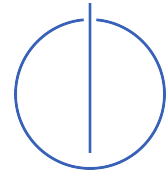




Technische Universität München

Lehrstuhl für Bildverstehen und
wissensbasierte Systeme



Everyday Perception for Mobile Manipulation in Human Environments

Ulrich Friedrich Klank

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Darius Burschka

Prüfer der Dissertation: 1. Univ.-Prof. Michael Beetz, Ph.D.
2. Ao. Prof. Dr. Markus Vincze
Technische Universität Wien / Österreich

Die Dissertation wurde am 29.06.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 20.02.2012 angenommen.

Contents

I. Preface	v
Abstract	vii
Zusammenfassung	viii
Acknowledgement	ix
II. Everyday Perception for Mobile Manipulation	1
1. Introduction	3
1.1. Manipulation in Domestic Environments	3
1.2. Perception in Domestic Environments	4
1.3. Human Capabilities as a Performance Measure	7
1.4. Motivation	7
1.4.1. Example Scenario 1: Pick and Place	8
1.4.2. Example Scenario 2: Simple Meal Preparation	9
1.5. Contributions of this Work	10
1.6. System Architecture	11
1.6.1. TUM-Rosie	11
1.6.2. TUM James, PR2	12
1.6.3. Perception System	13
1.6.4. Embedding Perception in Robotic Control	13
1.6.5. Perception Guided Manipulation	15
1.7. Outline of this Work	16
2. CoP - A Robotic Perception System	17
2.1. The Structure of CoP	17
2.2. Object Models and the Semantic Bridge	19
2.2.1. Model Annotation	19
2.2.2. Automatic Model Acquisition	20
2.2.3. Uninformed Object Analysis	21
2.3. Algorithm Selection Mechanism	21
2.3.1. Rule-based	22
2.3.2. Experience-based	22
2.4. Position-based Result Evaluation	23
2.5. Feedback Loop in a Perception System	23
2.5.1. Type of Feedback	24

Contents

2.5.2. Cascade of Feedback	25
2.6. Related Work	26
2.7. Summary	27
3. Robotic Hardware for Perception	31
3.1. Sensors for Autonomous Robots	31
3.1.1. Cameras	31
3.1.2. Stereo Setups	33
3.1.3. LASER Range Scanner	33
3.1.4. Time of Flight Camera	33
3.1.5. Structured Light Stereo	35
3.1.6. Sensor Combinations	36
3.2. Current Robots for Perception in Human Environments	36
3.2.1. Stanford - STAIR	37
3.2.2. HERB - Intel Research and Carnegie Mellon University	37
3.2.3. DLR Justin	38
3.2.4. KIT - ARMAR III	39
3.2.5. Tokyo Univ. HRP-2	39
3.2.6. IPA Care-O-Bot 3	40
3.2.7. LAAS - Jido	41
3.2.8. PR2 of Willogarage	41
3.3. Summary	41
III. Perception Tasks in Domestic Environments	43
4. Detection and Reconstruction	45
4.1. Perceiving Environment	45
4.1.1. Related Work	46
4.1.2. Scene Localization and Tracking	46
4.2. Detecting Objects	47
4.2.1. Point Cloud Clustering	48
4.2.2. Intensity Based Segmentation Methods	50
4.3. Detecting Features of Tools	51
4.3.1. Symbolic Tool Representation	52
4.3.2. Visual Tool Analysis	53
4.4. Transparent Object Detection and Reconstruction	57
4.4.1. Related Work	58
4.4.2. Absorption based Inconsistency Analysis	60
4.4.3. Segmentation	64
4.4.4. Matching	66
4.4.5. Inconsistency Check	68
4.5. Summary	71
5. Object Recognition and Categorization	73
5.1. Rigid Object localization	73
5.1.1. Related Work	73

5.1.2.	CAD matching	74
5.1.3.	Planar Shape Model Matching	79
5.1.4.	Surface Based Matching	80
5.2.	Rigid Textured Object Classification	81
5.2.1.	Related Work	82
5.2.2.	Bag-of-Visual-Words-based Classification	82
5.2.3.	Randomized-Fern-based Classification	84
5.3.	Rigid Textured Object Localization	86
5.3.1.	Related Work	87
5.3.2.	Descriptor 3D	88
5.4.	Summary	96
6.	Learning Perceptual Models in Domestic Environments	97
6.1.	Simple Model Inference	97
6.1.1.	Related Work	97
6.1.2.	Any to Color	98
6.1.3.	Any to Planar Shape Model	99
6.1.4.	Segment in 3D to Surface Model	100
6.2.	Acquisition of CAD models	101
6.2.1.	Semantic-aware Selection of CAD Models	102
6.2.2.	Model Selection with a Shape Distribution Function	109
6.2.3.	Morphing between CAD Models	111
6.2.4.	Selection by Best Match	114
6.3.	Texturing of Object Models	114
6.3.1.	Related Work	114
6.3.2.	Acquisition of Training Data	115
6.3.3.	Conflict Resolution via Bundle Adjustment	116
6.3.4.	Reconstructing the Texture	118
6.4.	Summary	119
IV.	Applications of CoP	121
7.	Perception Guided Robotic Manipulation	123
7.1.	Accuracy and Robotic Manipulation	123
7.1.1.	Camera Calibration	124
7.1.2.	Hand Eye Calibration	124
7.2.	Dealing with Uncertainty in a Robot System	127
7.2.1.	Covariance Propagation	128
7.2.2.	Located Object Tree	129
7.3.	Grasp Planning	132
7.3.1.	Related Work	133
7.3.2.	Grasp Pose Optimization on a Gaussian Point Distribution	134
7.3.3.	Detection of Collisions in the Fingers	138
7.3.4.	Motion Control	141
8.	Results	143

Contents

8.1. Object Localization	143
8.1.1. Planar Matching	143
8.1.2. CAD Based Matching	147
8.1.3. Descriptor Based Matching	151
8.1.4. Transparent Objects	154
8.2. Robotic Manipulation	157
8.2.1. Unmodeled object Grasping	157
8.2.2. Online Tool Calibration for Complex Tasks	159
8.3. System Evaluation	161
8.3.1. Offline Evaluation	162
8.3.2. Online Evaluation	164
9. Conclusion	169
9.1. Summary of Results	169
9.2. Hardware Challenges	170
9.3. Future Work	170
V. Appendix	173
A. Table of Sensor Properties	175
B. List of Implemented ROS Packages and their References in this Work	179
B.1. cognitive_perception	179
B.2. cop_halcon_plugins	179
B.3. cop_cad_plugins	180
B.4. cop_ros_plugins	180
B.5. cop_sr4_plugins	180
B.6. cop_transparent_objects_plugins	180
B.7. cop_barcode_plugin	181
B.8. cop_odu_plugin	181
B.9. cop_tool_plugin	181
Bibliography	183
Index	196

Part I.

Preface

Abstract

This work describes a modular perception system for mobile robotic manipulation which improves over time. The two key features of this system are the automatic adaption to its environment and the large set of available perception methods. In order to adapt to a changing environment, the system learns success statistics over executed perception tasks and acquires new perceptual models. The perception system can handle different perceptual tasks which are processed by an automatically selected triple of sensor, model and method. This triple is selected based on logical restrictions and the success statistics.

For the challenging scenario of a robotic platform in a domestic environment, the system is equipped with a large set of methods, which enable the perception system to process all relevant tasks. Examples for such tasks are the transformation of semantic descriptions into perceptual models or the detection of objects that are transparent to the sensors or to localize objects that were not seen before. Several methods to address such and similar problems were newly developed and are presented and evaluated in this work.

The system can autonomously infer new models by connecting known models with new sensor data, which is enabled by a set of state of the art methods that were embedded into the perception system. The system has a high degree of abstraction to be easily accessible by a highlevel system, while allowing dynamic reconfigurations of methods and models and providing full insight into the applied methods and the resulting data. The system supports spatial reasoning by a high-level system by allowing the setting of 6D search spaces. Alternatively, those search spaces can be generated automatically and task dependent.

The performance of the system will be evaluated in the context of general object localization, pick and place of unmodeled objects and in the context of a food preparation scenario.

Zusammenfassung

Diese Arbeit beschreibt ein Perzeptionssystem für mobile Roboter, mit der Fähigkeit sich an einen Einsatzort anzupassen. Die entscheidende Neuheit des präsentierten Systems ist neben der Fähigkeit sich automatisch an eine Umgebung anzupassen die große Anzahl an vorhandenen Methoden die spezielle Perzeptionsaufgaben lösen können.

Die Anpassung an eine Umgebung erfolgt mittels zwei Mechanismen: Erstens werden alle Ergebnisse bisher erledigter Aufgaben bewertet, um bei zukünftigen Aufgaben eine bessere Wahl der Methode treffen zu können. Zweitens können neue Objektbeschreibungen automatisch gelernt oder von außen hinzugefügt werden.

Die Bewertung von erledigten Aufgaben erfolgt basierend auf Tripeln bestehend aus der Methode und auch dem verwendeten Sensor und dem zugrunde liegenden Objektmodell. Für eine neu erteilte Aufgabe, wählt das Perzeptionssystem ein solches Tripel basierend auf den bisherigen Erfolg dieses Tripels aus, gibt es kein bisher erfolgreiches Tripel werden neue Tripel gebildet.

Das Perzeptionssystem ist vor allem für einen Einsatz in einem Haushalt mit vielen Methoden ausgestattet, die dort alltägliche Aufgaben der Wahrnehmung erledigen können. Beispiele für solche Aufgaben kann z.B. die Detektion von transparenten Objekten sein oder die Handhabung von bisher unbekanntem Objekten. Solche und ähnliche Aufgaben werden von den Methoden angegangen, die in dieser Arbeit beschrieben sind. Außerdem kann das System aus Sensordaten bestehende Modelle erweitern oder neue Modelle erzeugen.

Das vorgestellte System ist in eine Softwarearchitektur mit einem sogenannten Highlevel System eingebunden, für die ein abstraktes Interface entwickelt wurde, das sowohl semantische Annotationen als auch räumliches Schlussfolgern zulässt. Zusätzlich erlaubt es dem kontrollierenden Highlevel System auch vollen Einblick in Daten und Ausführung.

Das System wird anhand von Anwendungen im Bereich Objekterkennung, Pick-and-Place von unbekanntem Objekten und in einem Küchenszenario mit Essenszubereitung evaluiert.

Acknowledgment

I want to thank Prof. Michael Beetz and Prof. Bernd Radig for providing the great hardware setup that enabled this research, as well as for the scientific input, the literature hints and all the support I profited so much from. I also want to thank my colleagues for helping me in various ways to collect the material for this thesis: Special thanks to Markus Ulrich and Christian Wiedemann for teaching me so much about CAD matching and handling. After these lessons they always were available for questions. I want to thank Andreas Hofhauser and Bertram Drost for their cool methods to localize objects in HALCON that allowed me to solve many of the problems I encountered so fast during my research. To Lorenz Möslechner and Moritz Tenorth special thanks for always being interested in the results I could provide and for even accepting my interfaces. Without Alexis Maldonado, Federico Ruiz and Ingo Kresse the robots would have never worked how I needed them for this work, thanks guys and sorry I came so often to ask for more. Radu Rusu, Nico Blodow, Dejan Pangercic, Dominik Jain und Zoltan Marton I want to thank for their cooperation in the perception research and for the joint publications. I am thankful that my students Zeeshan Zia, Sun Li and Daniel Carton worked so hard with me for our publications. Additionally they allowed me to use parts of their thesis materials to better visualize interesting facts in this thesis, as did many of my other students: Pavel Mihailov, Klaus Gleissenthal, Janosch Peters, Florian Ferstl and Rok Tavcar.

This work is supported by MVTec GmbH, München and the CoTeSys (Cognition for Technical Systems) cluster of excellence. This work profited from HALCON and ROS [102], especially from rviz.

Part II.

**Everyday Perception for Mobile
Manipulation**

1. Introduction

The robotic butler that helps in our households not only is a dream of the authors of Hollywood movies, but it also became recently a realistic research target. Due to recent developments and with a human tele-operating, a state of the art robot may perform nearly any assignment in a domestic environment. Still, there are only few applications which can be performed nowadays by autonomous machines, even if the task itself can be executed using only the available hardware for manipulation. This raises the question why capable machines like state of the art robots are not able to help in any complex manipulation task in a kitchen or a living room, while already working completely autonomous 24/7 in factories around the world. The tasks are on the one hand solvable, but on the other hand they require very exact control as well as the complete understanding of the observed environment and an understanding of the limitations of the robotic hardware.

The understanding of the observed environment is one of the crucial key competences that autonomous systems still lack nowadays. This problem was not yet solved even with the state of the art in computer vision and 3D perception that provide methods to detect most of the relevant events and objects in a controlled environment. Obviously, the domestic environment cannot be seen as a controlled environment. This raises challenges like the presence of sensory noise or cluttered scenes. Driven by the requirements that a realistic domestic environment imposes, this work presents a perception system that allows autonomous robots to perceive objects and events in a household. The system allows recognizing and localizing all objects which are relevant for tasks like by tidying up and even preparing meals. It has mechanisms to improve its performance over time by automatic learning of context and new objects. All those mechanisms are targeted on the manipulation of the environment.

1.1. Manipulation in Domestic Environments

This work focuses on the perception for manipulation which will be discussed starting with the challenges for manipulation in such a scenario.

To enable robots to interact with an unstructured environment inhabited and influenced by humans, they require the capability to manipulate objects safely. The range of possible tasks

1. Introduction

may include the pick-up of objects as well as the usage of objects as tools or more general, in may include an application of force to the object.

How complex this task is, may be seen by looking at the requirement "safely" mentioned before. This requirement can be split up into the requirements of not harming any human in the process of manipulating the object and not destroying the robotic hardware itself as well as not producing any unwanted effect to the environment which is irreversible for the robot. By approximating the world to be momentarily stable and the hardware of the robot by being indestructible by the robot itself, the manipulation task still has to be performed under the constraint of not producing any unwanted effect is still not solved yet for unstructured environments. Even if there are methods for collision free motion planning (e.g. see [53]), the limitations of those methods are insufficiencies of the sensors reacting to objects made of metal or glass as well as the necessary calculation time and missing guaranteed convergence of the methods.

