interest points
3. Track planes modelled by homographies and try to estimate 3D motion for existing objects
   if plane does not support 3D motion **then**
   - trigger split event and create new objects from current and past keyframes

   end if
   if average displacement of the IPs $< d$ pixels **then**
   - **goto** step 1

   else
   - initialize a new keyframe and continue

   end if
4. Detect and renew planes
5. Merge and reconstruct planes greedily
   if new plane supports active object motion model **then**
   - insert plane

   else
   - create new 3D object and motion model (SfM)

   else if
6. Refine objects using incremental bundle adjustment
7. **goto** step 1

a dominant plane, this method is an alternative for the approach described in Section 5.2. The mathematical details of the decomposition can be found in Appendix A.

Additionally, in [LF96] Luong et al. have shown that a least-squares fundamental matrix can be computed from

$$\mathbf{H}^T\mathbf{F} + \mathbf{F}^T\mathbf{H} = 0, \tag{5.15}$$

where at least two homographies improve the accuracy of the reconstruction.

5.5 Pose Tracking

The next step in a standard SfM pipeline is to track the camera pose with respect to the environment. For our scenario, where a robot learns the appearance of an object, we are concerned about the pose with respect to a specific object. Standard SLAM approaches assume that more than 50% of features move consistently with the camera, i.e., they assume a predominantly static scene. In contrast, we deal with dynamic scenes, where the robot explores the environment and interacts with parts of the scene to verify object hypotheses. Hence, the assumption that more

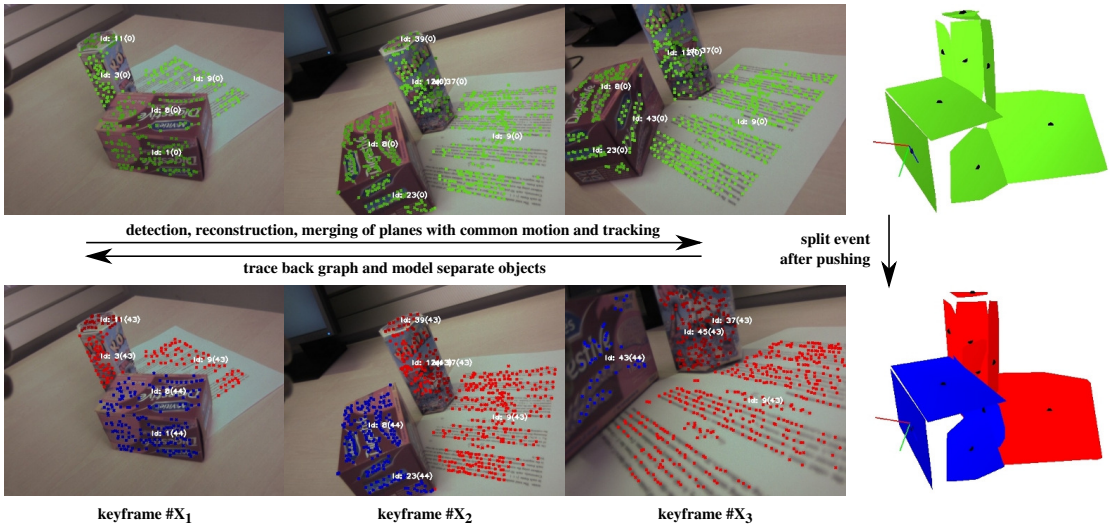


Figure 5.4: The upper row shows three keyframes of sequence 1 (897 frames) with detected planes in green which are merged because of common 3D motion. The brightness of the interest points indicates the assignment to different planes. After the gripper (two black dots on the left image border) pushes the plane 43, the keyframe-graph is traced back and the object model (44) and the background object (43) are created (lower image row). Changing plane ids of the top surface of the hexagonal object indicate that planes represented by homographies are substituted with better explanations.

than 50% of the features move consistently is not guaranteed anymore. To overcome this limitation we developed a pose estimation scheme which focuses on parts of the scene, where the robot intends to interact. In detail, the robot moves and creates object hypotheses out of planar scene parts. It then picks up a possible interaction point and the pose tracking algorithm focuses on the corresponding plane. To further stabilize tracking, consistently moving features from the remaining part of the scene are included. Hence, the termination criterion of the standard RANSAC algorithm described in Algorithm 1 for the case of dominant plane detection is adapted to

$$\epsilon = \frac{I_t + w\, I_s}{N_t + w\, N_s} \tag{5.16}$$

with

$$w = k_w \frac{N_t}{N_s} \tag{5.17}$$

where I_t is the number of inliers of the target plane, I_s is the number of inliers of the remaining planes tested for support and N_t and N_s are the corresponding total numbers of points. The weight w is used to force tracking of the target plane, where we set the constant $k_w = 1$ and thus the weight between target plane and

the remaining supporting planes is equal. According to the weight k_w sampling of features also needs to be distributed equally between the target plane and the supporting planes. Hence, independent of the size of the planes, tracking always focuses on the target plane. The only assumption is that the quality of the feature tracking is similar. If there are more outliers on the target plane, then the constant $k_w < 1$ needs to be adjusted to a lower value. Additionally, if the target plane is more prominent than the remaining planes, i.e., there are more interest points on it, we bound the weight $w = 1$ to one. Hence, the tracking algorithm switches to a standard RANSAC behavior. To compute the pose hypotheses we use a standard 3-point algorithm (Haralick et al. [HLON94]) and thus, in contrast to the RANSAC for plane detection described in Section 3.2, m is reduced to three.

In summary, in case the robot needs to keep track of a specific part of the scene, the corresponding interest points are marked, the algorithm focuses on these points and consistently moving interest points from the remaining part of the scene are used to robustify tracking.

5.6 Incremental Bundle Adjustment and Absolute Scale

In the previous sections we discussed how to reconstruct features and we have shown how to robustly track the camera pose in pairs of images. If the camera pose is sequentially tracked in an image sequence, the error is accumulated, which leads to a drift of the pose. To avoid this a nonlinear optimization of the camera poses for a bundle of images is introduced. Bundle adjustment is a well known method designed to minimize a global error [TMHF00]. Unfortunately, minimizing a global error becomes rapidly intractable. Hence, for real-time applications, such as SLAM, it is common to perform the bundle adjustment incrementally. That is, the bundle adjustment estimates the camera parameter and the 3D structure of the n most recent key-frames using the projections of the N most recent key-frames (e.g. $n = 3$ and $N = 10$). Thus, the computational effort is bounded with the 3D points in the last n frames. The drawback of that method is twofold: First, since we only use the most recent frames, the system tends to drift and the error increases with the number of key-frames. And secondly, it is obvious that if we only use one camera, the object can only be reconstructed up to an unknown scale factor.

Both drawbacks can be overcome if additional information from the encoders of the robot is introduced. Hence, we scale the camera translation using the pose of the robot. The scale corrected location

$$\tilde{\mathbf{t}_\mathbf{C}}(t) = \mathbf{t}_\mathbf{C}(t-1) + \frac{\|\mathbf{t}_\mathbf{R}(t) - \mathbf{t}_\mathbf{R}(t-1)\|_2}{\|\mathbf{t}_\mathbf{C}(t) - \mathbf{t}_\mathbf{C}(t-1)\|_2}(\mathbf{t}_\mathbf{C}(t) - \mathbf{t}_\mathbf{C}(t-1)) \tag{5.18}$$

is computed from distance ratio of the robots location change

$\Delta \mathbf{t_R} = \mathbf{t_R}(t) - \mathbf{t_R}(t-1)$ and the corresponding transformation estimated from the camera $\Delta \mathbf{t_C} = \mathbf{t_C}(t) - \mathbf{t_C}(t-1)$ between the last and the current key-frame. The scale has to be corrected directly after the camera pose is estimated. For the following triangulation of the new corner tracks and for the incremental bundle adjustment, the new camera location $\tilde{\mathbf{t_C}}$ is used. Hence, the drift of the incremental approach is reduced and additionally, the absolute scale of the object is determined.

5.7 Merging and Splitting in a Probabilistic Formulation

To recapitulate, in our scenario a robot explores the environment. It moves around and perceives some parts of the scene that indicate individual objects and motivate the robot to interact. In the preceding chapters, we detected homographies, i.e., planar parts estimated from image pairs, and reconstructed them with the motivation to create object hypotheses and to provide features for interaction. The following sections describe a framework for separating piecewise planar scene parts to individual objects. Motivated by Palmer [Pal99] – who stated that, although the vast majority of objects in ordinary environments are stationary for the vast majority of the time, objects that move are important – we first merge planes depending on common motion and, if they start moving separately, we split them to individual objects. The appearance model of planes computed from color and from a 3D interest point adjacency graph allows to assign already occluded planes to the split objects. Hence, the framework is able to create individual object models from planes visible in the current image and from currently occluded planes seen in a previous frame.

5.7.1 Merging of Planes with Consistent Motion

The merging of planes is nothing more than to check whether the motion of a new plane is consistent with the motion of an existing object. We developed a strategy to greedily assign homographies to a motion model. Analogous to Equation 3.9 in Chapter 3, a formulation is developed, which results in a confidence

$$p_{ij} = -\nu_1 + \frac{1}{n}\sum_{k=1}^{n}\left((1-\nu_2) + \nu_2 p(f_{i,k}^{proj}|H_j)\right) \tag{5.19}$$

that a plane i moves consistently with an existing motion model of an object j, where $p(f_{i,k}^{proj}|H_j)$ is the probability that an interest point k of a plane i belongs to the 3D object H_j. This is modelled using a Gaussian error model. The camera pose of object j is used to compute 3D points for plane i. Then the projections are compared to the corresponding tracked image points. ν_1 and ν_2 are constants

to weight the different factors, where ν_1 is an offset which must be reached to be considered as moving together. Homographies are assigned to the motion model according to the highest probability p_{ij}.

5.7.2 Separating Planes in case of Different Motions

While tracking the camera motion relative to an object hypothesis (cluster of planes), each individual plane is continuously tested if it still supports the motion. For this, the formalism outlined in the last section is used and in case p_{ij} is low, i.e., planes start moving separately, a split event is triggered. Planes which are currently visible build the initial split object hypotheses from which we compute the appearance model A_j. Then the frame of the original object (before splitting) – which is stored in a key-frame based graph structure – is traced back and the occluded planes are assigned to the most plausible split object. For this, the pseudo-likelihood

$$p^*(a_i|A_j) = \sum_{k=1}^{N} \left((1-\nu_3) + \nu_3 p(f_{i,k}^{3D}|H_j)\right) + \log(p(c_i|C_j)) \tag{5.20}$$

is computed, which combines the 3D adjacency of interest points and the color in a probabilistic manner. Interest point adjacency (first term) is based on a probabilistic voting scheme. For this purpose, a neighborhood graph of all currently available 3D points is constructed and the mean μ and the standard deviation σ of the length of edges which connect points of the same plane are computed. Then μ and σ are used to compute Gaussian votes $p(f_{i,k}^{3D}|H_j)$, where each 3D point of a target plane votes for the nearest object and thus the object which is close to the plane accumulates more votes and gets a higher probability that the plane belongs to that object. The second term models the color distribution of the objects. For this we build the $8 \times 8 \times 8$ bin color histogram c_i of the target plane i and the histogram C_j of the object j to which the plane should be assigned. We use normalized rgb colors to be insensitive to brightness differences of object planes. The border of the plane is approximated by the convex hull of the interest points. For comparison of color models, we use the Bhattacharyya coefficient

$$p(c_i|C_j) \sim \sum_q \sqrt{c_i(q)C_j(q)}. \tag{5.21}$$

Hence, the probability of a plane i which has to be assigned to an object j consists of a probabilistic vote of each interest point to the nearest object and a probability describing the color similarity. While we are aware of the fact that assigning occluded planes based on colour and 3D interest point adjacency is a critical point, our experiments indicate that for our scenarios, where only a few objects are modelled simultaneously, this is a good criterion.

(a) First reconstruction

(b) Labeled image to initialize the color model and the interest point proximity.

Figure 5.5: Initialization to evaluate the plane assignment using colour and point proximity.

5.8 Tests and Evaluation

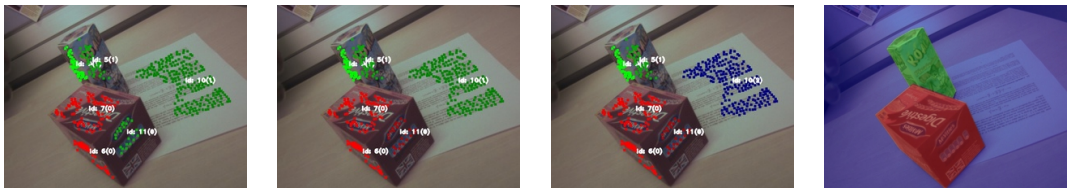
For all experiments we use Shi-Tomasi interest points [ST94] and a KLT-tracker [TK91]. In [MSC+09] it has been shown that a sub-pixel refinement essentially improves pose estimation. Hence, we use the affine refined location of the interest points with sub-pixel accuracy and finally compute a non-linear optimized homography using *homest* [Lou06].