What will be discussed in this work are the detection of difficult obstacles and the fast modeling of obstacles for simple object manipulations. Additionally, in order to manipulate the right object, the identity of objects has to be observable.

Any motion of a robot requires not only an accurate modeling of the endangered environment but as well of the robots position and its target object. The target has to be known in as high precision as possible to allow complex interaction. Constraints on motions must be extractable from the task and the object to handle. Such constraints may be e.g. the requirement to uprightly carry a filled glass or transport any food on a plate. Such manipulation with this kind of constraints and targets depends partially on capabilities to perceive and observe the environment in different ways. The kind of perception needed will be analyzed more precisely in the following section.

1.2. Perception in Domestic Environments

Coming from the view of just the discussed requirements on mobile manipulation, perception tasks can be classified using the scheme also depicted in Figure 1.1: First, a detection of present objects in order to separately handle them, second, a reconstruction of objects in order to avoid or move them, third, the recognition and localization of objects that are already known to the system, fourth, a categorization of objects to create a semantic context and last, the autonomous extraction of recognizable properties in order to be able to identify those objects at a later point in time.

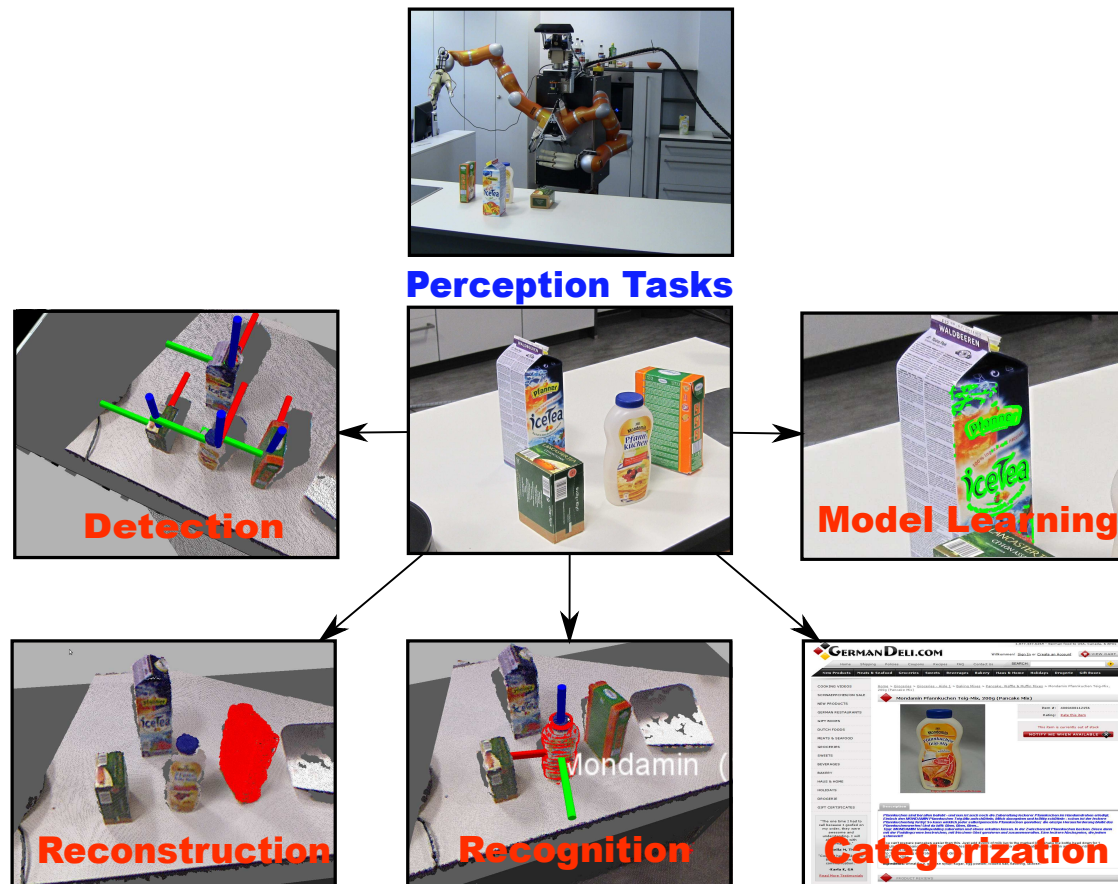


Figure 1.1.: Five classes of perception tasks, visualized by the results on a scene on a kitchen table.

- Detection of objects and obstacles and with it the generation of of the best object hypothesis is directly derived from the requirement to avoid any obstacle in manipulation. But it is required as well to reduce computation time in other following perception tasks by being able to reduce certain calculation to special classes of detections.
- Reconstruction of objects should complete missing sensor information. This would then allow simpler manipulation as well as more accurate handling of those objects. The reconstruction of missing sensors data can also lead to better models. In this context it is necessary to have knowledge about problems in sensors and be able to profit from any structuring of the current environment.
- Recognition of objects connects previous available data with a current instance of a specific object, which allows the accurate localization in space and time of a certain object.
- Categorization is a more general way to connect an object hypothesis with a certain

1. Introduction

kind of category or class of objects. This allows any knowledge driven system further inferences over the scene and the current world state.

- Model learning as last step prepares the recognition of future occurrences of objects and allows the robot to detect changes in the environment.

All those tasks have to be solved under the constraints that are imposed by being executed on a mobile platform in a domestic environment [52]. The solutions for all or at least for some of them demand of the tasks to:

- be robust against sensor noise,
- perform under time constraints to be able to interact with a dynamic world,
- generalize over as many certain parameters as possible to reduce the number of required specialized solutions,
- produce interim results that are helpful for other systems on the robot.

Additionally, there are many situations in a domestic environment, which have to fulfill still more requirements.

The detection of objects and obstacles is the basis for any safe behavior of a robot. If the robot cannot perceive the existence of its surrounding, it cannot act safely. Additionally, by identifying connections and separations in its environment, it simplifies the remaining perception problem, by introducing a degree of dependence or independence to the identified object hypotheses. As soon as it comes to the interaction with an object, the knowledge about the presence might not be enough, so a reconstruction of occluded or invisible parts will be necessary for a direct interaction. Without having a good approximation of the outer shape of an object, any grasp or the usage of an object will be difficult to impossible. If the robot is confronted with difficulties or good experiences in such manipulation processes, the necessity appears that the robot should be capable of recognizing similar situations which the robot interacted before. For any object that was manipulated or seen before it will be important to be able to identify it later again. This process of recognition combined with model learning should run completely autonomous. If this model learning is performed supervised or it is executed from an annotated database, the recognition can be paired with a categorization of the seen object. This allows the establishment of bridges to a semantic level, which we see as an important step to allow highlevel control of robots. Highlevel control can then allow the robot to act such that a human can understand its behavior and also can judge the quality of the robots performance by comparing it with other human's performance.

1.3. Human Capabilities as a Performance Measure

If we look at pick-and-place tasks, a human can still outperform most robots, especially in scenarios with dynamic obstacles and new objects. All tasks in a household are usually performed by humans and most humans are able to easily realize them, believing the tasks they are performing are easy to learn and understand. All those tasks are designed for humans and can be robustly solved by a human even in another home, given the necessary information where the eventually required tools are and what the target is.

Especially all perceptions tasks in domestic environment, like the detection of transparent objects, seem trivial to most humans, while many of them can be considered as real challenges for robotic perception systems. In order to reach this performance on a robot in a varying environment, a robotic system needs to gain experiences and collect knowledge in order to perform similarly well than a human.

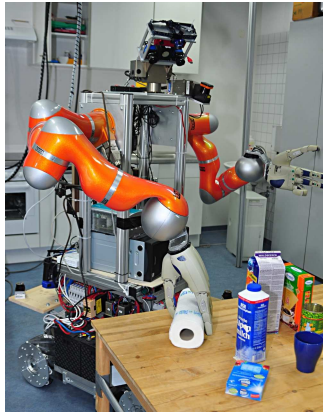
The list of perception tasks, in that humans outperform any robot so far, is long, and this list contains tasks like scene segmentation, categorization of previously unknown objects. The basis for this performance is the experience of decades of living in such domestic environments combined with the strong knowledge exchange between humans. While this kind of experience is not completely applicable to the standard idea of delivering a ready-to-work household robot, the possibility must be given to teach the robot before its delivery as much information as possible. A new help in a household can make minor mistakes on the first day in a new home, but the requirements are not low.

This comparison of the capabilities of a human with a potential robot shows the necessity of pushing the state of the art on the technical level as well as on the semantically-driven learning level. And this is what this work is all about.

1.4. Motivation

This work presents and discusses a perception system that it is capable to work under the complications introduced by running on a mobile platform in a domestic environment. It provides a large set of methods capable of performing all five tasks mentioned before. In order to explain the challenges and the necessary tasks to solve, two example scenarios are introduced here: First, tidying up with a robotic platform and second, the preparation of a pancake.

1. Introduction



(a) TUM-Rosie manipulates an object.



(b) TUM-James manipulates an object.

Figure 1.2.: Tidy-up scenarios.

1.4.1. Example Scenario 1: Pick and Place

Let us discuss the situation depicted in Figure 1.2(a): A mobile robot with a set of sensors mounted on its head picks up an object that is lying on a table among other objects. The presented perception system provides solutions for the following tasks required to clean up in this scenario:

The objects on the table have to be recognized and segmented, as well as reorganized by pick-and-place actions. It is also required to distinguish objects that have to be moved away or that should stay in place. While the recognition of the objects basically can be solved by segmentation and a successful classification of an object, any manipulation requires additionally a reference to 3D space. This can be achieved by measuring 3D coordinates of the object's center or by reconstruction of its shape. With a measured position or a direction in space, the robot can be ordered to move towards the object and try to grasp it. This movement must avoid all obstacles nearby and should avoid any collision with the object itself before finally grasping. Even with the best planning of this action, this attempt may fail and the attempt better should be supervised by the robot. This supervision might be difficult with a head mounted sensor that suffers from occlusions of the object by the robot. Additionally, the avoidance of the obstacles or the planning of the contact with the object depends heavily on knowing the geometry of the objects in the scene. However, in a domestic environment a robot will be confronted with objects that were never seen before. Nevertheless, the capability to manipulate so-called unmodeled objects is crucial for a task like tidying up. The distinction of objects can be achieved by the implemented capabilities to learn models to recognize objects that were once shown to the robot, as well as by the capabilities to use external sources for new models.

Another crucial capability of the presented system is the recognition of transparent objects. Those objects have properties that result in ambiguous readings in all light and laser based sensors. Looking at the data which is used for geometric segmentation in Figure 1.3 and comparing the reading from the opaque objects and the answer of the semitransparent bottle, it can be easily seen that a completely different method will be required to reconstruct the real 3D structure of the transparent object.



(a) A view of a table with two opaque objects on the sides and a semi transparent object in the middle.

(b) The frontal structures of the two opaque objects are visible in the 3D data, while the transparent parts of the centered object do not appear.

Figure 1.3.: The differences of opaque and transparent objects in depth measurements.

1.4.2. Example Scenario 2: Simple Meal Preparation

If the task is extended to something more complex, like the preparation of a pancake, it is necessary that the perception system can detect state changes in the environment and is able to handle tools. Descriptions for such kind of tasks are so complex that they might include name of objects which are unknown to the system. In the pancake scenario, the object of interest is a Pancake-Mix which is considered as an unknown object. Figure 1.4 shows the robots handling a spatula which is used to flip a pancake. This spatula is grasped and calibrated in the pancake preparation scenario, based on mechanisms of the presented perception system.

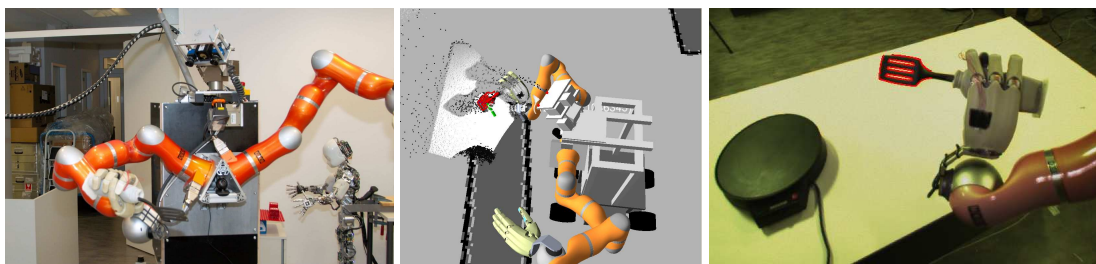


Figure 1.4.: Calibration of the spatula for flipping pancakes.

1. Introduction

For the usage of an object as a tool, the requirements regarding accuracy increase. In our system this is implemented by a post-grasp evaluation of the visible parts of an object in order to find out the position in relation to the prior end effector of the robot. In order to evaluate state changes, the different methods are required than used for object localization: To detect if a pancake is ready, the necessary specialization of the model is higher than for rigid objects. The pancake is localized in this scenario by using a model of the underlying pan and segmenting parts that are inside the pan based on color cues distinct from the pan itself. The case of the pancake shows that it is of interest to model not only the object but its effects on the environment which implies different methods of perception. Those methods must be able to compare a model with the current world state in a robust manner.

1.5. Contributions of this Work

The two exemplary scenarios include tasks which are addressed by the presented robotic perception system. As a system running on our hardware platforms, this work presents here **Cognitive Perception (COP)**: A robotic perception system which is capable to apply various perception methods to different perceptual tasks. This functionality is provided over an abstract interface that integrates well into highlevel control mechanisms. Most of the necessary model-knowledge can be acquired automatically by COP.

The main novelties which are implemented in the system itself are:

- Perception results from various methods are provided to highlevel systems.
- Automatic selection of the method improves the performance for previously performed task.

For selecting different methods and solving different tasks over a simple interface, a lot of automatic calculations and internal knowledge are necessary. Internal robot properties and also world knowledge are required to decide what is meant and what are the best method to solve the perception task and the correct perception result. With the capability to learn over time to use the implicit structure in its environment and allowing all parts of the system to be replaced or extended, COP is suited for the challenge to work in a dynamically changing environment. The autonomous learning capabilities based on the World Wide Web and system intrinsic feedback provide the possibility to work with new objects.

Additionally, the system contains contributions on the algorithmic level which are novelties regarding the following key points:

- Access to data sources in the World Wide Web is provided to acquire new models

- Methods to detect most kind of objects, including transparent objects, are presented
- Algorithm improvements are shown for several state of the art methods for object recognition

While on the algorithmic level a lot of research is done and solutions for many special problems are provided, the combination of those methods in order to provide a computational model for generic perceptual tasks is shown the first time in COP in a scale tackling a domestic setup.

How COP works inside a robotic system will be described in the following sections.

1.6. System Architecture

Ian Horswill [48] has proposed the specialization of visual routines to tasks and environments as a promising method to transform perception methods to environment- and task-specific routines. After such a transformation, the methods can outperform the general routines in terms of reliability as well as efficiency as long as the modeled properties of the respective tasks and environments are satisfied.

In addition, COP improves itself over time, both in terms of reliability and efficiency, based on experiences in performing the visual tasks of the respective robotic application and thereby specializes itself to the task, robot, and environment at hand. The robot system has a redundant set of perception methods, multiple sensors with similar as well as different capabilities, and the system is able to form different task-specific models of detected objects in the environment. With this design and various inspection interfaces, COP was integrated into a robotic system. The internal representations and the integration into the highlevel and action system will be described after the description of the two hardware platforms.

1.6.1. TUM-Rosie

The main platform used for testing is TUM-Rosie. This robot hardware is depicted in Figure 1.5(a). The robot drives on a mobile platform with omni-directional wheels that enable the robot to drive and turn freely in all directions. For manipulation it has two DLR/HIT hands each mounted on a KUKA LWR III. The hands have two torque sensors in each finger and are controlled using impedance control. The arms are compliantly controlled using force measurements in all joints as feedback. This allows safe working within the range of the robot. The sensor setup used for perception consists of two RGB cameras eco274 of SVS Vistek, a Swissranger 4k and a tilting Hokuyo UTM-30LX Laser. Most sensors are mounted on

1. Introduction

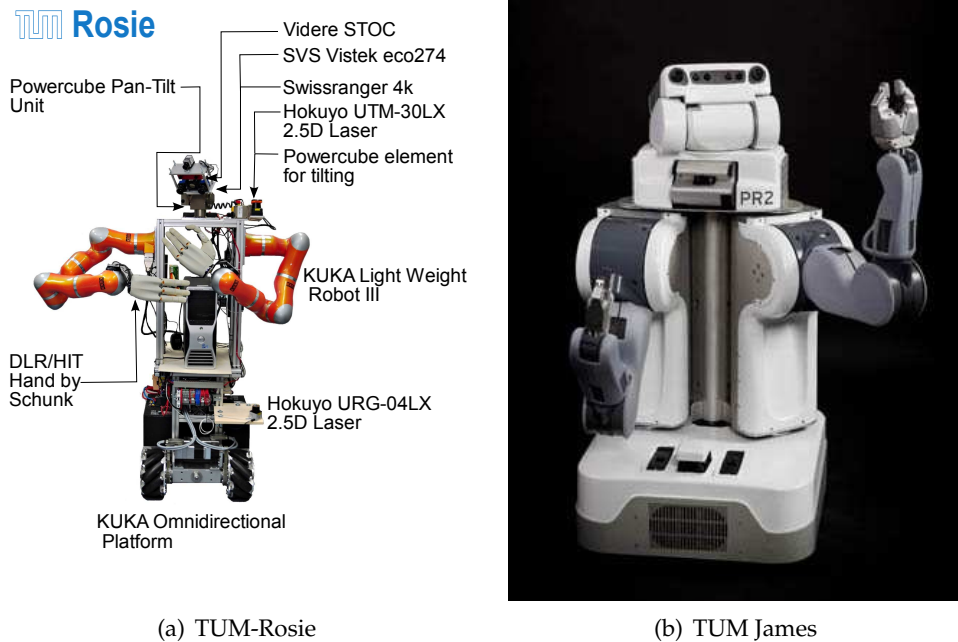


Figure 1.5.: Robotic platforms of the Intelligent Autonomous Systems (IAS) group at Technische Universität München (TUM).

the head assuming that this position allows observing the robot's actions while interacting in scenarios with elevated supporting planes like tables. For collision avoidance the platform has two Hokuyo URG-04LX Laser at approximately knee height. Those lasers are the major source for localization in known environments.

Thanks to the KUKA LWR arms and the sensor equipment, this robot is considered as a leading edge platform in 2009, while it lacks a bit the reliability provided by standard platforms like HRP2 or PR2. The over sized design causes some limitation for small human-scale objects like cutlery. Those objects are just too small to fit into the large hands. Additionally this platform cannot manipulate directly on the floor, since it lacks a mobile torso.

1.6.2. TUM James, PR2

The PR2 has less problems with human-scale objects: TUM James was granted by Willowgarage to the Intelligent Autonomous Systems group in order to implement CRAM, a proposal about a Cognitive Robot Abstract Machine. It is a Personal Robot, version 2, which has an extendable spine and simple grippers, to name the most significant hardware differences to TUM Rosie. Also the arms are gravity compensated by a spring system instead of the software approach provided by KUKA for the LWR arms.

The sensor setup provides additionally a structured light stereo system, while it lacks a ToF

Camera (see 3.1.6). The gripper is equipped with a sensitive tactile sensor and a reactive planner using those. The robot has an integrated trigger system to use all cameras, with and without its projected structured light synchronously. It is shipped with a complete modeling and an automated calibration procedure.

1.6.3. Perception System

The basic data structure of COP is the mapping of visual tasks onto a success statistics, wherein the visual task is represented as a combination of a method, a model, and a sensor configuration in space:

$$vis-task \times \langle method, model, sensor \rangle \longrightarrow success\ statistics$$

where each method is applicable only to a subset of model types and specific sensor configurations. The statistics for these method, model, sensor combinations form the basis for the selection of a task specific method, model, sensor combination. The acquisition of the success statistics is supervised by the highlevel control system of the robot that provides feedback to COP. This feedback contains information about whether or not a perceptual task has been solved successfully. For example, the highlevel controller might specify in the context of a pick-up task that the localization of an object was successful if the robot did successfully grasp the object; otherwise the success of the visual task is unknown, since a grasp failure could have been caused by the grasp execution as well as by the perception routine.

The presented COP system essentially learns in two ways. First, it infers from detected object candidates new models for objects and thus increases the set of applicable $\langle method, model, sensor \rangle$ triples. Second, it learns the decision rules for selecting the appropriate method/-model/sensor combination by continually collecting the results of their application whenever they have been applied in the context of problem-solving.

The results will show how using this autonomous specialization through this learning process, can substantially improve the performance of the robot perception system by specializing to the tasks and the environment.

1.6.4. Embedding Perception in Robotic Control

Fig. 1.6 shows the embedding of COP into the robot control system, which is described more detailed in [9]. In a nutshell, the control system consists of the perception component COP, the action module that translates action specifications into control signals, monitors

1. Introduction

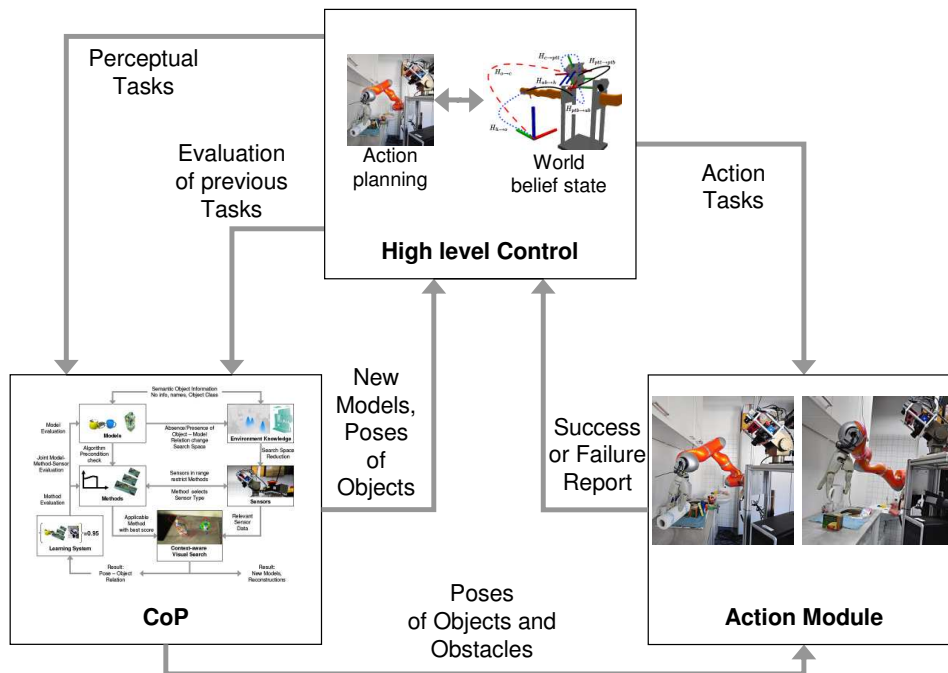


Figure 1.6.: The integration of the executive with the perceptual system and the actuation system.

the action execution and provides feedback about the execution of an action to the highlevel control system.

The highlevel control system specifies how the robot has to respond to perceptual input, to failures and to known context of the current situations in terms of parameterizing the task. The highlevel system has a set of plans that describe necessary action to achieve certain goals. If a new goal for the robot appears, by command or reactively as a subgoal of a larger goal, the respective plan is executed.

For the example of the preparation of a pancake, the information flow is sketched in 1.7. The system has to react to new tasks given from the outside, as well as information that can be acquired from non-static resources like the World Wide Web, it has to process sensor data and should observe the reaction of the environment to actions of the robot.

In order to provide the information needed for execution of an action, the highlevel system queries COP which will answer these visual tasks. The current capabilities of COP can be queried and adapted online in automatic or supervised ways. The highlevel control has to be aware of three main capabilities: scene segmentation, object localization and model refinement. Those three actions are triggered accordingly to the current goals of the highlevel system. The required parametrization of those actions can contain as much relevant infor-

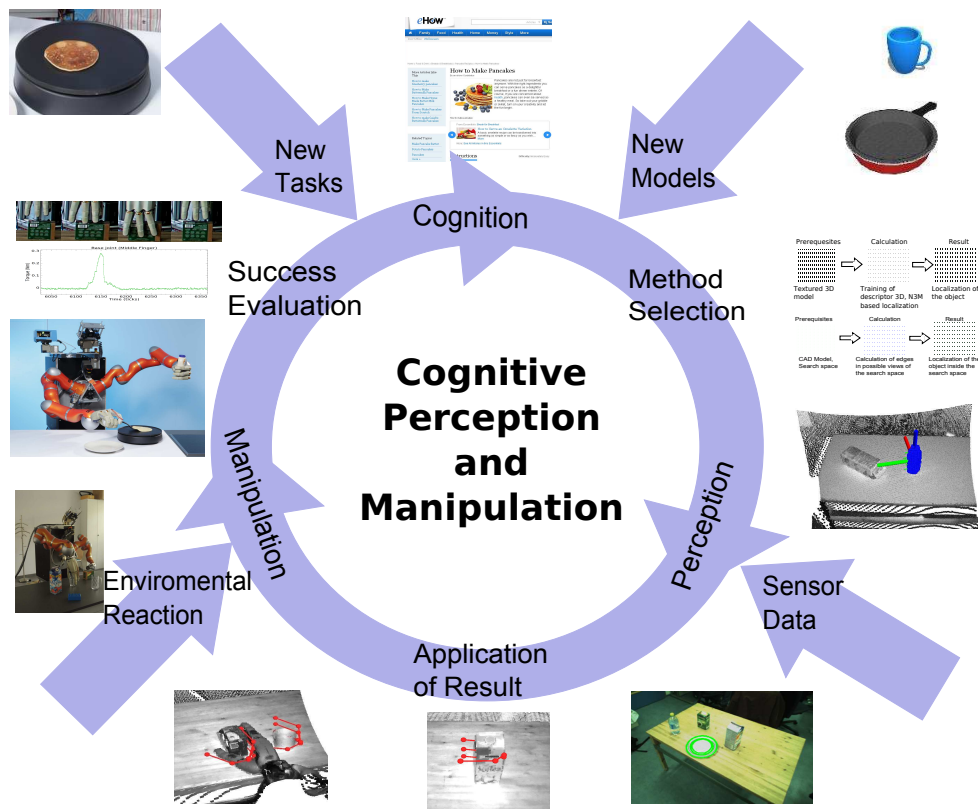


Figure 1.7.: A Cognition-Perception-Action loop for the example of pancake preparation.

mation as available regarding locations and models applicable to this task. The refinement of models contains all actions which learn new models or categorize object candidates and also all non-trivial reconstruction methods.

1.6.5. Perception Guided Manipulation

In robotics the most important application for object perception is manipulation of those objects. In order to enable manipulation given a certain perceptual result, the accuracy of the result must be known. How much does the result say about the shape and how accurate the position is estimated are crucial questions for manipulation. This is influenced by the choice of perception method and the entire accuracy of the actuation system. The accuracy of the system can be increased by a careful system calibration. Being aware of the actual and potential errors and uncertainties allows a manipulation system to act more adequately to a situation.

1. Introduction

1.7. Outline of this Work

The core system CoP will be presented in Chapter 2, which focuses first on details in the internal representation of models and the semantics in Section 2.2. This is followed by the selection mechanisms for methods in Section 2.3 and the position aware result evaluation in Section 2.4. The feedback system required for the Cognitive Loop is described in Section 2.5. The related work also contains systems with similar intents to enable robotic vision and compares them with the CoP in Section 2.6.

In Chapter 3, an overview over the field of service robotics is given. This starts with an overview over current sensor techniques in Section 3.1 and other robotic platforms with their capabilities and problems described in Section 3.2.