To test our system, we use five videos with about 800 frames each. Motivated by our cognitive robotic scenarios, the sequences show packaging of arbitrary shapes typically found in a supermarket (see Figure 5.4). We placed two different objects on a table and manually moved camera and gripper around them in a way that one half of the objects is already occluded before the gripper pushes one object.

5.8.1 Evaluation of Plane Assignment using Color and Interest Point Proximity

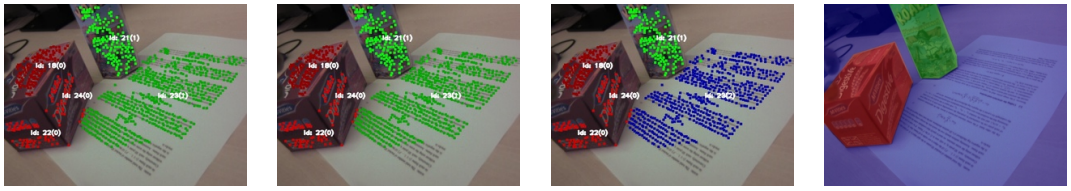
First, we evaluate the cues described in Section 5.7.2, which we used to assign occluded planes to individual object hypotheses. For this, we select about 200 keyframes of the first three videos and mark the objects. After reconstruction of the planes visible in the first frames, we build the color model and the interest point proximity model (see Figure 5.5). In the following frames, these models are used to assigned planes to separate objects, according to Equation 5.20. In all tests we set $\nu_3 = 1$ and we used the mean Euclidean distance d_m of the interest points to model the Gaussian $\mathcal{N}(d_m, (1.5\, d_m)^2)$).

Figures 5.5, 5.6, 5.7 and the Tables 5.1 – 5.4 show the results of the evaluation. We compare the individual cues color (Equation 5.21) and proximity (first part of Equation 5.20) with the combination of them. Figure 5.6(d) shows a typical ground truth image. We marked the object with red and green and the background with blue. The ideal case would be that the method is able to assign planes to "known"



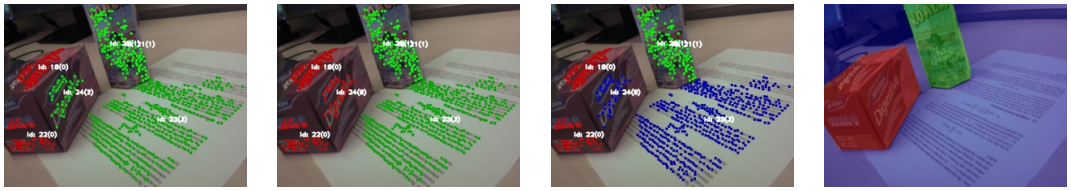
(a) (b) (c) (d)

Figure 5.6: Comparison of plane assignment using color (Figure 5.6(a)), interest point proximity (Figure 5.6(b)), the combination of color and proximity (Figure 5.6(c)) and the according ground truth image (Figure 5.6(d)) for sequence 1 frame 280.



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

Figure 5.7: Comparison of plane assignment sequence 1 frame 480.



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

Figure 5.8: Comparison of plane assignment sequence 1 frame 515.

	color [%]	proximity [%]	combined [%]
true	75.0	84.6	96.3
false	25.0	15.4	0.0
undefined	0.0	0.0	3.7

Table 5.1: Comparison of plane assignment for sequence 1

models and that it also labels unknown (uncertain) plane assignments. Because of the problems arising with thresholds, we decided to assign the planes to the model with the higher probability. Only in case the value drops below 0 because of the log-likelihood we set the label to "undefined". This is only possible for the combined color and proximity method. The numbers in the Tables 5.1, 5.2, 5.3 and 5.4 indicate that our method provides an appropriate heuristic for this setup.

	color [%]	proximity [%]	combined [%]
true	70.5	97.1	96.2
false	29.5	2.9	0.0
undefined	0.0	0.0	3.8

Table 5.2: Comparison of plane assignment for sequence 2

	color [%]	proximity [%]	combined [%]
true	70.5	97.1	96.2
false	29.5	2.9	0.0
undefined	0.0	0.0	3.8

Table 5.3: Comparison of plane assignment for sequence 3

	color [%]	proximity [%]	combined [%]
true	78.8	93.2	96.9
false	21.2	6.8	0.0
undefined	0.0	0.0	3.1

Table 5.4: Comparison of plane assignment for all sequences

Although color is a rather weak cue, in combination with the interest point proximity it helps to avoid a hard threshold. Figure 5.6 shows an example where color proposes a wrong plane assignment but the combination of color and interest point proximity finally leads to a correct decision shown in Figure 5.6(c). Figure 5.8 shows an example where color and proximity propose different assignments and the combination of both declares the plane as *unknown*. This is not correct, but in case of a high uncertainty we prefer the *unknown* label instead of a wrong assignment. This can be seen as knowledge gap of the robot, which leaves room for further exploration.

5.8.2 Tests of the Integrated System

The goal of the experiments is that our system detects the planes, reconstructs, tracks and merges them depending on common motion and that finally, after pushing one object, the system creates two separate piecewise planar object models. Figures 5.4, 5.9, 5.10, 5.11 and 5.12 show the qualitative results of our system. Planes merged to one object are drawn with the same color, whereas the brightness of interest points indicates the assignment to different planes. In each figure the third image of each row shows the perspective of the camera shortly before/after the object is pushed and the last one depicts the reconstructed objects. Figure 5.4 shows the whole event chain, that is, detection, reconstruction and merging of planes with a common motion colored green and separating planes as they start moving independently (indicated in red and blue). In Sequences 1, 2, 4 and 5,

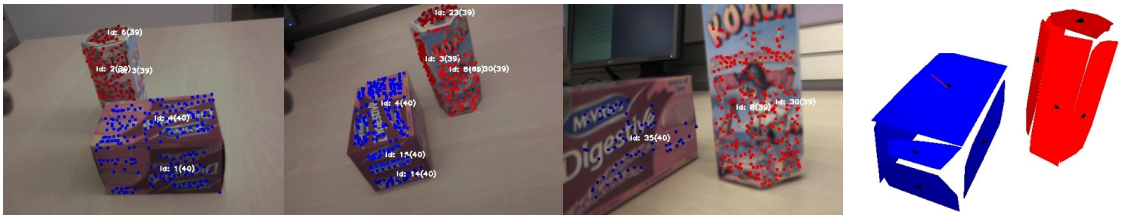


Figure 5.9: Sequence 2 (715 frames).

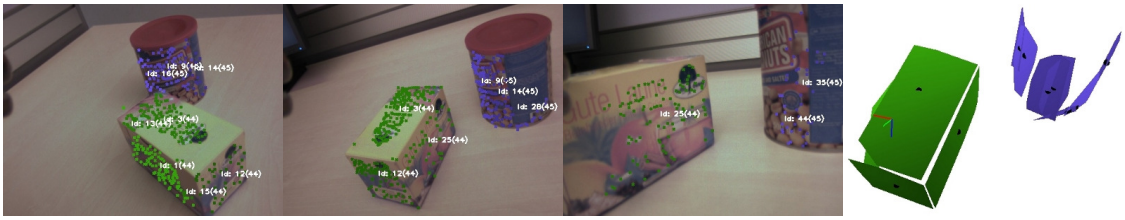


Figure 5.10: Sequence 3 (870 frames).

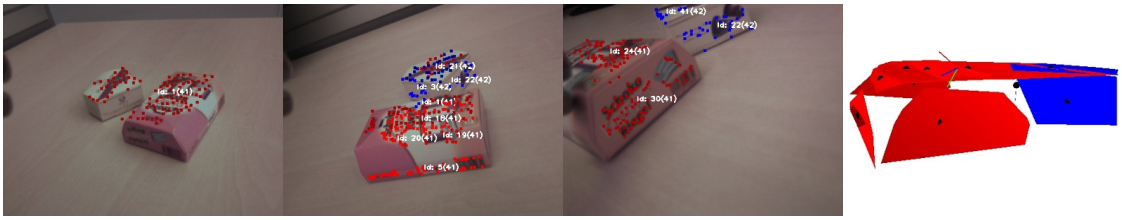


Figure 5.11: Sequence 4 (543 frames).

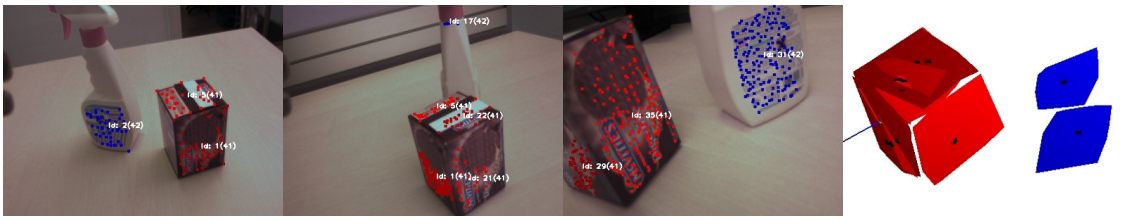


Figure 5.12: Sequence 5 (811 frames).

shown in Figures 5.4, 5.9, 5.11 and 5.12, object modelling was successful and accurate as expected. The 3D reconstruction (right most image of each row) shows that sometimes parts of an object, which we intuitively would mark as one plane, are split. On the one hand, this is due to the fact that these planes are in fact not flat but a little bit curved, and on the other hand that model selection within our

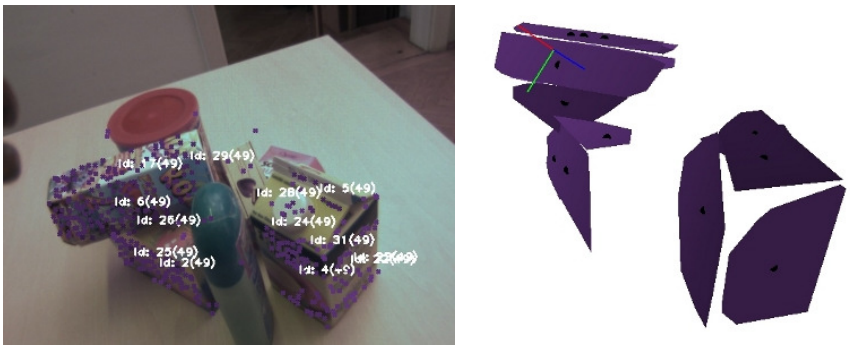


Figure 5.13: Example image and reconstruction of a small, more complex sequence which shows the limits of our system. Planes of the three dominant objects at the front are reconstructed, while the object in the center of the image and the objects in the background are not detected because of low texture and too few features.

plane detection algorithm replaces a plane in the following keyframes if a better, more complete/accurate plane is found. Figure 5.11 shows one of the failures that might occur. These two objects have approximately the same height and thus, one joined explanation instead of two separate ones was favored. In the case shown in Figure 5.11, this results in a much too big top surface of the red object which covers a part of the heart-shaped box. Figures 5.12 and 5.13 show the limits of our system. Our reconstruction relies on planes modelled by homographies and thus for one plane a minimum number of five interest points is necessary ($4 + 1$ that supports the homography). Because of reliability issues, we used a threshold of 10 points. Hence, in Figure 5.12, even though a small plane is detected (shown at the top of the middle image), the top of the cleaner bottle is completely lost. In Figure 5.13, the object in the middle has hardly any texture and the finer scene details in the background are invisible for our system, whereas the three prominent objects are nicely recovered.

5.9 Discussion

This chapter presented an approach to model the scene with piecewise planar parts using tracked interest points. Homographies build the basic plane model and are directly used for reconstruction of the scene. By using this approach, it is possible to model separate objects from pushing planar patches. If accidentally several objects are pushed at the same time, different motions will occur and the objects will be modelled as two different items. Consistent with plane detection we propose a Minimal Description Length (MDL) formulation to merge/split planes to independently moving objects in 3D. Merging as well as splitting is triggered based on a probability which combines 3D motion, structure and color information of the planes. Instead of a sparse point cloud, which is typical for Multi-body Structure-

and-Motion, we get a dense representation with planes.

The evaluation of the pseudo-likelihood, designed to assign occluded planes to separate objects if a split event occurs, provided promising results for table top scenarios and the packaging data sets. As shown in Table 5.4, the performance is rather low if only color is used, but in combination with *proximity* of the 3D location of interest points, color seems to provide reliable base costs to avoid wrong plane assignments.

Limitations of the system are shown in the Figures 5.11 and 5.12. In environments with low texture, two typical failures occur. Firstly, parts of objects, such as the top of the aerosol can in Figure 5.12 or the ground plane in most of our sequences, are totally missed. Secondly, because of missing parts of the scene, hallucinating planes are detected, such as the top surface depicted in Figure 5.11, which covers coplanar planes of two individual objects. Beside a pixel-wise segmentation using Markov random fields, or the dense reconstruction proposed by Newcomb et al. [ND10], this weakness can be overcome by using additional depth information from an RGB-D sensor (e.g. Kinect recently developed by PrimeSense).

In the next chapter, we describe the proposed recognizer and introduce our notion of completeness. Furthermore, we already use Kinect data to learn the object models.