Leaving the general system architecture, the Chapters 4, 5 and 6 explain the tasks and the proposed solutions for perception problems. The topics are ordered from first detection mechanisms, over localization methods for different object properties to automatic modeling and model acquisition. Chapters 4 goes over detection mechanism for environmental properties in general and more specialized for opaque and transparent objects. Chapters 5 describes the model based localizations and recognition methods based on e.g. CAD models or image templates. The Chapter 6 discusses solutions to several interesting problems for the perception tasks detection and reconstruction, categorization including CAD model inference and transparent object detection, which are part of the contributions of this work.

Continuing the data flow from the perceptual result to the execution of an action, Chapter 7 discusses the special challenges introduced by manipulation. Providing the presented perception mechanisms and proposing solution for robotic setup and calibration, the internal representation of results in 3D are discussed under consideration of uncertainty and a simple mechanism for grasp planning is described using this uncertainty representation.

Chapter 8 summarizes all results for the newly introduced techniques focusing on single perception tasks and the manipulation system. Additionally, interesting applications of the system will be shown in the context of simple meal preparation.

A conclusion and an outlook on possible future developments will be given in Chapter 9. There, the solutions and the open research questions regarding the presented perceptual system will be summarized.

2. CoP - A Robotic Perception System

In this chapter the core of COP is presented and the most essential components of the system are explained. We will explain in detail the mechanisms for including semantics, the learning system and the feedback mechanism. After a look at all important features, other perception frameworks and systems will be introduced in order to work out the novelties of COP.

2.1. The Structure of CoP

The structure of the COP can be visualized using three components containing different levels of data which are connected online to maximize the chance of getting a valid percept as result. The data contained in the system is on the one hand a model that describes a certain object and semantic object types and on the other hand data describing the robot's relevant hardware setup and the methods available to the system.

The models are representing the bridge to the highlevel, they can be queried and added or annotated over services the system provides. All models have a certain type, e.g. CAD models or color distributions, and they are annotated with a name representing its class, which is connected with a semantic concept. This method will be explained more detailed in the following section 2.2.

The system requires the information of the relative location of sensors and their field of view and a list of methods that it can be applied to those sensor data. Additionally, the basic requirements and conditions of those methods have to be modeled. Positional constraints will be formalized in section 2.4.

The most important step inside of COP is to establish the connection between models, methods and their results. This step will be analyzed in section 2.3.

COP can be seen as a functional component that takes semantic object descriptions and returns a probable position with respect to a model that was connected with the given description. Fig. 2.1 shows the data flow between the components of the system. The first step is to look up all semantic description in the model database (on the left side of the figure), which checks for already assigned models. The next step groups all found models to a collection

2. COP - A Robotic Perception System

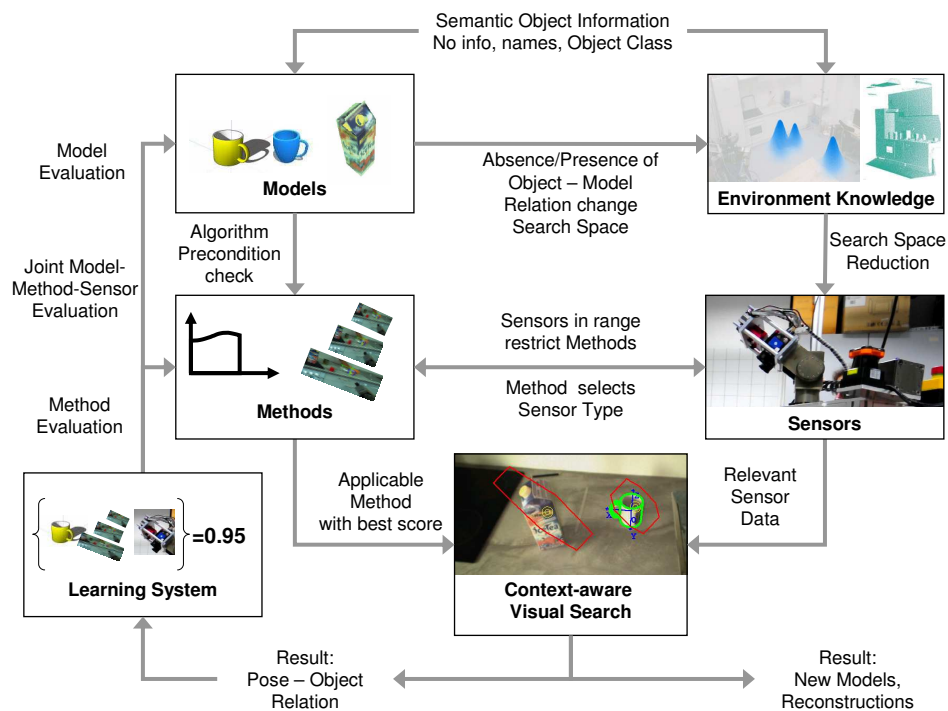


Figure 2.1.: The task to locate any object as precise as possible is formulated by giving the system object related information. This information may be already connected with model information (e.g. learned features). The available model information leads to the selection of a method and a sensor.

which is passed to the selection process for methods. Any previously unknown concept will be stored for model inferences, see section 6.1.

This collection can influence the search space which can be created by mechanisms of COP(see section 4.2.1) or is inferred from environmental knowledge and the current world believe state, which will not be discussed in this work.

The search space might be seen only from some of the sensors, which restricts the set of usable sensors. The model collection and the sensor collection make it possible to predict the probability of different methods that might work out. Many vision algorithms have models as preconditions and only work on a certain image type, those constraints are hard-coded in the respective method and are checked in this phase. Additionally, the collected experience is considered here, which consists of previous successes or failures of certain methods and certain model-method combinations. This final selection of model, method and sensor is a perception primitive. All model-method combinations selected here will be evaluated, if

there is a feedback provided in some manner. An unexpected failure of the search would be an immediate feedback, any result must be judged by an additional component, like the success of a grasping action or a contradicting physical simulation (to detect false-positive object detections).

After the system will be described in detail, an overview over the current literature will be provided. We will discuss robotic vision systems at other leading research institutes.

2.2. Object Models and the Semantic Bridge

The most important task for COP is to create a belief state over the environment. Therefore it provides the possibility to query the positions of objects. Objects are represented in the highlevel system as a semantic concept that usually has a name or identifier. Semantic concepts will appear in the interaction with a human or by interpreting task descriptions which are generated by a human.

This term will be passed to COP to create a mapping between term and perceptual model. This mapping can be either provided manually for all models, or it can be generated automatically. As a format for exchange of those semantic concepts, WordNet is used to represent the name of the concept and to pass it between different system components.

2.2.1. Model Annotation

To give objects names is a human habit, even if those names are not unique or task relevant. Given such a name and a known context and with common experiences, a name can be interpreted correctly by another human. Transferring this fact to a system that runs on a service robot, three capabilities are required: First, it has to be able to couple objects with their names. Second, those names have to be context sensitive and third they may alter given a new source of interaction with a new counterpart.

In the implementation covered by this work, only the first two capabilities are addressed. WordNet is implementing those capabilities, so it was a intuitive choice to build up on the WordNet taxonomy, which is already large and contains nearly all English words. It contains many words which are used in common naming of objects that also covers context in certain meanings.

2. COP - A Robotic Perception System

WordNet

WordNet was initiated by Miller et al. ([72]) in order to provide an online lexical database. It orders English vocabulary into groups of synonyms, so called synsets, and a hierarchical structure of those synsets. This hierarchy can be used to distinguish between different meaning of one word and similar meanings of several words. This is important in an automatic annotation process, which is based on communication with humans or human written information sources (like most parts of the Internet).

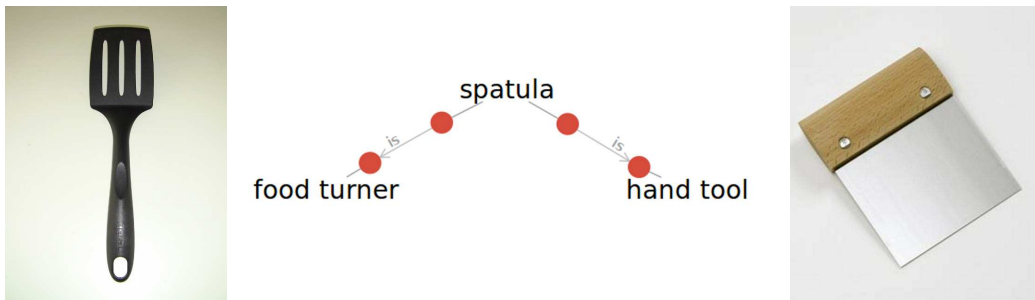


Figure 2.2.: The word spatula and two different meanings visualized on the left and the right, and the WordNet graph connecting the two meaning in the middle.

To compare words and annotations in Internet-based databases via distance measures, WordNet provides a set of methods. It provides several similarity and relatedness measures between synsets, which will be discussed later in the context of annotations of CAD models in Section 6.2. Thus, the concept of a synset can be used to identify objects by relating them via such a distance or relatedness measure with a concept in a database. For example, such a similarity can help to resolve ambiguity visualized in Figure 2.2, representing two possible results of the image search of Google for *spatula*. On the right ¹ there is a tool that is used for crafting and on the left there is the tool that is used in a kitchen which might be the one we were looking for.

2.2.2. Automatic Model Acquisition

This synset allows improved communication with Internet databases that require no further interfaces for the usage by a robot. Most common Internet databases that contain useful information can be explored via natural language search-terms with certain techniques specialized to different databases and data types (see e.g. Sections 6.2 or 5.2.2). We can acquire new models from these databases, even under the polysemy of a certain word.

¹Courtesy to <http://www.stuccoitaliano.it/images/stucco-tools/spatula-woodhandle-big.jpg>, 2011

2.2.3. Uninformed Object Analysis

With a distinct approach, coming from the percept, we want to gather as much knowledge as we can in order to understand previously unseen objects. While a geometric and visual description is enough for internal recognition, the communication with a human cannot easily be solved. However, for internal tasks like obstacle avoidance or moving a group of objects, a semantic analysis is not necessary.

Internal Uninformed Object Representations

Internal names like “Transparent Object” or “Cluster” describe such instances of unidentified objects with certain properties. Also descriptions which only help for debugging purposes like “Texture123” or “ColorModel321” can be interpreted by a human only partially, but still can be used analogous like the terms with a WordNet specification.

Elevation to Known Concepts

The same interface can be used to expand the internal representation by connecting a detect object candidate with a semantic concept. This could be done in an interactive way by the user or over a reasoning process that derives the identity or usage of an object. This information can be simply passed to COP.

2.3. Algorithm Selection Mechanism

In order to select the right perception primitive COP combines static rules with experience. The system holds a list of available models per semantic concept M and a list of methods A (=Algorithms). Both can change online and are initially evaluated with a certain quality. It also holds the list of currently available sensors S . For any method, there exist static rules regarding possible combinations with sensors and models. All methods, models, and model-methods combinations are evaluated in case they are executed.

We build a simple ontology for all methods, their prerequisites and results. The ontology reflects also relations between values in the results, regarding the meaning of equality and inequality. A snapshot of the current ontology listing the available methods without prerequisites can be found in Figure 2.3. The available models are depicted in Figure 2.4. The figures show all methods and model types that are currently implemented. The theory behind most of them will be described in later chapters. What we can take from it now is the variety of different methods from that the system can choose. Many of the model types are

2. COP - A Robotic Perception System

so specialized that they will only work with a single localization method. The refinement methods can convert different model types into others.

2.3.1. Rule-based

The possibility to combine a certain methods with a model can be evaluated for every method. We can represent this information as a binary matrix B_a for every method $a \in A$ expressing the possibility to combine it with a certain model. Similarly, the second rule that restricts methods to certain sensor types is expressed as another binary vector b_s for every method collected in B_s . Those vectors limit the possible combinations to meaningful combination, since e.g. a CAD matching cannot be applied using a color model, while the shape model inference could.

For the search space, a different decision criterion is applied. We distinguish two types of search spaces: the current view as primary search space and a certain position with a spatial distribution as secondary search space (see Section 4.2.1). All methods can support both or only one of the search space types. That means, on the one hand, that certain search spaces forbid methods, if the respective search space is outside of the view of all sensors combinable with the method. On the other hand, secondary search spaces also limit the results of those methods that would be applied otherwise to the complete scene.

2.3.2. Experience-based

To evaluate a certain combination, a vector containing the scores is calculated for a model-method-sensor combination. We have three types of experiences stored in the vectors e_M , e_A and the matrix E_{MA} , which represent the evaluation of the models, the methods and the combinations of models and methods.

For a certain semantic type the evaluation for all methods e is calculated with the following formula, given the binary vector, that represents available model types connected with the queried concept b_M :

$$e = \left((b_M \cdot e_M)^T (B_a \cdot E_{MA}) \right)^T \cdot e_A \cdot (b_s^T B_s)^T \quad (2.1)$$

The operator \cdot denotes element-wise multiplication, and b_s denote if a sensors sees the current search space. The highest entry in e selects the method, which will then pick the relevant sensors and all necessary models, which can be more than one.

Internal Reward Structure

Positive answers are increasing all evaluation values of the selected method and models. Exceptions during the execution of the method with a certain model will set the evaluation of a model-method combination to zero.

2.4. Position-based Result Evaluation

In order to handle areas in the environment that are badly responding to sensors or algorithms, COP also contains an a-posteriori evaluation of result which is based on positions. Examples for such areas can be a Ceran stove or a steel sink, which gives bad reading for a ToF camera. Other examples are corners in a room with e.g. cables which challenges any edge-based computer vision algorithm.

To learn that results from a certain algorithm in such areas are less reliable than in areas with good preconditions, every verified or disproved result is stored in a kd-tree of positions with result qualities. This allows searching for a any new result if there were earlier results in the close environment. All results are stored in map coordinates with their evaluation (from 0=disproved to 1.0=verified) assigned to the algorithm they were acquired with.

The final evaluation is then calculated based on the current evaluation v_{now} and the n next results with increasing distances d_1, d_2, \dots, d_n and the corresponding saved evaluations v_1, v_2, \dots, v_n :

$$v_{final} = \frac{1}{2} \left(v_{now} + \frac{1}{\sum_{i=1}^n d_i v_j \left(1 - \left\| \frac{1}{1+e^{-d_i}} \right\| \right)} \right)$$

This final evaluation is used to prioritize lists of results and to filter out improbable results.