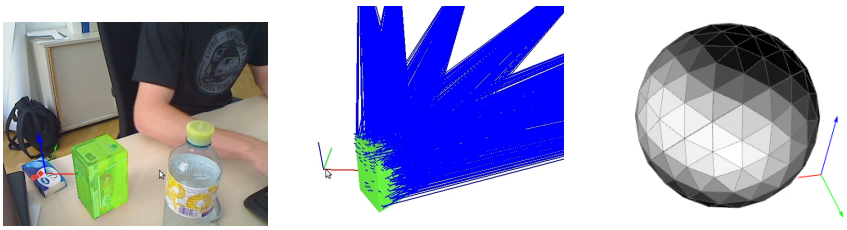
Chapter 6

Model Completeness and Object Recognition

Autonomous learning robots often face the exploration-exploitation dilemma in one form or another: Should I continue learning (exploration) or make use of the models learned so far (exploitation)? Or to put it another way, when have I learned enough in order to satisfactorily complete my tasks? This means that not only does the robot need to represent its knowledge, but also the limits of its knowledge. This becomes all the more important in integrated robotic systems which have to make decisions based on observations drawn from a multitude of modalities. All of these observations will carry some degree of uncertainty, and it is paramount that these uncertainties are formulated consistently in order to support well informed decisions. Moreover, the robot should be enabled to improve performance in future tasks by reducing uncertainties, i.e. it should be able to derive actions that reduce gaps in its knowledge. Again, this increase in knowledge must be formulated consistently to allow planning for optimal knowledge gathering actions.

In this chapter, we are concerned with learning 3D object models for recognition. Object recognition has made impressive advances and increasingly powerful methods have shown their applicability in challenging scenes [GL06, OFL07, RBSF09]. The focus of research, however, often lies on optimizing recognition (robustness, speed, generality) while the learning phase typically takes place offline, i.e. outside the robot's normal execution of tasks. Consider an autonomous robot which has to perform tasks including the recognition of objects such as fetching various items, when these items are not known in advance. So one of the tasks of the robot, e.g. while not being occupied with more urgent things, will be to wander about and learn new objects which might feature as part of a fetching task at a later time.

We explicitly concentrate on the process of acquiring these object models. This requires representing the completeness of models acquired so far as well as mechanisms to support planning for further knowledge gathering actions. We present



(a) (b) (c)

Figure 6.1: Learning an object model: The object in the scene (left) and bundles of features with their view vectors (in blue) after acquiring some views of the object. View sphere (right) with brighter shades of grey indicating that the object has been learned from the respective direction.

three probabilistic measures of *observed detection success*, *predicted detection success* and *model completeness* which allow to reason about when to extend the model, where to look next as well as to predict the probability of successful detection given the model learned so far.

Figure 6.1 illustrates the basic idea. Object models are based on associating SIFT [Low04] or SURF features [BETG08] with 3D locations on the object's surface. The 3D location either is computed by assigning the features to the object piecewise planar surface model described in Chapter 5 or it is acquired by an rgb-depth sensor recently developed by PrimeSense.

Models are built online, while the object is tracked based on the model acquired so far. New features are mapped onto the 3D object surface and associated with the view direction from which they were captured. Thus we know which object views are covered so far, which is represented as "bundles" of view vectors associated with features in Figure 6.1(b). The "view sphere" shown in Figure 6.1(c) illustrates the completeness of the model, where brighter parts indicate a higher probability of detecting the object from the respective direction. A weighted sum over the whole sphere gives the expected probability of detection and the fringes of the bright areas constitute learning opportunities.

In [ZPMV11] we have shown that our probabilistic model is well suited to predict when to learn new views and to predict the probability of successful detection given the model learned so far.

After a state of the art review, we describe learning of new object views in Section 6.2, followed by a description of the proposed codebook structure in Section6.3. Section 6.4 gives an overview of the proposed recognition approach, including the building of a vocabulary tree (Section 6.3.1), voting for object hypotheses (Section 6.4.1) and estimation of the poses of multiple objects (Section 6.4.2). In contrast to interactive learning of object models described in Chapter 5 this Chapter concentrates on probabilistic measures proposed for detection success and completeness (Section 6.5). Hence, for learning of the object model we use a simpler

approach with static objects and a table plane constraint. Finally, the evaluation is presented in Section 6.6.

6.1 State of the Art

In the following, we first review the state-of-the-art for the recognition of object instances and object classes. Next, we focus on online learning of object models and on methods for efficient matching of interest points, followed by a review of approaches targeted to interactive modelling. Finally, we present approaches that combine object recognition and pose registration.

Various approaches to recognize specific object instances [Low04, RLSP06, SZ09] or object categories [FPZ07, LLS08] in monocular images have been proposed. These approaches are optimized for recognition in complex environments, whereas training is done offline in a (weakly) supervised fashion. In contrast we concentrate on learning objects online while tracking the model acquired so far. There also exist some approaches which optimize detection for tracking in real time [VLF04, OLFF06, OFL07]. Özuysal et al. [OFL07] rely on randomized trees for wide baseline feature matching. Assuming the object moves slowly in the first frames, this approach is able to detect the object pose to initialize the initial model. In the following frames, new features are added and features which cannot be reliably found are discarded. Once all the training frames are processed, bundle-adjustment is used to refine the geometry. Özuysal et al. use simple image patches to describe the interest points. Instead, we rely on SURF [BETG08] which can be computed very fast and it has been shown that these features are adequate for recognition of multiple objects in complex environments.

The approaches described in [GGB07, RDB06] focus on online learning of object models to avoid manual labeling of training images. Whereas [GGB07] combines a robust background model, a tracker and an online learning method, in [RDB06] classifier-based keypoint descriptions allowing incorporation of background information are learned.

With respect to efficient interest point matching, the work by Riemenschneider et al. [RDB07] is most similar to our approach. They use a vocabulary tree originally developed by Nister et al. [NS06] which represents prototype descriptors in a hierarchical structure for a fast matching of interest points. Instead, Lowe [Low04] proposes an approximate nearest-neighbor method based on a kd-tree for matching of interest points. Sivic et al. [SZ09] propose matching of query interest points with a codebook that is weighted based on the entropy. For the ranking of matched database images a Term Frequency - Inverse Document Frequency (TF-IDF) scoring is used. Similar to [NS06], Philbin et al. [PCI+07] rely on a hierarchical structure and introduce a method based on randomized trees to avoid the quantization effects from k-means clustering.

An interactive modelling approach was introduced in Pan et al. [PRD09]. For

this purpose, an online Structure-from-Motion algorithm is used for camera pose tracking, and the geometry is reconstructed using tetrahedron carving on the Delaunay tetrahedralization of the point cloud. Weise et al. [WWLG11] propose an interactive 3D scanning system that allows users to scan complete object geometry by turning the object around in front of a real-time 3D range scanner. To avoid artefacts, they propose online loop closure and outlier handling for model reconstruction. Both approaches aim to reconstruct the 3D geometry of objects, either by means of Structure-from-Motion or with an additional sensor that provides the 3D data.

There also exist some approaches which combine object recognition and pose registration [GL06, EKH05, RBSF09, TRS10, RS10, HBN07]. Gordon and Lowe [GL06] build a 3D model composed of SIFT descriptors in an offline training phase by performing structure and motion estimation. The online phase then uses RANSAC to estimate 6D pose from 2D-3D correspondences. The system, though, is geared at augmented reality applications and the scene is not segmented into objects. The work by Collet et al. [RBSF09] is most similar to our approach. They extend the above for application in the robotics domain, specifically by augmenting RANSAC with a Mean-Shift clustering step to allow recognition of multiple instances of the same object. However, the system does require manual segmentation of the object in each training image. Furthermore, the obtained sparse 3D point model has to be manually aligned with a CAD model of the object, so the whole procedure requires considerable user intervention. Instead, we avoid user interaction by learning objects from salient regions popping out from a dominant ground plane. Furthermore, we combine different sensor modalities, namely the rgb-image and the point cloud acquired with the rgb-depth sensor *Kinect* to build scale corrected object models and to improve pose registration.

6.2 Online Learning of Objects

The first step when learning object models is to detect the object and segment it from the background. Typically, attention approaches and saliency operators are used for this purpose (e.g. [IKN98], [FRC09] and [PZV11a]). In contrast, here we use a modified version of *plane pop-out* proposed by Zhou [ZZV09]. This method relies on a robust plane estimation to segment objects that pop out from the table plane. It is more robust, but the objects need to be located on a flat surface and must be placed individually with a minimum distance of a few centimeter.

Thus, we propose to detect and segment objects on tables, build initial object models, track the object model acquired so far and add new views to the model whenever there is a lack of information (see Figure 6.2). For the detection of table planes, we use 3D point clouds acquired with the rgb-depth sensor *Kinect* recently developed by PrimeSense and Microsoft.

In detail, first the table plane is robustly detected using RANSAC. The point

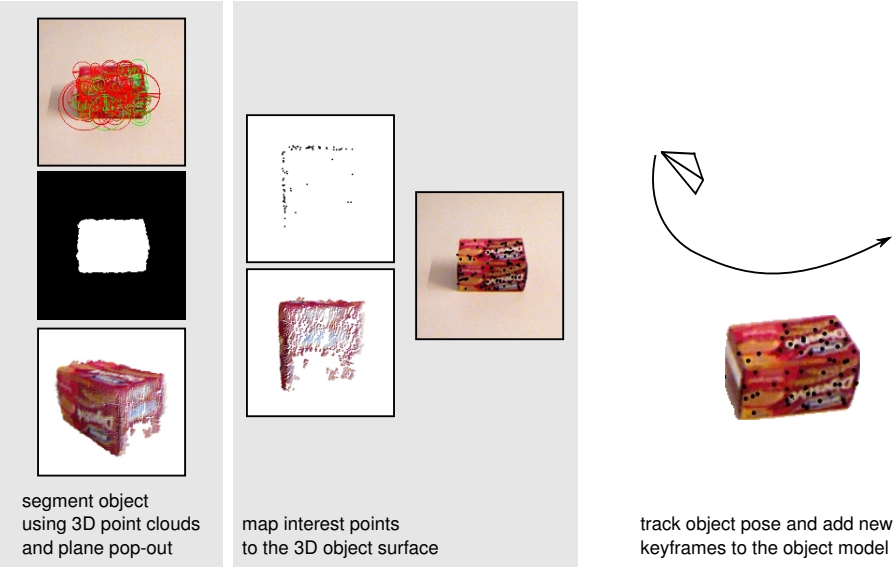


Figure 6.2: Learning of interest point models. The top image of the left column shows interest points detected within the region of interest (center) which is computed from the point cloud. The bottom image shows the point cloud where the table plane is already removed. Note that, although the images show a clean table plane because of the rgb-depth sensor, this approach handles both: highly cluttered scenes and clean untextured tables. The center column indicates the alignment of interest points and 3D structure, where the right part shows 3D locations of the interest points projected to the object. The right column of the image depicts a possible camera trajectory, where keyframes are added and hence, a more complete object model is created.

cloud acquired from Kinect is organized in a 2D grid. This can be exploited for clustering the remaining 3D points. Instead of using a neighborhood graph (e.g. the ANN [AMN+98] or FLANN [ML09]), we implemented a connected component analysis which directly parses the point cloud grid. To split objects which touch in the grid, but are separated in depth, a cut-off threshold of the depth value is used. Then, interest points within the region of interest, detected in the corresponding grey scale image are associated with their location of the object in 3D. Because of the extrinsic calibration of the rgb-camera and the depth-camera, the initial object model can simply be created by assigning the interest points to the corresponding location in the organized point cloud.

In the following frames, the relative pose of the camera is detected. Therefore, the interest points which have been learned up to now are matched with interest points detected in the current image. Then the robust pose is computed using RANSAC [FB81] and the 3-point pose algorithm [HLON94]. Additionally, we use the depth data for early pruning of RANSAC hypotheses and immediately discard pose hypotheses which are not supported by the point cloud.

To decide when to learn a new view we project the 3D model points of the currently recognized view into the image and count the number of projected points which are supported by a detected interest point, i.e., where the distance to an interest point is less than a threshold t_{inl}. Consequently, we compute the confidence values

$$c_{i,j} = \frac{n_{support,i,j}}{n_{model,i.j}}, \tag{6.1}$$

$$c_i = \max_j c_{i,j} \tag{6.2}$$

where $n_{support,i,j}$ is the number of interest points that support 3D model points and $n_{model,i,j}$ stands for the number of points of the detected view j of the object model M_i.

In case the confidence sinks below 0.6, indicating that almost half of the interest points of the current view are no longer visible, a new view is added to the model. However, this only happens if at the same time the probability is high that the object was tracked correctly. This probability of *observed detection success* is introduced in Section 6.5.

For learning a new view, first the camera pose is refined with non-linear optimization using the sparse bundle adjustment implementation by Lourakis [LA09]. Then, the 3D point locations from the object are assigned to interest points. If the interest point can be matched to a model interest point it is linked to the according 3D point, otherwise the corresponding 3D location from the point cloud is transformed to object coordinates.

In summary, the object model consists of interest points which are organized in keyframes. To distinguish between object and background, as well as for the metric reconstruction of the interest points the point cloud from the rgb-depth sensor is used. Hence, learning is performed online while the object is explored and new viewpoints are visited. In case viewpoints visited before are passed again, recognized interest points are linked for a non-linear refinement of the structure. Furthermore, the completeness of the object model is represented by a spherical histogram. Figure 6.2 depicts the different learning steps. The probabilistic formalization for this is shown in Section 6.5. In the next section, we briefly describe how the online characteristic of model learning fits to the proposed recognizer.

6.3 Codebook Structure

For recognition of multiple objects, an efficient representation of the interest points is important. In [NS06], Nistér et al. have shown that vocabulary trees can be used for efficient indexing of large image databases. We adapted this approach for recognition of specific objects and use the resulting ranked list of found objects for a successive pose registration. Additionally, we develop a generative codebook based on incremental mean shift clustering. In the following we describe these two

methods to efficiently store the interest points and discuss their advantages and limitations.

6.3.1 Vocabulary Tree

The vocabulary tree defines a hierarchical quantization represented by prototype interest points, which is learned offline by unsupervised hierarchical k-means clustering. Instead of k defining the final number of clusters, k is the number of children of each node. Nistér has shown that for image retrieval, a large vocabulary of up to one million leaves improves the recognition performance. Once the tree is created, the interest points of a particular image can be matched efficiently by matching against prototype descriptors of nodes and parsing down the tree. To recognize an image, the task now is to compare how similar the paths are for the descriptors from a database image and a query image. To account for ambiguous descriptor assignment, weights w_i based on entropy are assigned to nodes. Thus, query q_i and database vectors d_i are defined by

$$q_i = n_i w_i \tag{6.3}$$
$$d_i = m_i w_i \tag{6.4}$$

where n_i and m_i are the number of interest points of the query image and the database image with a path through a particular node. The score for a database image

$$s(q,d) = \| \frac{q}{\|q\|} - \frac{d}{\|d\|} \| \tag{6.5}$$

is then defined by the normalized difference between the query and the database vector. Likewise, as proposed by Nistér and by Sivic et al. [SZ09] we use a *term frequency-inverse document frequency (TF-IDF)* scheme. Thus the weight w_i results in

$$w_i = \ln \frac{N}{N_i}. \tag{6.6}$$

N is the number of object views in the database and N_i stands for the number of object views in the database with at least one descriptor path through node i. Scoring can be efficiently implemented with an inverted file structure, where every node holds an inverted file and stores the ID of the object, the corresponding view in which it occurs and the term frequency m_i. Thus, according to the score $s(q,d)$, a ranked list of object view hypotheses is generated. Furthermore, we directly store the ID of the interest points of the models in the leaf nodes and thus, we additionally get tentative matches for the query interest points.

Once the vocabulary tree is created, interest points can be matched very fast resulting in a ranked list of object views of the database. Nistér has shown that the performance increases with an increasing number of leaf nodes up to one million leaves. The tree structure is learned with hierarchical k-means clustering. For

indexing of a movie, Sivic et al. [SZ09] learned a codebook using a random subset of images of the movie. But still, the question remains how to create an optimal tree, i.e. how to choose the number of levels and the number of branches of the nodes to avoid over-fitting? Is it possible to learn a vocabulary tree which covers the big variety of all possible objects? To avoid parameter tuning resulting from these questions, we propose a mean shift algorithm to sequentially extend the codebook while learning new object views.

6.3.2 Codebook Extension with Mean Shift Update

In contrast to the vocabulary tree described above, this codebook is created online while learning new views of objects. The main idea is to create a new codebook entry whenever an interest point does not match any prototype feature. Hence, the approach starts with an empty codebook which is sequentially extended while objects are being learned. The ideal case would be that parts, represented by interest points can uniquely be assigned to prototypes and in case a part is unknown it becomes a new prototype. In fact this works fine with a robust interest point descriptor, such as SIFT, but the drawback is, that descriptors are used as prototypes which are on the decision boundary and thus they are not representative for this part. Another possibility is to update the prototype with the mean of all descriptors assigned to the cluster. In practice, this tends to drift and thus it is possible that the updated prototype is not representative for the cluster anymore.

Instead, we propose to update the prototype descriptor for a codebook entry using a mean shift kernel which leads to a cluster mean

$$m(x) = \frac{\sum_{x_i \in C(x)} K(x_i - x) x_i}{\sum_{x_i \in C(x)} K(x_i - x)}, \tag{6.7}$$

where $C(x)$ stands for the interest points assigned to the codebook entry, and

$$K(x_i - x) = e^{c||x_i - x||^2} \tag{6.8}$$

represents the Gaussian kernel.

In summary, if a new view is learned, each interest point is matched with the codebook. If a match is found, it is assigned to the corresponding entry and the cluster center (the prototype descriptor) is updated using Equation 6.7. Likewise, as described in Section 6.3.1, for recognition a TF-IDF scheme is used to rank object views. There are two parameters which we empirically adjusted. First, the cut-off threshold for matching and secondly, the constant c which defines the Gaussian kernel. In fact, if the parameter c is set to a low value, the codebook gets noisy and if it is set to a high value, the codebook entries tend to drift. For the experiments we set $c = 0.2$, which seems to be a good compromise. Another constraint of this approach is that the codebook cannot be organized as tree and thus, the number of objects which can be handled at once is limited.

6.4 Object Recognition

Recognition of objects ends up in a similar procedure as learning of objects. First, the interest points of the query image have to be computed. Subsequently, the interest points are matched against the codebook and the scores for the query image and the object views in the database are computed, which results in a ranked list of matching object views. The scores are either computed using the vocabulary tree 6.3.1 or the codebook described in 6.3.2. According to the matched object views, each interest point of the query image has several tentative matches which are then used for geometric verification and to estimate the object pose. Algorithm 5 gives an overview of the different steps of our recognition pipeline. In

Algorithm 5 Object recognition pipeline

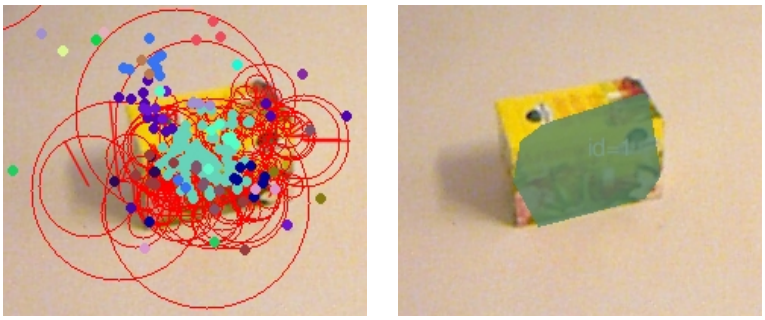
```
Detect interest points of query image
Match interest points with codebook
Query ranked list L of matching object views
for all Matching object views L do
  Let matches vote for object centers
  Cluster consistent votes
  for all Clusters do
    while Unmarked matches do
      Estimate object pose using LoRANSAC
      Mark inliers
    end while
  end for
end for
Compute probabilistic confidence value
Select recognized objects (non maxima suppression)
```

the following sections, details about voting for object hypotheses 6.4.1 and the final object pose estimation using the locally optimized RANSAC 6.4.2 are described.

6.4.1 Voting for Object Hypotheses

The final result of our recognition approach should include the location of the object with respect to the camera. The number of iterations required for RANSAC increases with the outlier rate. Because of the vocabulary based matching approach, however, the outlier rate can increase up to 90% and thus it is necessary to filter the matches. We integrated a voting scheme followed by mean shift clustering of the votes to group the most promising matches. In detail, the relative scale s_{rel} and the relative orientation θ_{rel} of the query interest point and the model interest point is used to vote for an object location

$$\mathbf{x}_v = \mathbf{R}\,\mathbf{x}_d\,s_{rel} + \mathbf{x}_q, \tag{6.9}$$



(a) (b)

Figure 6.3: Interest points (red circles) vote for object centers (coloured dots) which results in the correct pose indicated with the green overlay (left image).

where $\mathbf{R}$ stands for the rotation matrix computed from θ_{rel}, $\mathbf{x}_d$ is the location of the model interest point relative to the object center and $\mathbf{x}_q$ is the location of the interest point in the query image. Thus, votes of correct matches accumulate at the center of the object. To group the votes we use mean shift clustering (cf. [CM02]), which is an appropriate tool because of the unknown number of objects in the image.

Figure 6.3(a) shows detected interest points (red circles) and the corresponding votes for object centers (colored dots). It can be seen, that the correct mean shift cluster (cyan dots) accumulates most of the votes and thus leads to the most accurate pose indicated with the green overlay in Figure 6.3(b).

6.4.2 Locally Optimized RANSAC for Object Hypotheses Verification and 3D Pose Registration

RANSAC, introduced by Fishler and Bolles [FB81] in combination with the 3-point algorithm [HLON94], is the most widely used approach for perspective pose estimation in case at least three 3D/2D point correspondences are available. Thus, one possibility would be to cluster interest points, which vote for object centers and then use this approach to compute the pose and select consistent interest points. RANSAC finds all inliers and the corresponding model by repeatedly drawing random samples from the input set of data points. The number of inliers is typically not known and has to be estimated with the best model found up to the current iteration. It can be observed that the classical formulation runs much longer than theoretically predicted. This is due to the assumption that a model with parameters computed from an outlier-free sample of data points is consistent with all inliers. Because of the noisy measurements, this assumption rarely holds in practice. In [CMK03], Chum et al. propose a local optimization scheme applied to selected models. The idea is that an uncontaminated minimal sample is sufficiently close to the optimal solution and thus the local optimization step applied

to selected models leads to an approach which is close to the optimal solution.

We adapted the ideas by Chum for robust pose estimation and added a model upgrade step. Algorithm 6 shows our version for pose estimation. In contrast to

Algorithm 6 Locally optimized RANSAC with model upgrade

```
k ← 0, ε ← 3/N, I_max ← 0
while η = (1 − ε^3)^k ≥ η_0 do
  Sample 3 point pairs from interest point matches
  Compute the affine transformation from model image to query image A'
  Count number of explained interest points (inliers) I_a for A'
  if I_a > I_max then
    for z = 1 → Z do
      Sample 5 points from inlier set and compute pose P'
      Count number of explained interest points I_p for P'
      if I_p > I_max then
        I_max ← I_p
        ε ← I_max/N
        P ← P'
      end if
    end for
  end if
  k ← k + 1
end while
```

the standard implementation (described in Algorithm 1 for plane detection), we propose a weaker affine model computed from a minimal set of three points for the main loop. Only if a better explanation is found the pose is estimated with the DLT algorithm by sampling five points in an inner RANSAC loop. The pose that is more accurate is used to compute the termination of the main loop.

Hence, our implementation detects an almost outlier free subset of interest point matches very fast, which is then used to estimate an accurate pose. Figure 6.4 shows a typical test scenario where more than ten objects are recognized at the same time.

6.5 Self-Evaluation and Prediction

While being a valid and intuitively clear indicator for the quality of a detection outcome and lying in the range $[0, 1]$, the confidence defined in Equation 6.2 is not a probability. Making informed decisions based on such a value is difficult, especially when fusing detection results with other observations within a complete robotic system. Simple thresholding in order to force crisp outcomes of "found" and "not-found" leads to brittle systems. This section presents a sound probabilistic notion of detection success, as well as a measure of model completeness.

Figure 6.4: Typical scenario to test the object recognizer. The colored overlays are computed from the convex hull of the 3D points of the recognized object views, which are projected into the image.

6.5.1 Probability of Detection

Taking an approach from multi-view object detection by [LA06, HMM+10, MGL10] we define a generative detector model

$$\begin{aligned} p(c \,|\, o = true) \\ p(c \,|\, o = false) \end{aligned} \tag{6.10}$$

That is, we model the probability of detection confidence given that we have a true positive or a true negative, respectively. E.g., for the confidence as defined in Equation 6.2, it turns out that typically, a confidence of 0.4 already indicates almost absolute certainty of having a successful detection, while confidences under 0.1 tend to be false positives. We obtain training examples by transforming a virtual object model with 1000 random rotations, 252 scales and varying levels of artificial noise and blur. These virtual training examples can be created during the learning phase, though in our case they were generated beforehand for later comparison to ground truth. Figure 6.5 shows examples of a true positive (in green) and true negative (in red). The threshold for accepting a detection result as a positive example was set to $4\,cm$ of distance deviation, where we found that changing the threshold had no significant impact on results. Moreover, evaluation (Section 6.6) further justified this particular value.