2.5. Feedback Loop in a Perception System

The possible errors that could occur in a system are manifold, while the points an error can be detected are limited. Inside a perception system a “second opinion” might be the only way to detect something is going wrong. Regarding the means of a perception system, this might be a different method applied on the same sensor data, the usage of the same method on different sensor data or integration over time or point of view.

All those methods are tediously better performing with the result of the earlier tries as initialization, which also lowers the probability of obvious contradictions. Anyways, an internal

2. COP - A Robotic Perception System

detection of false readings is only possible if there are methods with different error patterns for different conditions. But the initial problem, how to find out which methods have which error pattern without extensive design of those properties is pretty difficult to elicit.

Without external feedback, the proposed unsupervised learning tends to favor solutions with a relatively high false-positive rate. But false-positive matching can be handled easier by a highlevel system than false negatives. This is managed by allowing feedback on search results: Any hint for wrong perceptual results is reported back, so evaluation of the respective model-method combination can be corrected. Also all successful manipulations of perceived objects are reported as a success which will amplify the earlier reward for the detection and also to the cascade of previous perception primitives that created the models involved in the successfully fulfilled task.

2.5.1. Type of Feedback

The next level after the intrinsic error awareness is a reasoning level in the highlevel. There are several possibilities to compare a certain percept with physical, temporal, or even self-including models.

Depending on the used perception mechanism a highlevel system should be able to react in a specific way to different kinds of errors. An object can be missed, it can be hallucinated (false positive) or the result can be not accurate enough for a certain task. All three problems can lead to a failure of tasks, but are difficult to distinguish from observing only the success of execution.

Discussing at first grasping actions as the most relevant task for this work more detailed, a highlevel system can distinguish the three kinds of perceptual error:

- Any unpredicted collision far from the target object are a good hint for an object that was not detected even if it was there.
- Any collision in the direct neighborhood of the target object is a good hint that the perception of the target object was not accurate enough.
- On the other hand, if there is no collision at all even when the hand closes around the predicted position of the target object, this occurrence is most probably a false positive detection.

This reasoning of course assumes a perfect detection of collisions, which is unfortunately also based on a noisy perception. Luckily, this perception is made usually with a different kind of sensor, so that the reasons for erroneous percepts are different: Collision detection

2.5. Feedback Loop in a Perception System

is mostly based on forces that are applied on the actuators. This again is depending on the mass of the colliding object.

Another method besides the collision detection in the robots actuation would be a second glance at the scene best from a significant different point of view. This requires of course highlevel planning, to make sure there are no other known objects in the sight line, the scene and the lighting did not change significantly in the meantime and the model which were applied work for the new point of view or there are additional models which do.

The result can either verify the first percept or it will contradict, which is the fourth detectable error, which can be reported to COP.

Overall, COP supports five types of feedback: First and most important: the task succeeded, which means there was no detectable error. All others are errors: An undetected obstacle, an inaccurate localization, a hallucination and contradicting perceptions.

2.5.2. Cascade of Feedback

A common action pattern in an object manipulation process is the detection of object candidates, their verification or refinement, a pick and a place back, and a verification of the placed objects position.

In this scenario the internal process, which happens inside COP can be explained regarding the feedback. A perception primitive for detection creates a set of object candidates and will after execution wait for an evaluation. This is not possible until any action which can fail is executed. A detection for example usually does not fail, it either reports objects or no objects regardless if this result is correct. There is no possibility to judge this result.

Assuming, that the pickup action fails, a feedback will be given for one of the objects informing about the negative result. This feedback will contain the information which of the multiple detected objects was selected. The feedback will cause an evaluation of the model resulting from the refinement as well as of the initial model triggering the detection and to the methods used for detection and refinement. If the grasp fails without contact with an object, the position at which the object was detected gets a bad reward, too. Suppositional the manipulation succeeds and only the re-detection fails; only the models involved in the re-detection get a decreased evaluation.

2.6. Related Work

We present here a perception system for robotic manipulation. Similar objectives can be found in the following systems: A perception system for indoor environments is presented in [16]. It uses a two step combination of color saliency with a boosted image classifier in order to enable a robot platform to perform environment aware manipulation. These techniques are applicable to a special scenario and do not include a learning step.

The work of Kragic et al. [61] proposes a system that distinguishes between different visual modalities (monocular CCH requiring a CAD model, and a SIFT-based matching) and handles known and unknown objects. The system segments first using a combination of 3D and 2D information which can be seen in a typical example in Figure 2.5 on the left side, and then uses SIFT and the CAD information for a final and more accurate pose estimation, which can be seen on the right side of Figure 2.5.

The framework presented recently in [4, 37, 36] coordinates sensing with learning over time based on color segmentations, SIFT-based object localization, and CAD based object localization. This framework is tailored for real time self localization and uses different sensor types like force sensors and vision. An example how to use this system is depicted in Figure 2.6(a) showing ARMAR-IIIa picking up an object detected and tracked with a method combining a CAD model and color information.

This system was recently extended with the capabilities to explore new objects using active vision [141]. The new capability allowed the robot to capture a partial textured 3D model of objects in his hand.

Another system that is context and target aware can be found in [86]. It requires a completely modeled world including a 3D model for all handled objects and world modalities. Images from the cameras on a HRP2 can be seen in Figure 2.6(b), which shows possible matching on the top and the interpretations achieved by this system.

A system for manipulation of unknown objects was presented by [116]. This system relies on the visual recognition of grasping points which would require a large labeled database of objects to be handled. The work was continued by incorporating 3D data into the segmentation which precedes the classification step. The results were surprisingly good, while the possible generalization can be doubted.

Trujillo and Devy in [27] present a vision framework for the recognition of objects on a robotic platform. This framework uses several vision modalities to be robust for a large set of different objects. Unfortunately, the system requires a long modeling phase, and requires the acquisition of a large set of images which implies active movements of the robotic system.

2.7. Summary

After seeing the other state of the art systems, the properties of COP differ from the discussed related systems in the following points: COP is capable to learn in an abstract way independent from implemented methods, and it is transparently interfacing any underlying mechanism to a highlevel system over the semantic bridge. The internal selection mechanisms can provide performance and robustness at the same time.

Due to these properties, it can support a large list of specialized methods. A subset of the implemented methods will be discussed in the Chapters 4, 5 and 6. Those methods integrate well into the same framework while they have different interfaces and result types and semantics, but still can be used without knowledge about those differences on the point of view of a high-level system. Additionally, COP supports different kinds of sensors and also integrates well with different robotic platforms. The state of the art for supported sensors and what other robotic platforms are existing and what are the limitations of those systems will be discussed in Chapter 3.

2. COP - A Robotic Perception System

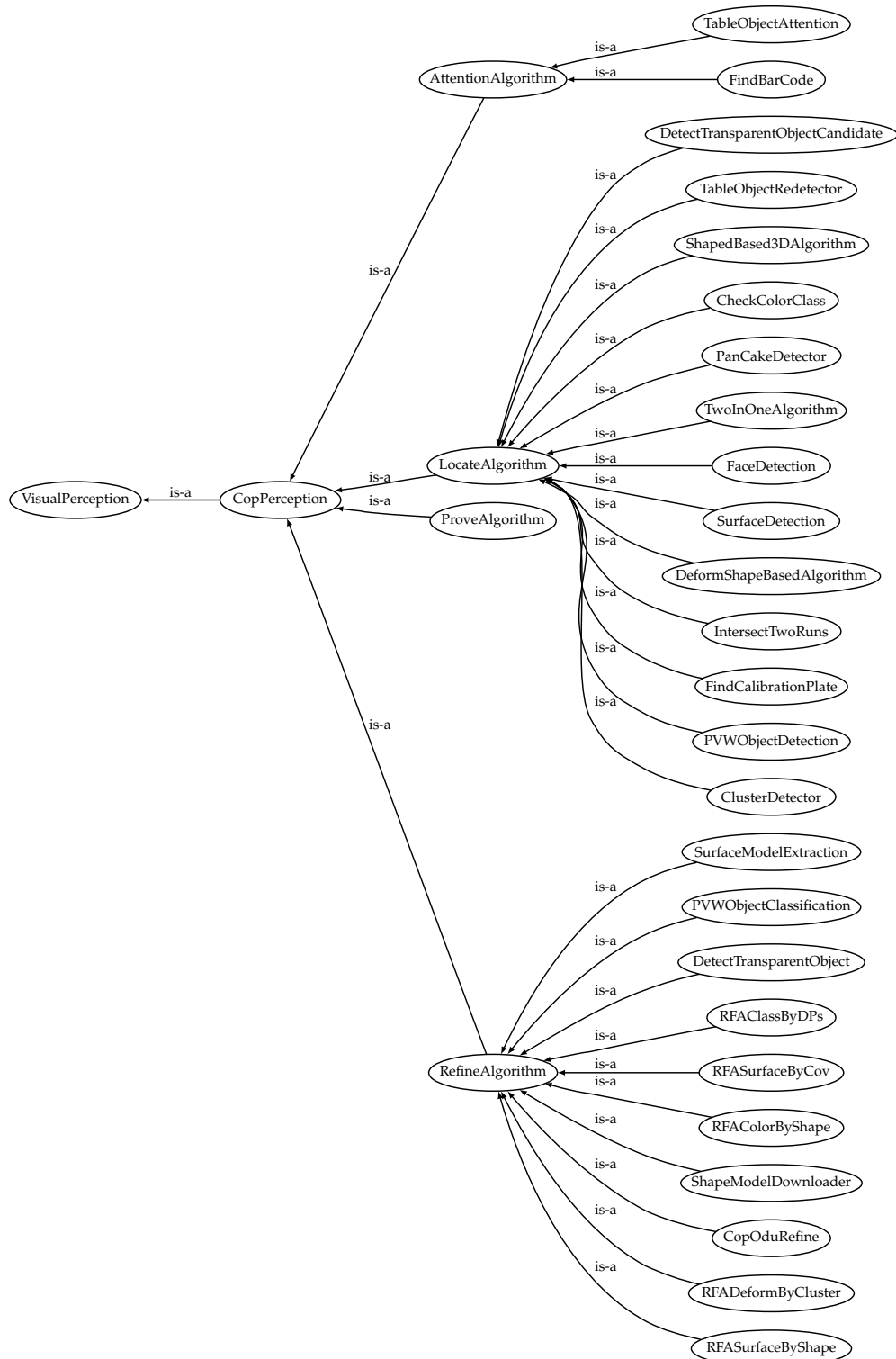


Figure 2.3.: An overview of the currently implemented methods in COP.

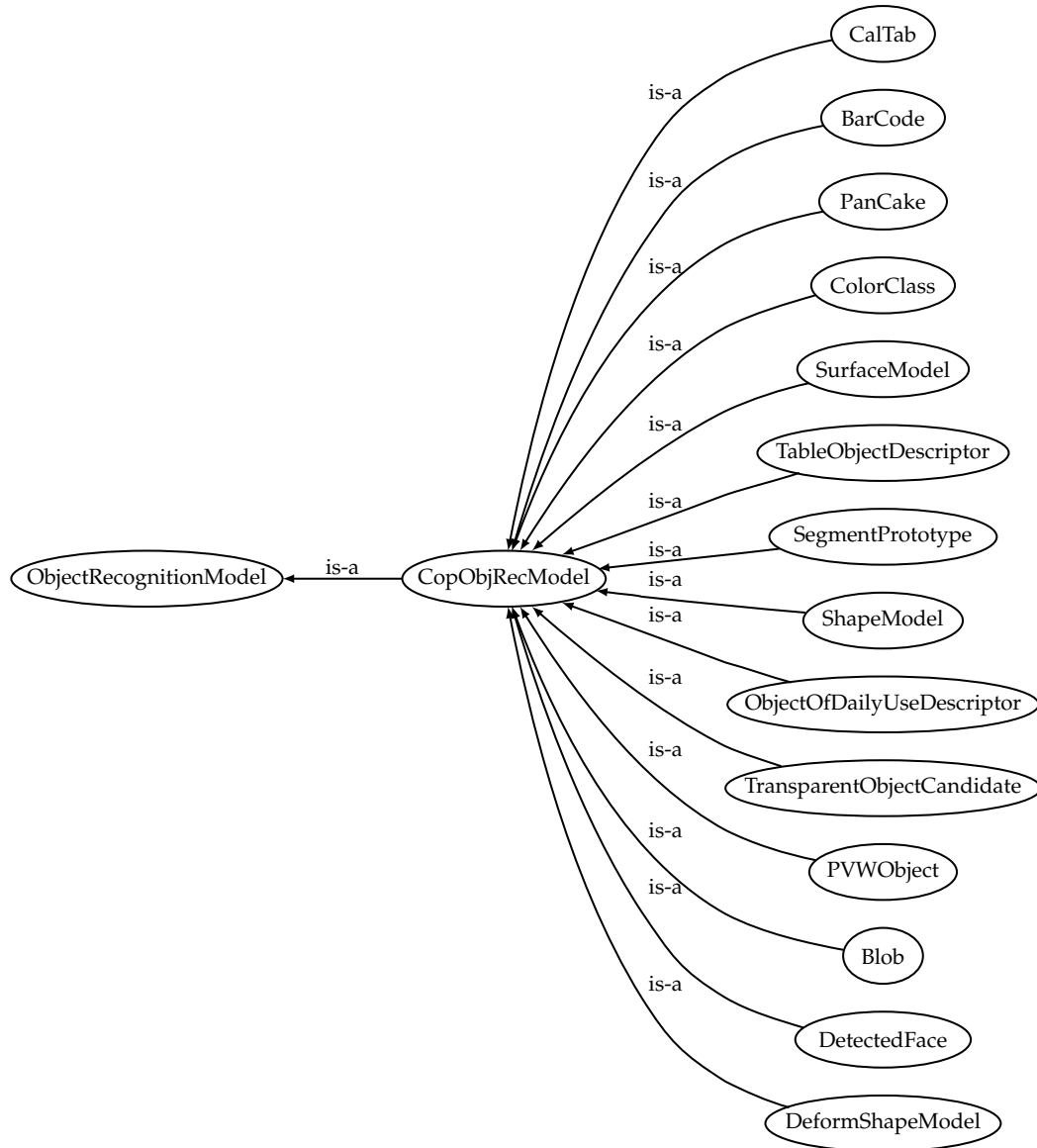


Figure 2.4.: An overview of the currently implemented model types in COP.