Figure 6.6 shows examples of histograms of confidence values for true positives and true negatives for one of our objects. As can be seen, we obtain monomodal distributions (if we did not, this would indicate a badly chosen confidence measure) and we fit two Gaussians, as shown in Figure 6.7. Following Bayes rule we can then

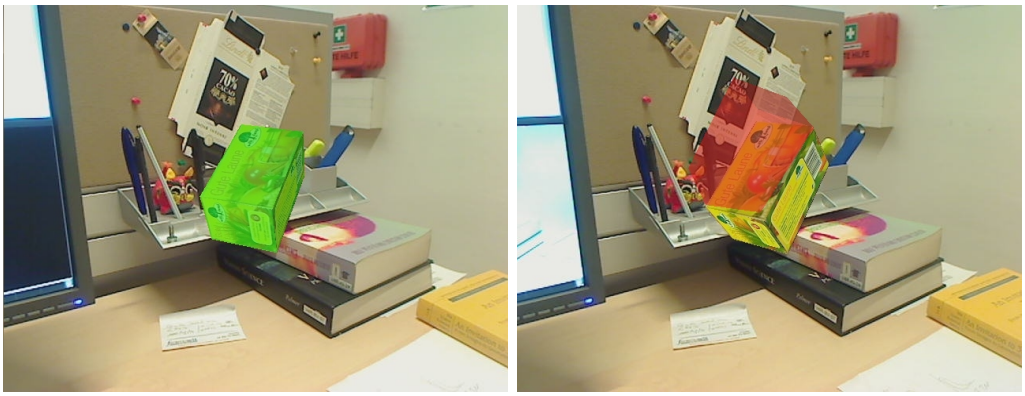


Figure 6.5: Rendered virtual views of an object for evaluating detection probability, with a true positive (left) and a true negative (right).

infer the posterior probability of a detected object

$$p(o \mid c) = \frac{p(c \mid o)\ p(o)}{p(c)} = \frac{p(c \mid o)\ p(o)}{\sum_{k \in \{t,f\}} p(c \mid o = k)} \tag{6.11}$$

where the prior for detecting an object could come from context knowledge such as current room category or bottom-up attention. Equation 6.11 is used during learning in Section 6.2 to decide whether tracking of the current view is still reliable. Furthermore, it returned to the user by the recognizer as a measure of *observed detection success*.

Figure 6.8 shows the posterior probability of having successfully detected an object given detector confidence for the training data of Figure 6.6.

6.5.2 Representing Completeness

In order to obtain a quantitative measure of completeness, we need to know the probability of detecting the object given the views learned so far. To this end, we learn the probability of detecting an object view for a given out of plane rotation θ. Again we start with labeled training data, to obtain

$$\begin{aligned} p(\theta \mid o_j = true) \\ p(\theta \mid o_j = false) \end{aligned} \tag{6.12}$$

where o_j is the j-th view of the object, and use Bayes rule to get the *predicted detection success* for a given rotation

$$p(o_j \mid \theta) = \frac{p(\theta \mid o_j)\ p(o_j)}{p(\theta)} = \frac{p(\theta \mid o_j)\ p(o_j)}{\sum_{k \in \{t,f\}} p(\theta | o_j = k)} \tag{6.13}$$

Figure 6.9 shows that recognition probability for a single learned view drops to 50% around 22°, which is about the expected value for the feature descriptor used.

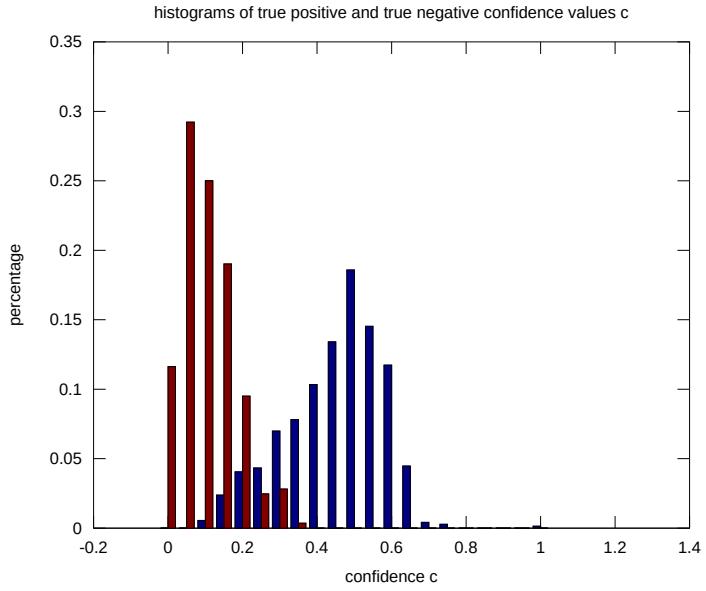


Figure 6.6: Histograms for true positives (blue) and true negatives (red)

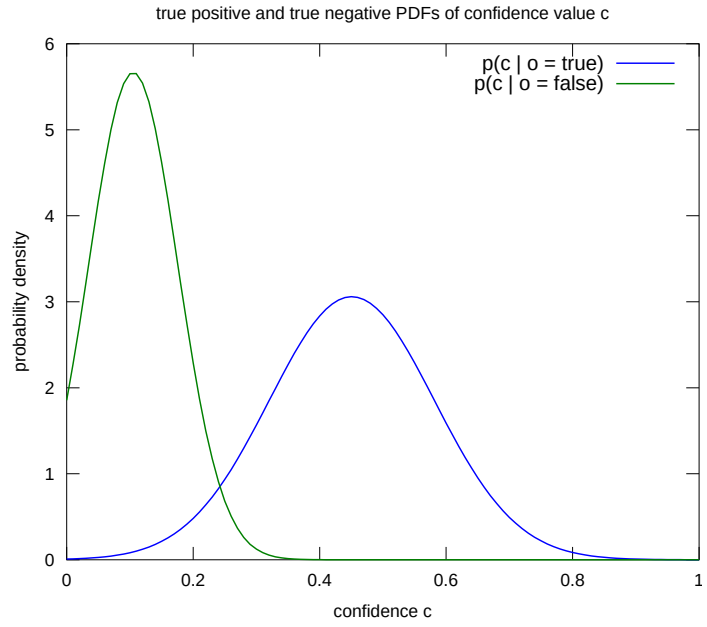


Figure 6.7: Estimated Gaussian PDFs of confidence for true positives (blue) and true negatives (green)

Note that the same procedure applies to varying scale instead of rotation, and in fact other environmental factors, such as lighting, which affect recognition. However, it is only rotation and scale that we can actively influence.

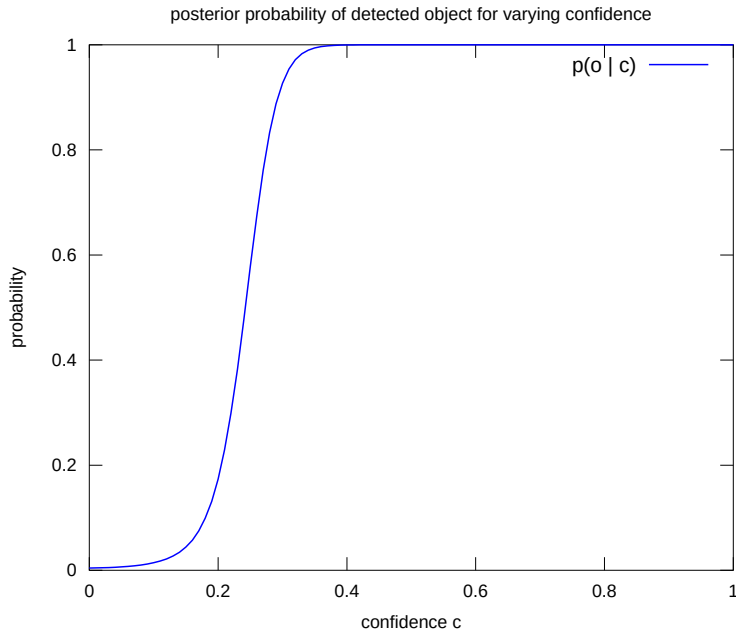


Figure 6.8: *Observed detection success*: Posterior probability of detected object for given confidence

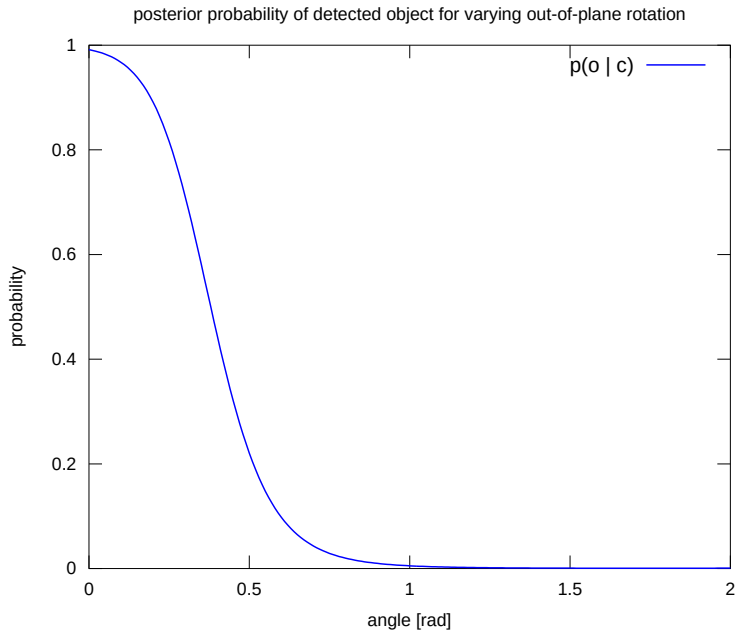


Figure 6.9: *Predicted detection success*: Posterior probability of detected object for given rotation, for a single learned view.

To arrive at a measure of *model completeness*, we take the expected detection probability over all learned views (note that we actually have to vary over two angles - azimuth and inclination, for notational simplicity, however, we use a single angle θ)

$$\hat{p}(o) = \sum_{\theta,j} p(o_j \,|\, \theta) p(\theta) \tag{6.14}$$

where $p(\theta)$ takes into account that certain views are less likely than others and thus are possibly not even learned (such as the underside of an object).

6.5.3 Taking Action

The above view-based representation with associated detection probabilities and measure of completeness allows us to inform the system about the next learning steps.

Starting with a single learned view we can choose a rotation for which the probability of detecting the object in the new view is high enough to allow to associate the two views with the same object model but low enough to warrant learning the additional view. Note that our approach relies on tracking the previous most probable view in order to associate newly added views with a correct object pose. Therefore, learning can only take place at the fringe of currently available knowledge, i.e. the border between "bright" and "dark" areas in Figure 6.1(c).

To arrive at a utility to drive exploration, we need a gain (higher probability of detection after acquiring a new training view) as well as costs associated with reaching that view and attempting a learning step. Costs are measured in run-time and are composed of path planning for either moving the arm mounted camera or the whole platform (e.g. $0.5\,s$), executing the planned movement (several seconds) and attempting the learning step. The learning step consists of first detecting the object using the views learned so far (e.g. $0.5\,s$), which will succeed with a probability given by Equation 6.13 and adding the new view. (Note that actually, new views will only be added if they are sufficiently different to previous views, which is determined inside the learner using various metrics. So there is a (small) probability that a new view might not be added. But we can safely ignore these cases)

We define gain as the expected decrease in $\hat{p}(o = false)$ after learning the new $(n+1)$-th view:

$$g \;=\; \hat{p}_n(o = false) - \hat{p}_{n+1}(o = false) \tag{6.15}$$

with

$$\hat{p}_n(o = false) \;=\; 1 - \sum_{\theta} \sum_{j=1}^{n} p(o_j \,|\, \theta) p(\theta) \tag{6.16}$$

$$\hat{p}_{n+1}(o = false) \;=\; 1 - \sum_{\theta} \sum_{j=1}^{n+1} p(o_j \,|\, \theta) p(\theta) \tag{6.17}$$

That is, we tentatively add the (empty) future view to our model together with its *predicted detection success* mode (which we assume to be the same for all views) and calculate the increase in detection probability.

6.6 Tests and Evaluation

In this section, we evaluate the measures for *observed detection success* (Equation 6.11) and *predicted detection success* (Equation 6.13). Note that it is not our goal here to evaluate recognition performance. Nor can we at this point evaluate a complete system including *model completeness*, where learning is guided using our measures. Before doing so, we first need to establish that these measures learned from virtual training data actually match ground truth obtained from real images.

To learn our measures, we rendered five virtual objects into background images of realistic scenes with randomly chosen poses (see Figure 6.5). We selected a random object pose of each object and trained an initial recognition model. Then we rotated the virtual object to 1000 randomly chosen poses with different viewing angles from 0..60° and tried to recognize the object. Furthermore, we changed the scale of the object from 0.5 to 2 times the learning distance, which results in 252 images for each of the 5 objects. In order to simulate realistic conditions, we added Gaussian noise and blur at 8 levels, with $\sigma_{noise} = 0..16$ grey levels (out of 255) and blur with $\sigma_{blur} = 0..1.6$ pixels. After learning the measures for *observed detection success* and *predicted detection success*, we evaluated them on four real sequences, where we rigidly attached a checker board pattern to the objects to create the ground truth pose data. The pose of the pattern is estimated with a standard DLT algorithm, followed by a non-linear optimization of the pose using the sparse bundle adjustment implementation by Lourakis [LA09]. In total, we used about 50000 virtually rendered images for learning and 461 real images for testing.

Figure 6.10 shows the comparison of *observed detection success* learned from virtual vs. real ground truth training data for one of the objects. As can be seen, increasing noise levels shift confidence (as is to be expected), and the true curve lies within the band defined by the various virtual curves. The real curve, however, has a steeper slope indicating that training from real data allows better discrimination between positives and negatives.

Figure 6.11 shows the comparison of *observed detection success* for different objects. As can be seen, the curves differ considerably indicating that different objects pose varying difficulties for recognition. This further indicates that *observed detection success* should be learned for each object individually.

Figure 6.12 compares *predicted detection success* for virtual (with varying noise level) and real training data for the same object as in Figure 6.10. We can see that the curves more or less intersect around 22° where probability drops to 50%, and that again training from real data allows for better discrimination between positives and negatives.