2. CoP - A Robotic Perception System

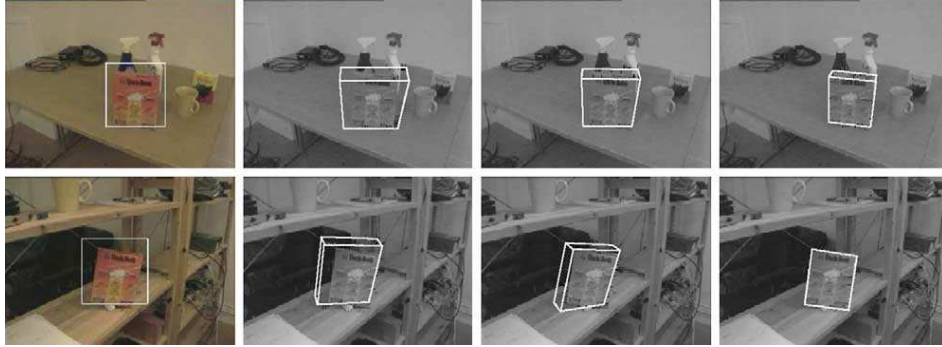


Figure 2.5.: An segmentation and three iterations of a monocular CCH to a final fitting of the object, taken from [61].



(a) Grasp execution with the humanoid robot ARMAR-IIIa using the developed object recognition and pose estimation system [4]

(b) The perception framework of the Tokio university uses CAD models [86]

Figure 2.6.: Vision systems from KIT and Tokio University

3. Robotic Hardware for Perception

Capabilities of robots are nowadays various enough to fulfill complex tasks, but they are still not general enough to be capable of solving generic tasks. Grounded in current sensors and manipulation hardware, the limitations to the capabilities are overcome by domain and task restrictions, or hardware multiplication. How sensors and robotic hardware are applied in the current development of service robots is discussed in this chapter.

3.1. Sensors for Autonomous Robots

For gaining autonomy, a robot must be capable of perceiving its world. In the developmental robotics, a large variety of sensors is in application. From a theoretical point of view, the sensor development is advanced enough to provide all information which is required in the setup of a household assistant robot. Practically, the sensor answers are still very context dependent, which leaves space for new developments in the sensor and their interpretation.

The most popular sensor types used in robotics are CCD based cameras for visible light, followed by active LASER or infrared based time of flight measurement techniques. A newly spreading sensor is the Microsoft Kinect sensor, which is a combination of a structured light sensor and a camera. We will discuss the properties of those sensors as far as they affect the usability in the household assistant scenario.

Additionally, there are other sensors used for pressure, force and torque measurements or just internal position measurements, which will be used at some points in this work for world perception. Those sensor types will not be discussed here, but it should be mentioned that they are complementary and well combinable with the visual sensors discussed here.

3.1.1. Cameras

A standard camera for visible light is relatively cheap, and creates images that a human can usually understand immediately. This is especially useful for developers working with this kind of camera and it also allows a human-like experience. Additionally, for this kind of images, the research in object and scene recognition methods is relatively advanced. In

3. Robotic Hardware for Perception

theory the results propose that the available methods are already capable of most of the atomic perception problems that have to be considered in this scenario.

Problematic for the visible light cameras is, that they have a limited dynamic and depend heavily on additional light sources. The perceived colors depend on external light sources, and the differences of the same perceived scene in full spectrum of daylight and limited spectrum of halogen office light is immense. The differences can be seen in Figure 3.1 which contains a similar shot from a robots camera of the same table once in daylight and once in halogen light.

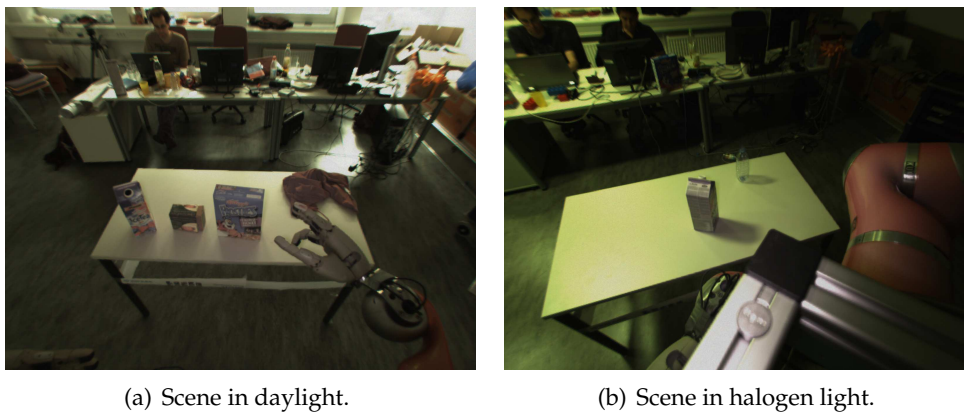


Figure 3.1.: The two pictures above show the impact of the environment lighting on readings of a RGB camera.

Usually the affordable cameras are restricted to a 3-byte intensity resolution with different filtering being able to receive all light with wavelengths in the spectrum visible for humans. Usually those cameras return their image separately in red, green and blue (RGB) channels. Unfortunately, in human life glass and other transparent materials play an important role which will not reflect much visible light. This implies that a robot also needs to be capable to perceive such objects, which is very challenging given only standard RGB cameras. If the objects are not transparent, but unicolor or textured, this color allows an easy identification of those objects. Although, the effects of different light sources shown before makes the identification by color challenging again. The third important clue humans use to identify objects or their functionality is the shape of the object or the shape of important subparts. In RGB images shadows or illumination changes appear on objects, on which they creates edges without geometric grounding. In order to allow the analysis of edges and surfaces by this clues, first the problem of distinction between geometric and shadow or texture edges has to be considered.

3.1.2. Stereo Setups

Especially the last effect of texture-imposed edges versus geometric edges can be overcome with a stereo setup. With a careful calibration of the relative position between at least two cameras, the depth in an image can be estimated. The depth estimation depends usually on a search of correspondences on epipolar lines. This calculation is theoretically relatively simple, but requires a lot of computation. The simple versions of the stereo calibration can be parallelized and can be implemented on FPGAs as it is done on Stereo-on-Chip-Cameras, like the STOC previously mentioned on TUM-Rosie.

The problems of this technique on the other hand appear on surfaces without any texture: the matching along the epipolar line is ambiguous, since there is no definite optimum for the matching of two points on such surfaces. Additionally, this technique suffers from structural defects at geometric edges which might be the most interesting information in the reconstructed 3D. The data on the edges is either smoothed by the process or there is a gap around the edges in the data.

3.1.3. LASER Range Scanner

LASER range scanners use a sweeping LASER beam to measure the time until the reflection of the beam returns to the sensor. This leads to several properties that are useful for perception: Generally those LASER range scanner return the most accurate distance measurement currently available. Unfortunately, there is only one measurement at a time per sensor. This means, to obtain scans of a greater volume, a significant amount of time is required in that the sensor must be moved in order to see the complete volume (sweeping/tilting movement or similar). This makes it difficult to deal with changing environments. Additionally the LASER beam is often colored red or even infrared light, which causes glass and some sorts of plastic to be invisible, while metal and other reflective materials disturb the measurements significantly.

3.1.4. Time of Flight Camera

Similar problems apply to the so-called Time-of-Flight (ToF) cameras, since they work with similar wavelengths to produce the flash that is used for distance measurement here. A flash is emitted and the passed time of the reflection is approximated on an area sensor by measuring the phase shift of the returning wave. This gives low resolution images, that have besides intensity also depth information. Unfortunately, this depth information is of lower quality compared to the LASER range sensors working with a single LASER beam.

3. Robotic Hardware for Perception

Since this sensor will be used for many experiments in this work in the following, a detailed analysis of the properties of the camera Swissranger 4000 (SR4k,[84]) is shown to analyze noise in the measurement of different surfaces in a domestic environment.

Estimate Noise Level in Range Data

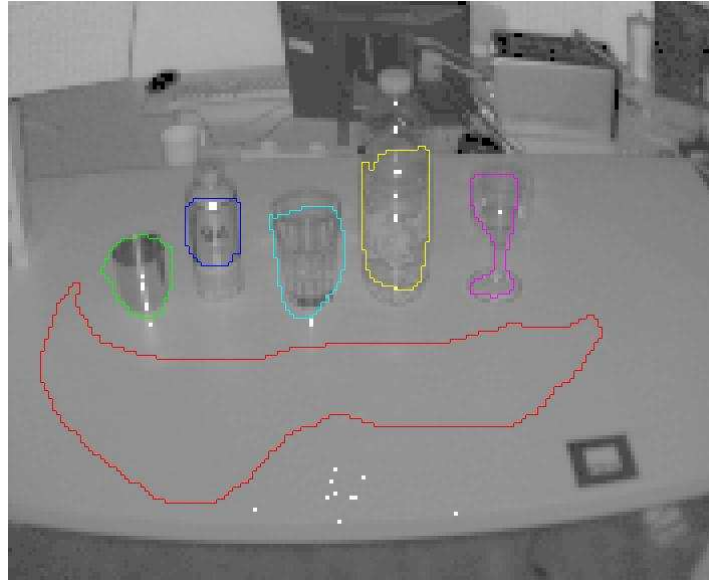


Figure 3.2.: One of the sample scenes with the manual segmentation used to estimate the noise level. The inclination of the sensor to the table plane was approximately 45° .

In this discussion we want to estimate two types of noise: First, the intrinsic white noise and second the structural miss-readings depending on geometric structures and surfaces.

Measured with a SR4k, Table 3.1 shows the influence of different materials on the deviation of subsequent observations of a static scene. The experiment took scenes like displayed in Figure 3.2. The data contains approximately 1000 observations with an observation frequency of 1 Hz. A hand-made segmentation allowed the distinct measurements for the objects. The deviation per pixel as well as the mean of the pixel-wise deviations for one segment can be found in the table.

The data shows that given a good segmentation of the table and enough valid measurements, the height of the table can be estimated up to accuracy below a centimeter. On the other hand, there are several problematic materials that might influence also the depth estimation of a table. Glass, for example, appears flat on the table, so it would be classified as part of the table. Still by the view through the glass the noise level changes significantly and too

Object	Deviation in z (in m)	Mean z (in m)	Dev. in Intensity	Mean Intensity
wooden table (white paint)	0.0039	1.154	69.47	7262.29
Chinaware, mug	0.0048	1.279	115.54	7555.18
Paperboard, box	0.0050	1.138	114.35	7463.31
Paperboard, cylinder	0.0051	1.225	115.87	7639.08
Metal cup	0.0140	1.309	341.80	5665.42
Metal spray (printed)	0.0124	1.331	261.49	5752.84
Glass	0.0105	1.381	261.48	5474.76
Thick Plastic, wine glass	0.0075	1.421	137.70	5387.81
Thin Plastic, bottle	0.0105	1.379	236.55	5597.33

Table 3.1.: Intrinsic deviation for average pixel-wise noise for an inclination of the sensor toward the major plane of the objects of 45° and 1000 subsequent readings of the SR4k on a stable scene.

many of those objects will have negative influence on the accuracy of the height estimation of a table.

Additionally, the data shows a significantly worse behavior of the measurements for metal and glass in intensity and depth.

3.1.5. Structured Light Stereo

The low resolution can be tackled by replacing the complex time of flight measurement with the previously mentioned stereo camera setup, given there was texture on the objects. The fact of texture on the objects can be simply enforced by projecting patterns on the scene. This combination of projection of a static pattern with a stereo setup delivers similar quality like a ToF sensor with a significantly higher resolution. Unfortunately, also this technique has some drawbacks, like structural measurement errors on geometric edges and no measurements in areas one of the two cameras is not seeing.

Microsoft Kinect Sensor

A very exciting example for a structured light sensor is the Microsoft Kinect Sensor. In this case the structured light is produced by a LASER lighting through an optical lattice [115], which provides a multi-resolution pattern with sharp dots at various depth. This effect allows this sensor to be faster and more accurate than most available sensors for a fraction of the usual price. With this technique one camera is enough to estimate the depth, since the pattern projected by the LASER can be well predicted and detected dots can be identified directly.

3. *Robotic Hardware for Perception*

3.1.6. Sensor Combinations

The best performance can be achieved by combinations of different methods. E.g. we can get the accuracy of a tilting LASER range sensor, if we additionally use a ToF camera to be able to deal with a changing environment faster. Additionally we can overcome the resolution problems of a ToF camera if we mount a set of stereo cameras next to it. These are the reasons for the sensor setup of TUM-Rosie, which serves quite well for the purpose of object recognition. Other examples for successful sensor combination are the DLR 3D Modeller or the PR2 Sensor Head of Willowgarage (also on TUM James).

PR2 Sensor Head

The collection inside a PR2 sensor head contains 5 cameras and one projector. The projector can be used to illuminate the area in front of the robot with a static pattern, which allows robust depth estimation using stereo. This so-called structured light stereo can be performed with two different pairs of cameras inside the sensor head: the foveal narrow stereo cameras and the wide angle stereo pair. Additionally the sensor head contains a high resolution camera with approximately 5 Megapixel resolution that allows the analysis of texture of objects and any kind of image processing that requires high resolution.

3.2. Current Robots for Perception in Human Environments

The capabilities of robots can be best surveyed by an introduction of the state of the art platforms used for service robotic research. The list of robotic platforms is restricted to such projects that perform research in object recognition. Namely, we will go over the following platforms to STAIR of Stanford University, as well as HERB of Intel Research and the CMU robotics institute, the DLR robot Justin, the AMAAR project at Karlsruhe Institute of Technology, the research with HRP2 at the Inaba Laboratory at Tokyo University, the robots at LAAS-CRNS in Toulouse and the Care-O-bot of the Fraunhofer Institut IPA in Stuttgart.