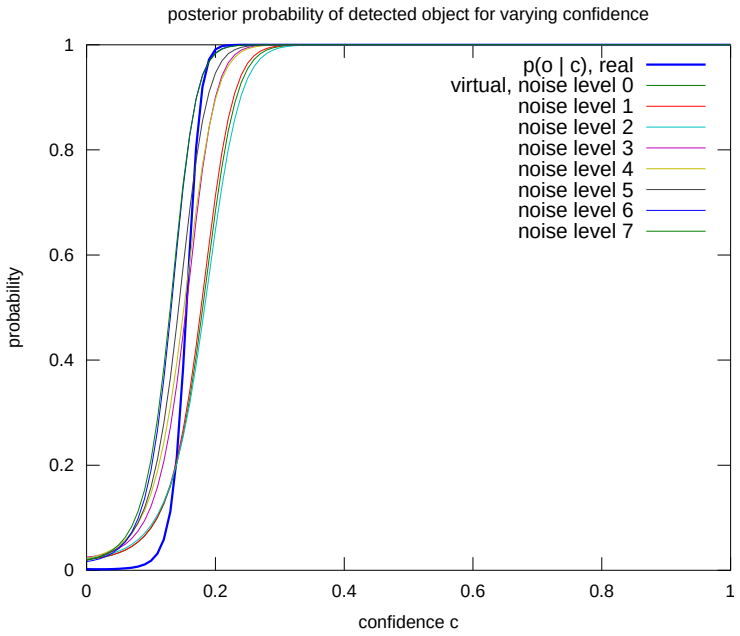


Figure 6.10: Comparison of *observed detection success* learned from virtual training data including various levels of noise (thin lines) and learned from real ground truth data (thick line).

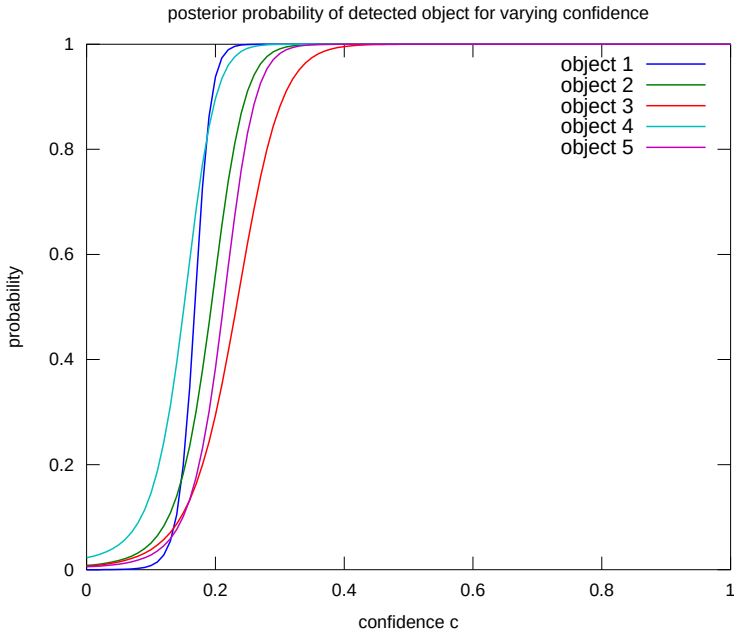


Figure 6.11: Comparison of *observed detection success* for different objects.

Figure 6.13 shows the accuracy of pose registration of the proposed approach for the real image sequences. As expected, it can be seen that the depth error

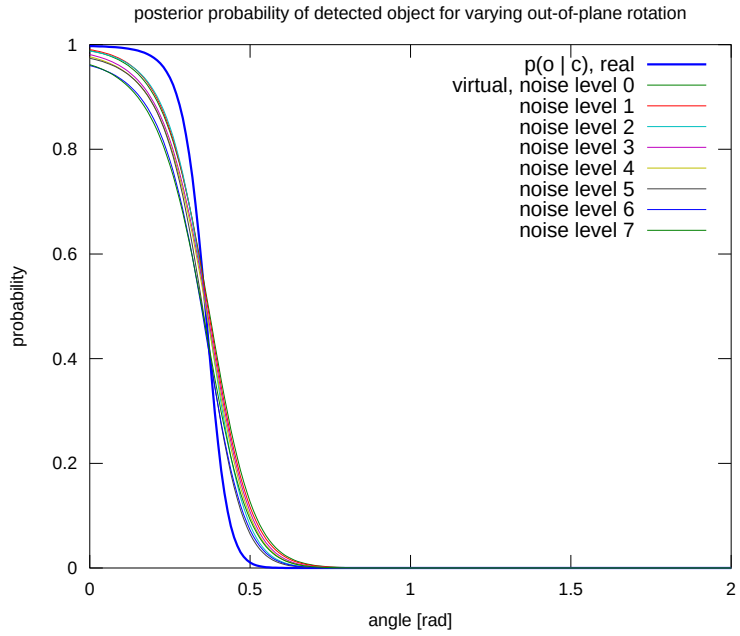


Figure 6.12: Comparison of *predicted detection success* learned from virtual training data including various levels of noise (thin lines) and learned from real ground truth data (thick line).

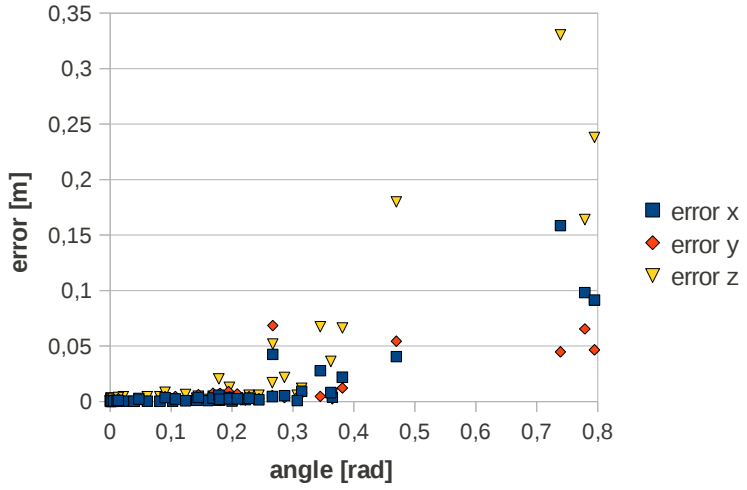


Figure 6.13: Accuracy of the pose of the recognized objects for real data. The evaluation is done in camera coordinates with the z-axis pointing towards the object. Hence, *error x* and *error y* represent the deviation from ground truth within the image plane and *error z* is the depth error.

is slightly higher than the error within the image plane. Furthermore, the error increases with an increasing rotation in depth θ and passes $4\,cm$ at about 20°, after

which performance rapidly degrades. This is in accordance with Figure 6.12 which predicts detection success to drop below 50% around 22°. Motivated by the results of the accuracy evaluation, we use this threshold of $4\,cm$ to distinguish between true positive (tp) and false positive (fp) in Section 6.5.1.

6.7 Discussion

This chapter presented an online learning approach for object recognition targeted to a robotics exploration framework. To overcome the exploration-exploitation dilemma, i.e., when to stop learning in order to complete the task, we developed a probabilistic model. The model is designed for self-assessment of observed detection success, as well as for the prediction of detection success. This leads to a measure for the completeness of learned object models. The learning and recognition approach combines different sensor modalities, namely the rgb-image and the point cloud acquired with the rgb-depth sensor Kinect to build scale corrected object models and to improve pose registration. We propose mean shift clustering to incrementally extend the codebook and compare it to the recently proposed vocabulary tree for an efficient representation of interest points. Using the proposed incremental mean shift extension of the codebook, online learning of objects can start without any a priori knowledge, i.e., without a trained codebook or vocabulary tree.

The main contribution of this chapter lies in proposing three learned probabilistic measures for *observed detection success*, *predicted detection success* and *model completeness*. The evaluation has shown that learning these measures from virtual training data, which can be generated from partially learned object models during object learning, shows results comparable to learning from real ground truth data (which is of course not available during learning). These measures allow the robot to represent its knowledge of objects as well as the limit of its knowledge in a probabilistic manner compatible with probabilistic reasoning mechanisms elsewhere in the system and thus to plan for actions to extend its knowledge.

Chapter 7

Summary and Discussion

Learning and interaction are basic skills for a cognitive robotic system. A robot must be able to expand its knowledge in a human tutor-driven way or even learn entirely on its own. Hence, the real challenge is to understand and build cognitive systems that are able to handle situations unforeseen by their designers. The system must be capable of self-extension, that is, it must be able to learn, represent what it does not know, reason about what it can learn and how to act so as to learn it, execute those actions and then learn from the resulting experience. We argue that for a cognitive robot those abilities, namely *explore, explain and extend* are not only manifested in a high level reasoning component – the brain of a robot –, but need to be implemented at each level of the system based on an intelligent perception system.

In this thesis we developed a perception system for cognitive robots. In Chapter 1, we introduced a concept for visual perception that enables the robot to explore human environments. The hypothesis is that planar surface patches are the key element of a suitable object representation. In contrast to sparse point clouds, typically used for camera localization and robot navigation, planar patches have an orientation, a size and a shape. Hence, it is possible to compute approaching vectors for interaction, and the patches support reasoning about contact points for touching or grasping objects. The following chapters describe how to explain the environment in terms of piecewise planar surfaces, how to create initial object hypotheses and how to verify these object hypotheses by interaction. Furthermore, a probabilistic completeness representation is introduced which allows to reason about unseen object views and where to extend the models.

In detail, Chapter 3 developed a new approach to detect multiple planes in image pairs. The theoretical framework for this is grounded in model selection and Minimal Description Length (MDL). Model selection is embedded in an iterative scheme, where existing planes have to compete with newly generated hypotheses. We have shown that this framework allows to limit the search space, which leads to a fast explanation of the entire image in terms of piecewise planar surfaces.

Next, in Chapter 4 we examined how to robustify segmentation and tracking

with reasoning in image sequences. Typically, segmentation in single images or image pairs is inaccurate and results are under- or over-segmented. Furthermore, trackers with online update need to handle the drift which comes from the tracking inaccuracy. We have shown that the reasoning system is able to keep track of objects under partial and full occlusion, as well as to handle the drift.

In Chapter 5, we investigated the reconstruction and the merging of planes to individual objects. Therefore, consistent with the MDL-formulation of the previous chapter, a pseudo-likelihood has been developed which combines motion, colour and the spatial arrangement of planes. The presented approach is related to the problem of Multi-body Structure-and-Motion (MSaM). Instead of direct reconstruction from of interest points, we first cluster them to planar patches and then reconstruct patches with consistent motion. We have shown that the color and proximity of patches, which are currently not visible but have been seen before, can be used to separate the foreground object patches from the background.

Finally, in Chapter 6 we presented an approach for learning models for object recognition. We developed a probabilistic model to evaluate the observed detection success as well as the prediction of detection success, which leads us to a measure for the completeness of learned object models. The goal of this chapter is to overcome the exploration-exploitation dilemma and to provide measures for the cognitive robot to decide when to continue learning and when to stop learning and use the knowledge acquired so far.

In summary, this thesis developed an object model that enables a robot to interact with its environment. It is based on piecewise planar surface patches detected in image pairs from consistently moving interest points. The planar patches are tracked and reconstructed. They build the initial object hypotheses and provide contact points which are linked to affordances for the robot. Simple interactions by the robot lead to the final object model. For example, if the robot accidentally pushes several objects, different motions will occur and they will be modelled as different items. State-of-the-art interest points, efficiently stored in a codebook, are used to recognize the reappearance of patches. The codebook is incrementally extended if never seen interest points appear. Chapters 3 – 5 focused on a monocular camera system and dynamically changing scenes to reconstruct objects. The last chapter introduced the recently developed rgb-depth sensor *Kinect* for recognition and to evaluate the probabilistic completeness representation. RGB-depth sensors provide depth information also if the scene is static and untextured. Hence, this sensor provides the opportunity for our curious robot to detect planar parts without the need to change the view point. This is already a point that will be investigated in the future work.

7.1 Discussion

In this thesis, we developed a perception system for cognitive robotic exploration. This carries advantages and limits that we discuss here and investigate how to proceed with future work.

The basic parts of our object model are planar patches. Thus, the first question that comes up is how to handle round or free formed surfaces. Up to now, the results strongly depend on the motion between image pairs. We detect planes using random samples and model selection and a fixed threshold to distinguish between inliers and outliers. Hence, if the motion is small, the approximation is rather weak and, for example, a cylinder will be approximated with only a few patches. This immediately leads to the next question, namely: How robust is the approach in case there is large motion and the images are rather blurry? We addressed this issue in two ways. First, we have shown that reasoning improves segmentation and tracking. But in our understanding, it is even more important that the system self-assesses the current status and that it is able to predict what comes next. We addressed this with the probabilistic measure about the uncertainty and the model to predict the result if the viewpoint is changed.