Generally, all current robotic platforms have limited motion capabilities and are equipped with a limited sensor set including in most cases 3D sensing. The limitations in the motion capabilities express themselves often in the tasks which are addressed in the published work. This implies that often lack of generality in the proposed solutions results from hardware limitations.

3.2. Current Robots for Perception in Human Environments

3.2.1. Stanford - STAIR

STAIR is a project to implement a robotic platform capable to fetch and deliver items, tidy up or prepare a meal in a normal kitchen. The research focus lays on perception mechanisms for manipulation. The hardware platform consists of a wheeled base with a vertically mounted arm with a gripper and a set of sensors on top of a longer fixed arm.



(a) STAIR opening a door. [101]



(b) HERB from the Robotics Institute of Intel Research and Carnegie Mellon University.

Figure 3.3.: Robotic platforms from Stanford and Pittsburgh.

One of the achievements reached with STAIR is autonomously opening of doors, see Figure 3.3(a), which is part of a larger task to go around in office buildings and find all objects of a certain type.

Another example for results from this project is a method for automatic grasp point selection based on images even for previously unknown objects based on features learned from annotated objects [116]. Given enough annotations and similar enough objects, it seems an interesting way for grasp point selection. The platform is limited to manipulate on desk height and a bit below, and addresses also major problems in the working place desktop setup. The most famous demo which was presented by the group of Andrew Ng was to search in a room for a stapler and deliver it to a predefined place [81]. Interaction with humans as well as laser based sensing is not a research focus.

3.2.2. HERB - Intel Research and Carnegie Mellon University

Hardware wise similar to the STAIR project, HERB is the home exploring robotic butler of Intel Research and the Carnegie Mellon University presented in [123]. It is capable of grasping textured objects and navigates safely using Visual SLAM through its environment.

3. Robotic Hardware for Perception

It consists of an omnidirectional platform for movement and has a 6dof arm mounted on top of this with a Barrett hand for manipulation. The work done in object recognition [106] as the works in grasping are showing how little hardware is required to achieve good results [19].

3.2.3. DLR Justin



(a) ARMAR III opening a fridge.



(b) Rolling Justin from DLR [144].

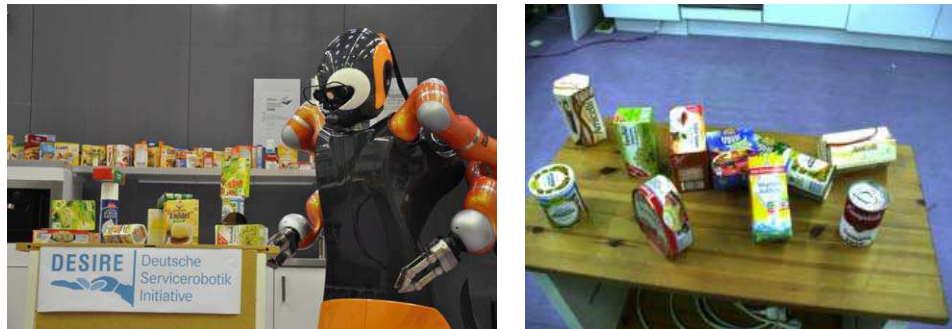
Figure 3.4.: KIT and DLR robots.

The DLR has the robotic platform that was provided to the Desire project also in house with the name Justin. The enormous capabilities of this platform were demonstrated in several occasions: The arms are fast enough to catch a ball in the air [6], and the entire platform is accurate enough to pour liquids, even active liquids like wheat beer. The integration to a more complex system was not pushed here at DLR directly, since the hardware development and space application are more in focus than household activities. This was emphasized more at the Desire project, which uses parts of Justins hardware.

Desire Project

The desire project was meant to combine the research on service robotics from a large set of institutions and companies. The participating institutions are among others Kuka, Schunk, Siemens, Universität Freiburg , UniversitätBielefeld, the DLR and the Fraunhofer Instituts IAIS and IPA.

3.2. Current Robots for Perception in Human Environments



(a) The service robot used in the Desire project. [21]

(b) A view of a cluttered scene that was used to evaluate the perception mechanism of the robot.

Figure 3.5.: The Desire robot and a sample scene from a demonstration.

Regarding perception they presented a well integrated vision system for known object based on texture features (SIFT) located via stereo cameras making use of 3D segmentation techniques. The latest advances contain work in active perception [21]. Using a database of 100 exhaustively modeled objects it was shown that even cluttered scenes could be reliably reconstructed. The robot named Desire can be seen in Figure 3.5(a). Figure 3.5(b) show a scene from the point of view of the robot. This scene can be analyzed at once by identifying their histograms of SIFT features.

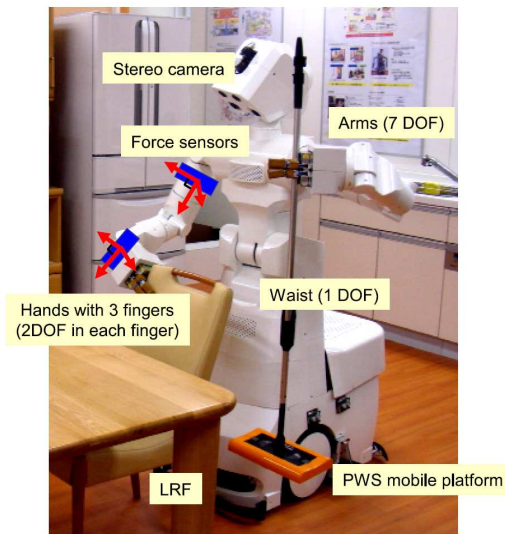
3.2.4. KIT - ARMAR III

The ARMAR project at the Karlsruhe Institute of Technology tries to build a robot with human like capabilities. The project targets on the construction as well as on highlevel control of the robot. Figure 3.4(a) shows ARMAR III during a demonstration showing the capability of opening a fridge and getting items out of the fridge. During the opening, the hand and the fridge's door are tracked visually and the forces applied to the handle are controlled using the visual feedback as well as the force measurements in the arm. The robot localized itself visually; see Gonzales et al. [35]. The robot relies for object recognition mainly on fast SIFT implementations.

3.2.5. Tokyo Univ. HRP-2

Figure 3.6(b) show the HRP-2W that was enabled by the researcher of the Inaba-Lab at the Tokyo University to wash dishes or fold towels. The robotic hardware allows dexterous manipulation and is controlled by Lisp program, which allows failure recovery and dynamic replanning. Thanks to this controller the robot is able to show complete demonstration action sequences, like pouring liquid in a vessel, checking for eventually caused messes with

3. Robotic Hardware for Perception



(a) A daily assistive robot, presented in [145]

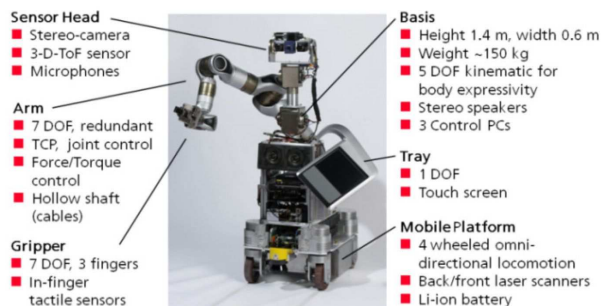


(b) The HRP-2W, the wheeled version of the HRP-2 with its sensor set [51].

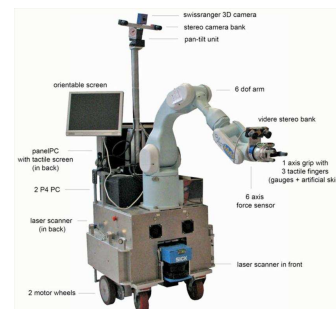
Figure 3.6.: Selected robots at the Tokyo University

the liquid and cleaning any produced mess. The perception capabilities are based mostly on stereo vision, and include matching of projected contours of 3D objects using a method based on particle filtering. An even more advanced sequence of action was shown by the researchers in Tokyo with the robot shown in Figure 3.6(a), which cleaned up a room including picking up clothes, putting them into a washing machine and wiping the floor.

3.2.6. IPA Care-O-Bot 3



(a) The Care-O-bot 3 and its hardware setup.



(b) Jido, a robot designed and used at LAAS in Toulouse.

Figure 3.7.: Robots at the IPA and LAAS.

Thought to be a ready-to-sell robot butler, the Care-O-Bot 3 can be seen without casing in Figure 3.7(a). It is with only one arm less human like, than some of the other mentioned

platforms, but is composed of industrial proved parts and can be counted as a reliable hardware platform. The software currently enables the robot to pick up drinks and serve them to people avoiding meanwhile other moving persons. The sensor setup contains a stereo setup, a LASER range sensors and a ToF camera.

The tasks which are performed are mostly preprogrammed, including the map and the object knowledge. This allows on the one hand the robust behavior, but imposes additional complexity on extending the capabilities of the Care-O-Bot. Currently the robotic middleware is replaced by ROS, which might help to extend the capabilities.

3.2.7. LAAS - Jido

Mainly developed for human-robot interaction, Jido is a robot that is used for passing objects to humans, see Figure 3.7(b). The perception on this robot is implemented by an outside-in tracking system installed in the laboratory. The research that is done on this platform is focused on navigation and action in the presence of humans, e.g. [121].

3.2.8. PR2 of Willowgarage

The robotic research platform PR2 was developed by Willowgarage in order to provide a common research platform for robotic software development. The target is to speed up research to reach the breakthrough in personal robotics faster. In order to achieve this, a robust robot was developed that has already a large set of drivers, controllers and perceptual functionality to perform a bunch of basic tasks out of the box. Willow garage also granted PR2s to Stanford, Freiburg, Berkley, Tokyo, BOSCH Research, Leuven, Gorgia Tech, MIT, University of Pennsylvania and the University of Southern California.

3.3. Summary

This chapter showed an overview of the research in the last years towards service robotics and perception systems. By going over these systems, the most interesting challenges and hardware imposed difficulties are shown. The variability of the appearance of robotic platform is very high. Also the different applications they are made for vary a lot as do the capabilities regarding manipulation and perception of the platforms. It can be additionally stated, that TUM-Rosie and TUM-James are relatively highly developed platforms, which also have competitive sensor setup.

3. *Robotic Hardware for Perception*

Based on this conclusion, the next chapter will give details for the implementation of perception methods on such high-end platforms.

Part III.

Perception Tasks in Domestic Environments

4. Detection and Reconstruction

A robot in a human living environment is confronted with a large set of moving and changing subjects, objects and scenes. In this work several challenges are pointed out in this scenario and solutions are proposed for some of them. This chapter starts with the task of localizing a camera or a robot in space in a known environment, continues with the task of detecting objects, and ends with the task of 3D reconstruction. We want to show methods for different kinds of objects, depending on material properties like transparency.

4.1. Perceiving Environment

The first step of a robot in an environment is the self localization: Even if for most service-robots this task is in most cases only a three dimensional problem, since the height from the floor is mostly known like the rotation of the sensors against the floor plane, we still want to focus on a full 6D approach. Before we will go into detail of our approach, which we tested on a mobile camera wear by a human on his head, we want to give an overview over the field of visual synchronous localization and mapping (SLAM) and localization in video sequences.

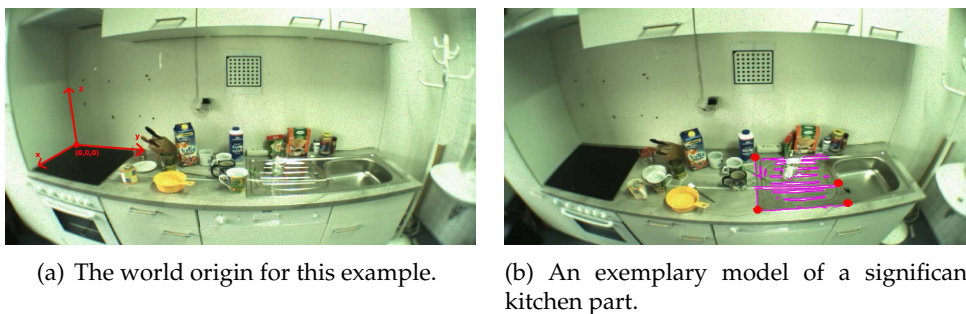


Figure 4.1.: Even in much distorted images, an absolute 3D localization is possible, even without having any prior about the cameras position and calibration.

4. Detection and Reconstruction

4.1.1. Related Work

Robots that localize themselves often use Monte Carlo Localization, usually variants of the so-called Adaptive Monte Carlo Localization (AMCL). This method was described in [129] and methods based on this early work are the state of the art in robotic self localization. Usually, the sensor measurement used for this method is a static LASER sensor mounted near the floor on the robotic platform. The localization hypotheses that are necessary to make use of the Monte Carlo Localization are based on a map that usually was generating before. The most common approach for this is GMapping [38].

The general idea behind map building and localization in a static map can be extended to a visual approach, the so-called Visual SLAM, which is a widely discussed topic, improved lately by works like [58]. Most such approaches use local features, which require a rich texture at walls of rooms to work reliable. In the kitchen scenario most textured objects in our environment are the movable objects. So any completely autonomous approach would have to work in an empty kitchen. This assumption of having an empty kitchen is too strong to be applied to this work.

More interesting for the service robot is the 3D Mapping , e.g. presented by Rusu et al. in [112]. This technique can abstract easier from the movable objects, since the mobility correlates heavily with the size of the objects, which is in 3D information the most reliable. Anyways, in order to locate a camera in respect to a 3D structure an edge template can also answer this question.

4.1.2. Scene Localization and Tracking

Beside general Visual SLAM, if the problem is reduced to a known and non-changing environment, the selection of landmarks can be reduced to a few distinctive points that can be seen often. This fact can be used in a setup without a robot but with a head mounted camera with a gaze tracker to localize the camera's positions.

The same idea can help the robot to localize itself relative to very specific landmarks with over-average accuracy. Or with a controlled environment even a calibration with the environment is thinkable.

The technique we use to localize landmarks in an indoor environment is restricted to planar parts which might be in a kitchen e.g. walls or working surfaces. It does not require distinctive texture but builds a complete model of all edges visible in a certain area that are assumed to be planar. One camera image might be enough to learn the model.