For robotic applications it is important that results are available within a desired time. In Chapter 3, we discussed how to speed up plane detection. Results show that incremental model selection based plane detection has a superior performance. The reason is that the incremental approach allows us to guide random sampling to unexplained regions, while new samples still have to compete with plane hypotheses from the last iteration. The real-time issue also arises in Chapter 4 and Chapter 5, where image sequences are represented within a hypotheses graph. To keep the hypotheses graph manageable, early pruning of weak hypotheses is necessary. Scaling in general is an open issue. Until now, we tested our system with table top experiments. The next step for future work is to implement the system on a real robot with the ability to drive in a whole flat or house. Although we tested object modelling by manually pushing planar patches, experiments with a real robot are left for future work. Another question is whether it is possible to describe more complex affordances and functions. For example, a future robot butler should be able to prepare and serve a cup of tea. Therefore it is necessary that the robot fills water into a cup. Currently available implementations use preprogrammed scripts, but in the future, models need to be developed so that robots can learn functions (e.g. *fillable in*) on their own.

Beside conceptual issues there is a lot of space for improvements. In the following paragraph we mention a few examples. We propose a piecewise planar object model. The developed algorithms are based on interest points. This implies that the objects need to be textured. In case of sparsely textured surface parts, details of the scene will be missed. This drawback can partially be overcome with a multi-label segmentation using a Markov Random Field (MRF) optimization and graph-cuts, e.g., such as proposed by Sudipta et al. [SSS09] and Micusik et al. [MK09] or

by an optical flow based optimization proposed by Newcombe [ND10]. But still, if parts of the scene are single colored, they cannot be detected. To some extent, these challenging environments can be handled with a recently developed rgb-depth sensor which we already used to test our recognition system. The additional depth information can also be used to improve robustness.

Appendix A

Homography Estimation

The homography H is the projective transformation of points located on a plane from one image to another one. The homography transformation of a 2D image point $\mathbf{x} = [x\, y\, w]^T = [x/w\ y/w\ 1]^T$ to its counterpart $\mathbf{x}'$ can be written as follows:

$$c\,\mathbf{x}' = \mathbf{H}\,\mathbf{x} \tag{7.1}$$

$$c \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7.2}$$

The simplest way to estimate the parameters of the homography matrix is the *Direct Linear Transform* (DLT). With a minimum of four point pairs Equation 7.2 can be reformulated to:

$$\mathbf{A}\,\mathbf{h} = 0 \tag{7.3}$$

$$\begin{bmatrix} x'_1 & y'_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -x_1y'_1 & -x_1 \\ 0 & 0 & 0 & x'_1 & y'_1 & 1 & -y_1x'_1 & -y_1y'_1 & -y_1 \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \\ & & & & \cdots & & & & \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \tag{7.4}$$

The solution to this equation can be computed with the *Singular Value Decomposition* (SVD) of $\mathbf{A}$, where the parameters of $\mathbf{H}$ are given by the singular vector corresponding to the smallest singular value. Hence, the solution for

$$\mathbf{h} = \frac{v_{19}, \cdots, v_{99}}{v_{99}} \tag{7.5}$$

is given by the last column of V^T, normalized by its last entry of the SVD of

$$\mathbf{A} = \mathrm{UDV}^T = \mathrm{U} \begin{bmatrix} d_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & s_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & s_{99} \end{bmatrix} \tag{7.6}$$

The solution to Equation 7.6 minimizes the least-squares error for the parameters of the homography matrix $\mathbf{H}$.

More details, including a more elaborate estimation based on nonlinear optimizations can be found in [Har08] and there is also a c-implementation by Lourakis [Lou06] is available online.

Homography Decomposition

According to [Ma04], to decompose the homography matrix, it has to be normalized with the second singular value

$$\mathbf{H_n} = \frac{\mathbf{H_{cam}}}{\sigma_2} \tag{7.7}$$

and decomposed to

$$\mathbf{H_n}^T\mathbf{H_n} = \mathbf{V}\,\mathbf{S}\,\mathbf{V}^T, \tag{7.8}$$

which results in four solutions:

$$\mathbf{R_{1,2}} = \mathbf{W_1}\mathbf{U_1}^T \tag{7.9}$$

$$\mathbf{n_{1,2}} = \pm\hat{\mathbf{v}}_\mathbf{2}\mathbf{u_1} \tag{7.10}$$

$$\frac{1}{d}\mathbf{T_{1,2}} = \pm(\mathbf{H_n} - \mathbf{R_1})\,\mathbf{n_1} \tag{7.11}$$

and

$$\mathbf{R_{3,4}} = \mathbf{W_2}\mathbf{U_2}^T \tag{7.12}$$

$$\mathbf{n_{3,4}} = \pm\hat{\mathbf{v}}_\mathbf{2}\mathbf{u_2} \tag{7.13}$$

$$\frac{1}{d}\mathbf{T_{3,4}} = \pm(\mathbf{H_n} - \mathbf{R_2})\,\mathbf{n_1} \tag{7.14}$$

where

$$\mathbf{V} = [\mathbf{v_1}\ \mathbf{v_2}\ \mathbf{v_3}], \tag{7.15}$$

$$\Sigma = diag(\sigma_1^2\ \sigma_2^2\ \sigma_3^2\ ) \tag{7.16}$$

and

$$\mathbf{u_{1,2}} = \frac{\sqrt{1-\sigma_3^2}\,\mathbf{v_1} \pm \sqrt{\sigma_1^2-1}\,\mathbf{v_3}}{\sqrt{\sigma_1^2-\sigma_3^2}}. \tag{7.17}$$

Furthermore $\mathbf{U_{1,2}}$ and $\mathbf{W_{1,2}}$ stand for

$$\mathbf{U_{1,2}} = [\mathbf{v_2}\ \ \mathbf{u_{1,2}}\ \ \hat{\mathbf{v}}_\mathbf{2}\mathbf{u_{1,2}}] \tag{7.18}$$

and

$$\mathbf{W_{1,2}} = [\mathbf{H_n}\mathbf{v_2}\ \ \mathbf{H_n}\mathbf{u_{1,2}}\ \ \hat{\mathbf{H_n}}\mathbf{v_2}\mathbf{H_n}\mathbf{u_{1,2}}]. \tag{7.19}$$

The correct solution can be found using the physical constraints described in Section 5.2.

Bibliography

[AA06] Martin Antenreiter and Peter Auer. A reasoning system to track movements of totally occluded objects. In *2nd International Cognitive Vision Workshop (ICVW), Graz, Austria*, 2006.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45:891–923, November 1998.

[APVA09] Martin Antenreiter, Johann Prankl, Markus Vincze, and Peter Auer. Using a spatio-temporal reasoning system to improve object models on the fly. In *33rd Workshop of the Austrian Association for Pattern Recognition (AAPR 2009)*, 2009.

[AWB88] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1:333–356, 1988. 10.1007/BF00133571.

[Baj88] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966 –1005, August 1988.

[Bal91] Dana H. Ballard. Animate vision. *Artif. Intell.*, 48:57–86, February 1991.

[BETG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[BLST02] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 1014 – 1019 vol.1, 2002.

[BMCH04] Brandon Bennett, Derek R. Magee, Anthony G. Cohn, and David C. Hogg. Using spatio-temporal continuity constraints to enhance visual tracking of moving objects. In *ECAI-04*, pages 922–926, 2004.

[BT98] F. Brémond and M. Thonnat. Tracking multiple non-rigid objects in video sequences. *IEEE Transaction on Circuits and Systems for Video Technology Journal*, 8(5), September 1998.

[BTZ96] Paul A. Beardsley, Philip H. S. Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *Proceedings of the 4th European Conference on Computer Vision-Volume II - Volume II*, ECCV '96, pages 683–695, London, UK, 1996. Springer-Verlag.

[CK95] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. pages 1071 –1076, jun. 1995.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:603–619, May 2002.

[CMK03] Ondej Chum, Ji Matas, and Josef Kittler. Locally optimized ransac. In Bernd Michaelis and Gerald Krell, editors, *Pattern Recognition*, volume 2781 of *Lecture Notes in Computer Science*, pages 236–243. Springer Berlin / Heidelberg, 2003.

[CPM00] Rita Cucchiara, Massimo Piccardi, and Paola Mello. Image analysis and rule-based reasoning for a traffic monitoring system. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):119–130, June 2000.

[CWS09] T.-J. Chin, H. Wang, and D. Suter. Robust fitting of multiple structures: The statistical learning approach. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[ED01] Ahmed M. Elgammal and Larry S. Davis. Probabilistic framework for segmenting people under occlusion. In *ICCV*, pages 145–152, 2001.

[EKH05] Staffan Ekvall, Danica Kragic, and Frank Hoffmann. Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. *Image and Vision Computing*, 23(11):943–955, 2005.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[FH06] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *Int. J. Comput. Vision*, 70:41–54, October 2006.

[FLP01] Olivier Faugeras, Quang-Tuan Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.

[FPK+06] Gerald Fritz, Lucas Paletta, Manish Kumar, Georg Dorffner, Ralph Breithaupt, and Erich Rome. Visual learning of affordance based cues. In *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 52–64. Springer Berlin / Heidelberg, 2006.

[FPZ07] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *Int. J. Comput. Vision*, 71:273–303, March 2007.

[FRC09] Simone Frintrop, Erich Rome, and Henrik Christensen. Computational Visual Attention Systems and their Cognitive Foundations: A Survey. *ACM Transactions on Applied Perception*, 2009.

[FSB06] Friedrich Fraundorfer, Konrad Schindler, and Horst Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, 24(4):395 – 406, 2006.

[FSB10] D. Fouhey, D. Scharstein, and A. Briggs. Multiple plane detection in image pairs using j-linkage. In *20th International Conference on Pattern Recognition (ICPR 2010)*, To appear 2010.

[FZ00] Andrew W. Fitzgibbon and Andrew Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. In *In European Conference on Computer Vision*, pages 891–906. Springer-Verlag, 2000.

[GGB07] Michael Grabner, Helmut Grabner, and Horst Bischof. Learning Features for Tracking / Tracking via Discriminative Online Learning of Local Features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[Gib86] James J. Gibson. *The Ecological Approach To Visual Perception*. Lawrence Erlbaum Associates, new edition edition, September 1986.

[GL06] Iryna Gordon and David G Lowe. What and where: 3D object recognition with accurate pose. In J. Ponce, M. Hebert, Schmid. C., and A. Zisserman, editors, *Toward Category-Level Object Recognition*, chapter What and where: 3D object recognition with accurate pose, pages 67–82. Springer, 2006.

[Gre04] Gustaf Gredebäck. *Infants Knowledge of Occluded Objects: Evidence of Early Spatiotemporal Representation.* Number Dissertation, ISBN 91-554-5898-X. Acta Universitatis Upsaliensis; Faculty of Social Sciences, 2004.

[Ham06] Fred H. Hamker. Modeling feature-based attention as an active top-down inference process. *Biosystems*, 86(1-3):91–99, 2006. Brain, Vision and Artificial Intelligence (BVAi) 2005: Papers from a Symposium on Brain Basics and Natural Vision.

[Har08] A. Hartley, R.; Zisserman. *Multiple View Geometry in computer vision.* Cambridge University Press, 2008.

[HB04] J. Hawkins and S. Blakeslee. *On Intelligence.* Times Books, New York, NY, USA, 2004.

[HBN07] S. Hinterstoisser, S. Benhimane, and N. Navab. N3m: Natural 3d markers for real-time object detection and pose estimation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –7, October 2007.

[HE05] Yan Huang and Irfan Essa. Tracking multiple objects through occlusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, volume 2, pages 1051–1058, San Diego, CA, USA, June 2005.

[HGJ05] Stephen Hart, Roderic Grupen, and David Jensen. A relational representation for procedural task knowledge. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 3*, pages 1280–1285. AAAI Press, 2005.

[HJL+89] R.M. Haralick, H. Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(6):1426 –1446, nov. 1989.

[HLON94] Bert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nlle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13:331–356, 1994.

[HMM+10] Scott Helmer, David Meger, Marius Muja, James J. Little, and David G. Lowe. Multiple Viewpoint Recognition and Localization. In *Proceedings of the Asian Computer Vision Conference*, Queenstown, New Zealand, 2010.

[HS88] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[IKN98] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254 –1259, nov 1998.

[KK04] Y. Kanazawa and H. Kawakami. Detection of planar regions with uncalibrated stereo using distributions of feature points. In *British Machine Vision Conference (BMVC)*, 2004.

[KM08] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *Proc. 10th European Conference on Computer Vision (ECCV'08)*, pages 802–815, Marseille, October 2008.

[Kro88] Eric Krotkov. Focusing. *International Journal of Computer Vision*, 1:223–237, 1988.

[LA06] Catherine Laporte and Tal Arbel. Efficient Discriminant Viewpoint Selection for Active Bayesian Recognition. *International Journal of Computer Vision*, 68(3):267–287, 2006.

[LA09] M.I. A. Lourakis and A.A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[LAO02] M.I.A. Lourakis, A.A. Argyros, and S.C. Orphanoudakis. Detecting planes in an uncalibrated image pair. In *British Machine Vision Conference (BMVC)*, 2002.

[LF96] Quan-Tuan Luong and Olivier D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17:43–75, 1996.

[LGB95] Aleš Leonardis, Alok Gupta, and Ruzena Bajcsy. Segmentation of range images as the search for geometric parametric models. *Int. J. Comput. Vision*, 14(3):253–277, 1995.

[LLS08] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *Int. J. Comput. Vision*, 77(1-3):259–289, 2008.

[LNGPS05] G. Lopez Nicolas, J.J. Guerrero, O.A. Pellejero, and C. Sagues. Computing homographies from three lines or points in an image pair. In *CIAP 2005*, pages 446–453, 2005.

[Lou06] M.I.A. Lourakis. homest: A c/c++ library for robust, non-linear homography estimation. [web page] http://www.ics.forth.gr/ lourakis/homest/, Jul. 2006. [Accessed on 20 Jul. 2006.].

[Low87] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, pages 355–395, 1987.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[Ma04] S.; Kosecka J.; Sastry S.S. Ma, Y.; Soatto. *An Invitation to 3-D Vision - From Images to Geometric Models*. Springer, 2004.

[MA09] Ajay K. Mishra and Yiannis Aloimonos. Active segmentation. *I. J. Humanoid Robotics*, 6(3):361–386, 2009.

[Man04] Jean Matter Mandler. *The Foundations of Mind: Origins of Conceptual Thought*. NY: Oxford University Press, New York, 2004.

[MF03] Giorgio Metta and Paul Fitzpatrick. Better vision through manipulation. *Adaptive Behavior*, 11:109–128, 2003.

[MGL10] D. Meger, A. Gupta, and J.J. Little. Viewpoint detection models for sequential embodied object category recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5055–5061, Anchorage, AK, 2010.

[MH90] Takashi Matsuyama and V. S. Hwang. *SIGMA: A Knowledge-Based Aerial Image Understanding System*. Perseus Publishing, 1990.

[MJD+00] 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, October 2000.

[MK04] J. Modayil and B. Kuipers. Bootstrap learning for object discovery. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004.

[MK06] Joseph Modayil and Benjamin Kuipers. Autonomous shape model learning for object localization and recognition. In *IEEE International Conference on Robotics and Automaton (ICRA-06)*, pages 2991–2996, 2006.

[MK07] Joseph Modayil and Benjamin Kuipers. Autonomous development of a grounded object ontology by a learning robot. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1095–1101. AAAI Press, 2007.

[MK08] Joseph Modayil and Benjamin Kuipers. The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems*, 56(11):879–890, 2008. Semantic Knowledge in Robotics.

[MK09] B. Micusik and J. Kosecka. Piecewise planar city 3d modeling from street view panoramic sequences. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:2906–2912, 2009.

[ML09] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[MLBSV08] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *Robotics, IEEE Transactions on*, 24(1):15 –26, 2008.

[MMFF03] Giorgio Metta, Giorgio Metta, Paul Fitzpatrick, and Paul Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11:109–128, 2003.

[MSC+09] C. Mei, G. Sibley, M. Cummins, P. Newman, and I.D. Reid. A constant-time efficient stereo slam system. In *British Machine Vision Conference (BMVC)*, 2009.

[MTN+02] D.R. Myatt, P.H.S. Torr, S.J. Nasuto, J.M. Bishop, and R. Craddock. Napsac: High noise, high dimensional robust estimation - it's in the bag. In *British Machine Vision Conference (BMVC)*, 2002.

[ND10] Richard A. Newcombe and Andrew J. Davison. Live dense reconstruction with a single moving camera. 2010.

[Nis04] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:756–777, June 2004.

[NNB04] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. volume 1, pages I–652 – I–659 Vol.1, june 2004.

[NS06] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006.

[oC10] Committee of CogRob08. Website of the bi-annual workshop on cognitive robotics. [web page] http://www.cse.yorku.ca/cogrob08/index.html, Oct. 2010. [Accessed on 25. Oct. 2010.].

[OFL07] Mustafa Özuysal, Pascal Fua, and Vincent Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[OLFF06] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature Harvesting for Tracking-by-Detection. In *European Conference on Computer Vision*, volume 3953, pages 592–605, 2006.

[OSG10] Kemal Egemen Ozden, Konrad Schindler, and Luc Van Gool. Multibody structure-from-motion in practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1134–1141, 2010.

[Oxh] Houses data sets published by the visual geometry group at the university of oxford. [web page] http://www.robots.ox.ac.uk/ vgg/data/. [Accessed on 6 Jul. 2011.].

[PAAV09] Johann Prankl, Martin Antenreiter, Peter Auer, and Markus Vincze. Consistent interpretation of image sequences to improve object models on the fly. In *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, ICVS '09, pages 384–393, Berlin, Heidelberg, 2009. Springer-Verlag.

[Pal99] S.E. Palmer. *Vision Sciene - Photons to Phenomenology.* The MIT Press, 1999.

[PB08] S.J. Pundlik and S.T. Birchfield. Real-time motion segmentation of sparse feature points at any speed. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 38(3):731–742, June 2008.

[PCI+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[PP06] J. Piazzi and D. Prattichizzo. Plane detection with stereo images. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference*, pages 922 –927, may 2006.

[PRD09] Qi Pan, Gerhard Reitmayr, and Tom Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *Proc. British Machine Vision Conference (BMVC)*, 2009.

[Pup13] Pupils of A. Meinong, editor. *A. Meinong's Gesammelte Abhandlungen (Collected Works)*, volume 2: Abhandlungen zur Erkenntnistheorie und Gegenstandstheorie (Works on epistemology and object-theory). J.A. Barth, Leipzig, Germany, 1913. Reprinted in: Alexius

Meinong Gesamtausgabe (A. Meinong complete edition), Volume 2, Graz (1971).

[PVGV+04] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59:207–232, September 2004.

[Pyl01] Z. Pylyshyn. Visual indexes, preconceptual objects, and situated vision. *Cognition*, 80:127–158, 2001.

[PZLV10] Johann Prankl, Michael Zillich, Bastian Leibe, and Markus Vincze. Incremental model selection for detection and tracking of planar surfaces. In *Proceedings of the British Machine Vision Conference*, pages 87.1–87.12. BMVA Press, 2010. doi:10.5244/C.24.87.

[PZV09] J. Prankl, M. Zillich, and M. Vincze. Motion guided learning of object models on the fly. In *5th International Cognitive Vision Workshop (ICVW), St Louis*, 2009.

[PZV11a] Ekaterina Potapova, Michael Zillich, and Markus Vincze. Learning What Matters: Combining Probabilistic Models of 2D and 3D Saliency Cues. In *Proceedings of the 8th International Conference on Computer Vision Systems: Computer Vision Systems (ICVS 2011)*, 2011.

[PZV11b] Johann Prankl, Michael Zillich, and Markus Vincze. 3D Piecewise Planar Object Model for Robotics Manipulation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference*, May 2011.

[Qui93] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[RBSF09] Alvaro Collet Romea, Dmitry Berenson, Siddhartha Srinivasa, and David Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA '09)*, May 2009.

[RBTH10] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 10/2010 2010.

[RDB06] Peter M. Roth, Michael Donoser, and Horst Bischof. On-line Learning of Unknown Hand Held Objects via Tracking. In *Proc. 2nd International Cognitive Vision Workshop (ICVW)*, Graz, Austria, 2006.

[RDB07] Hayko Riemenschneider, Michael Donoser, and Horst Bischof. Robust Online Object Learning and Recognition by MSER Tracking. In *Proc. 13th Computer Vision Winter Workshop (CVWW)*, 2007.

[RDR95] Ehud Rivlin, Sven J. Dickinson, and Azriel Rosenfeld. Recognition by functional parts. *Comput. Vis. Image Underst.*, 62:164–176, September 1995.

[Ris84] J. Rissanen. Universal coding, information, prediction, and estimation. *Information Theory, IEEE Transactions on*, 30(4):629 – 636, July 1984.

[RLSP06] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *Int. J. Comput. Vision*, 66:231–259, March 2006.

[RPD10] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010.

[RS10] Alvaro Collet Romea and Siddhartha Srinivasa. Efficient multi-view object recognition and full pose estimation. In *2010 IEEE International Conference on Robotics and Automation (ICRA 2010)*, May 2010.

[RSL10] B. Ridge, D. Skocaj, and A. Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5047 –5054, May 2010.

[SB94] Louise Stark and Kevin Bowyer. Function-based generic recognition for multiple object categories. *CVGIP: Image Underst.*, 59:1–21, January 1994.

[Sch09] M.J. Schlemmer. *Getting Past Passive Vision – On the Use of an Ontology for Situated Perception in Robots.* PhD thesis, Vienna University of Technology, 2009.

[SMB00] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *Int. J. Comput. Vision*, 37:151–172, June 2000.

[Sol08] M. Solms. What is the "Mind"? A Neuro-Psychoanalytical Approach. In D. Dietrich, G. Fodor, G. Zucker, and D. Bruckner, editors, *Simulating the Mind – A Technical Neuropsychoanalytical Approach*, pages 115–122. Springer, Vienna, Austria, 2008.

[SPV09] M.J. Schlemmer, J. Prankl, and M. Vincze. Vision for situated robot companions – fusing top-down knowledge and bottom-up data. In *AFRICON, 2009. AFRICON '09.*, pages 1–6, September 2009.

[SS07] J. Sinapov and A. Stoytchev. Learning and generalization of behavior-grounded tool affordances. In *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pages 19 –24, 2007.

[SS10] Vladimir Sukhoy and Alexander Stoytchev. Learning to detect the functional components of doorbell buttons using active exploration and multimodal correlation. In *In Proceedings of the 10th IEEE International Conference on Humanoid Robots (Humanoids)*, December 2010.

[SSS09] N. Sinha Sudipta, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *International Conference on Computer Vision (ICCV 2009)*, September 2009.

[SSW08] Konrad Schindler, David Suter, and Hanzi Wang. A model-selection framework for multibody structure-and-motion of image sequences. *Int. J. Comput. Vision*, 79(2):159–177, 2008.

[ST94] Jianbo Shi and C. Tomasi. Good features to track. pages 593 –600, jun. 1994.

[Sto05] A. Stoytchev. Behavior-grounded representation of tool affordances. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3060 – 3065, 2005.

[Sto08] Alexander Stoytchev. Learning the affordances of tools using a behavior-grounded approach. In *Proceedings of the 2006 international conference on Towards affordance-based robot control*, pages 140–158, Berlin, Heidelberg, 2008. Springer-Verlag.

[SvH01] Elizabeth S. Spelke and Claes von Hofsten. Predictive reaching for occluded objects by 6-month-old infants. In *Journal of Cognition and Development*, volume 2(3), pages 261–281. Lawrence Erlaum Associates, Inc., 2001.

[SZ09] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):591 –606, 2009.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, September 2005.

[TF08] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 537–547, Berlin, Heidelberg, 2008. Springer-Verlag.

[TK91] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[TMHF00] Bill Triggs, P. McLauchlan, Richard Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In B. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000.

[Tor98] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London A*, 356:1321–1340, 1998.

[Tor00] P. H. S. Torr. Model selection for structure and motion recovery from multiple images. In A. Bab-Hadiashar and D. Suter, editors, *Data Segmentation and Model Selection for Computer Vision*. Springer Verlag, 2000.

[TRS10] Manuel Martinez Torres, Alvaro Collet Romea, and Siddhartha Srinivasa. Moped: A scalable and low latency object recognition and pose estimation system. In *Proceedings of ICRA 2010*, May 2010.

[Ver06] David Vernon. The space of cognitive vision. In Henrik Christensen and Hans-Hellmut Nagel, editors, *Cognitive Vision Systems*, volume 3948 of *Lecture Notes in Computer Science*, pages 7–24. pringer Berlin / Heidelberg, 2006.

[VH04] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and gpca. volume 2, pages II–310 – II–316 Vol.2, jun. 2004.

[VL01] E. Vincent and R. Laganiere. Detecting planar homographies in an image pair. In *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pages 182 –187, 2001.

[VLF04] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking using Online and Offline Information. *PAMI*, 2004.

[VM04] Ren Vidal and Yi Ma. A unified algebraic approach to 2-d and 3-d motion segmentation. In Toms Pajdla and Jir Matas, editors, *Computer Vision - ECCV 2004*, volume 3021 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2004.

[WF01] Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *Information Theory, IEEE Transactions on*, 47(2):736 –744, February 2001.

[WWLG11] Thibaut Weise, Thomas Wismer, Bastian Leibe, and Luc Van Gool. Online loop closure for real-time interactive 3d scanning. *Computer Vision and Image Understanding*, 115(5):635–648, 2011. Special issue on 3D Imaging and Modelling.

[XK10] Changhai Xu and Benjamin Kuipers. Towards the object semantic hierarchy. In *International Conference on Development and Learning (ICDL-10)*, 2010.

[Zil07] Michael Zillich. Incremental Indexing for Parameter-Free Perceptual Grouping. In *31st Workshop of the Austrian Association for Pattern Recognition*, pages 25–32, 2007.

[ZPMV11] Michael Zillich, Johann Prankl, Thomas Mörwald, and Markus Vincze. Knowing Your Limits – Self-Evaluation and Prediction in Object Recognition. In *Intelligent Robots and Systems, 2011. (IROS 2011). Proceedings. 2011 IEEE/RSJ International Conference*, 2011.

[ZZV09] Kai Zhou, M. Zillich, and M. Vincze. Reconstruction of three dimensional spatial clusters using monocular camera. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pages 689 –694, march 2009.