The basic algorithm we use was introduced by Hofhauser et al. in [45]. We wrapped this algorithm into a system selecting the adequate template depending on the robots position estimated over odometry and AMCL applied on laser scans, which give already localization accurate up to several centimeters. In order to extract such a landmark, we first need planar substructures in the world. This was applied in [126].

Plane recognition

Given a point cloud acquired by a sensor like the ToF camera SR4k of a tilting LASER range sensor we can detect planes in the environment. In order to estimate planes we can distinguish four different methods, which apply to different situations:

- A Sample Consensus approach over full 4D plane equations
- A Sample Consensus with normal restrictions (e.g. extract walls by using points with normals that are orthogonal to the floor)
- A maximum search on a histogram containing the height (e.g. extract tables and floor candidates)
- A filtering by a local normal that is estimated by calculating the SVD on the covariance matrix of the distance vectors to the mean of the point cloud

Depending on the size of the relevant data set and the expected result, a certain technique should be favored. If we can extract such a plane in a scene, we can learn a template for this plane and register it later on with the current camera view. Examples for this technique will be additionally presented in the next Chapter in Section 6.1.3. For table detection we use the third method, the maximum search on the height histogram, since it is the fastest method, which is also robust against noise due to the restrictive model.

4.2. Detecting Objects

In order to explore a scene or to find a special object, a mobile robot might need to generate hypotheses where to start looking. Provided that a highlevel system already localized the robot well, a direction to look at is granted. A query from the highlevel system to the perception system is then posed regarding a known volume like on a table, in a cupboard or on the floor. This allows us to apply several methods, e.g. the following:

4. Detection and Reconstruction

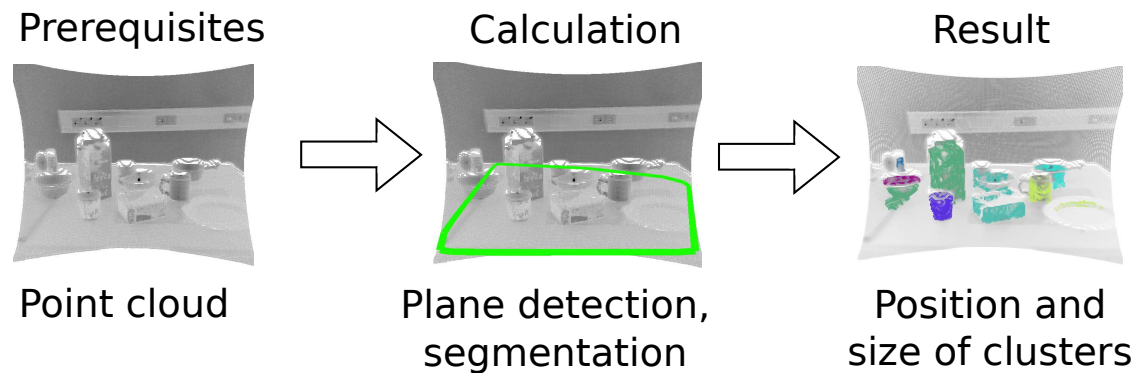


Figure 4.2.: Prerequisites, Calculation and Result of point cloud clustering.

4.2.1. Point Cloud Clustering

In a 3D point cloud we can detect and subtract known substructures like tables or the floor or parts of a cupboard, mostly by extracting planar or other known parts. All remaining points in the volume of interest can be clustered into object hypotheses. Those object hypotheses can be compactly represented by a point probability distribution, or by all points in the cluster. See Figure 4.3 for an example of a scene segmented with this method. The volume described by the distribution of the point of the objects serves as secondary search space.

Especially over planes like tables or a shelf this method works well. See Section 4.1.2 for the detection of object hypotheses. Even without having a structural map, a horizontal plane can be easily detected and used as a supporting plane to segment object standing on top of it.

There are several critical parameters involved in this clustering: When a cluster must be split, what a significant size is and what are the thresholds to consider a sensor reading as an object and not noise. With enough context knowledge, those parameters can be calculated or estimated online given the current task. In current literature, those parameters are mostly hard coded for a certain scenario and can be maximally adapted on failure in interactive way [96].

Approximate Minimal Height

An object can be recognized if it sticks out of the table significantly more than the usual noise in depth measured on a table. In this case, all points can be selected to be part of the table or of object in linear time. Usually the noise of a table depends highly on the material and on the angle of observations. Sensors like LASER range sensors and ToF Cameras tend to have more noise on steep views. Given a certain table which we know the internal noise level, (see

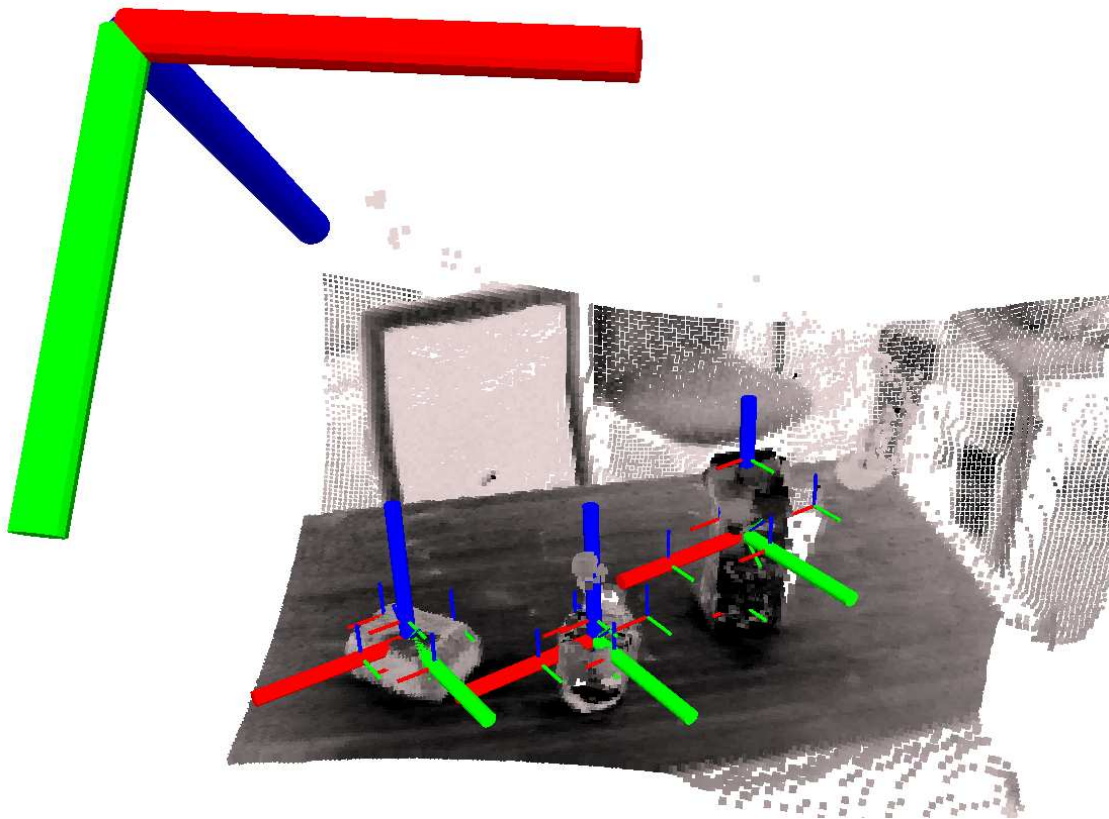


Figure 4.3.: An image of a ToF camera with an initial segmentation shown by the large coordinate systems, and approximations of the point distribution displayed with small coordinate systems.

Table 3.1) the minimal height that can be robustly detected is usually twice the deviation in the depth measurement for the table material. The angle of observation depends mainly on the height and the range of the robot and can be considered constant for one platform.

Initial Segmentation

For most 3D data it can be assumed that the local neighborhood of the voxels is known. This allows the usage of a so-called range image, which we can use to identify connected components. If a threshold is applied to the z component of all points in the 3D data the rest is tested for connections. This test just creates regions of all the voxels, which have a neighbor and also are selected as being above the table.

Each connected component is an initial segmentation candidate and its mean and deviation is calculated. Those values are passed to the following pre-selection method.

4. Detection and Reconstruction

Pre-selecting Object Candidates

An object candidate has to fulfill several conditions before the splitting process:

- It contains at least 0.25 % of the initial points
- It needs at least a certain amount of particles per m^3

If a candidate passes the pre-selection it is checked for the possibility to be split. A valid minimal density was measured at five million measurements per m^3 for SR4k and Kinect.

Splitting Object Candidates

The mechanism requires a dimension along which the splitting will be performed. This dimension can be ideally calculated by a PCA on the points that result in the dimensions with the highest entropy. Alternatively, all coordinate axes can be chosen, to simplify the selection of this dimension.

The values of the distances of all points in this dimension to a center are put into a histogram. This histogram is smoothed as long as the number of points in the histogram allows it or until the number of significant maxima is reduced to a low enough number. All of those maxima are considered as a potential mean value for a new candidate and the initial candidate is split under one condition: Two neighbored maxima have at least a distance over the noise level of the sensor (see 4.2.1). If two neighbored maxima do not fulfill this, they are not split and the weighted mean of them is compared with the next maximum.

Merging Object Candidates

The condition to merge two clusters is that the Mahalanobis Distance [42] is smaller than 1% or the final distance is smaller than the threshold for splitting. This is tested for all object candidates. All other objects are returned and sorted by their size in voxels as a quality measure.

4.2.2. Intensity Based Segmentation Methods

Supposed that an unstructured object is in a known or predictable environment, it can be recognized using intensity or color cues. An example for such a task is the detection of a newly generated pancake in a pan. Given the assumption the pancake is in the pan, it has a significantly different color than the pan. The only exception might be a burned pancake in a black pan.

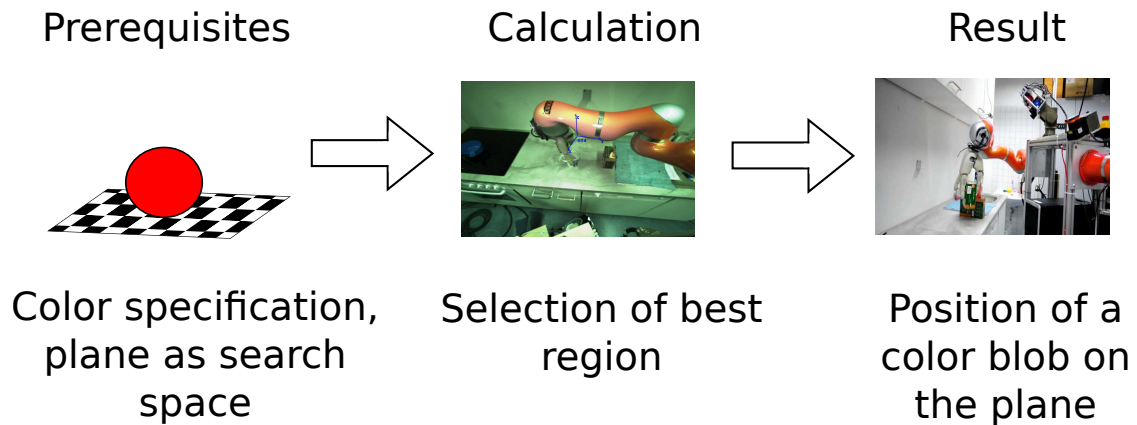


Figure 4.4.: Prerequisites, Calculation and Result of the intensity based segmentation methods.

In this example, all non-pan like colors can be segmented if the position of the pan is well-known. The position of any structure that differs from the pan can then be determined in relation to the pan. For segmentation a dynamic threshold is used that distinguishes dark and bright area in relation to the local neighborhood inside the known region of interest.

As extension of this informed segmentation, a color classification based on [140] can be applied on a previously segmented object in order to create a histogram of colors that can be compared to a stored or predicted set of colors. This can be used to recognize objects based on human descriptions of colors like "red", "green" or "blue".

4.3. Detecting Features of Tools

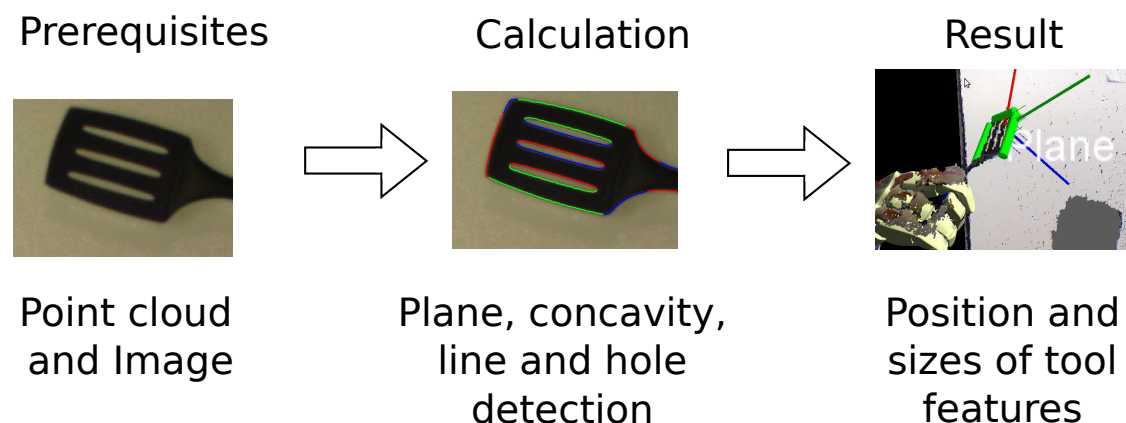


Figure 4.5.: Prerequisites, Calculation and Result of detection of tool features.

More specialized than the detection of present objects is the detection of certain features. If a

4. Detection and Reconstruction

geometric part of an object like a plane, which was discussed before, bears functionality, the extraction of features that stand for such parts can lead to implications to possible usage.

4.3.1. Symbolic Tool Representation



Figure 4.6.: The set of tools we analyze here, the two tools on the left side were used by TUM-Rosie for food preparation.

The features **SharpEdge**, **FlatSurface**, **Concavity** and **Hole** were chosen based on the requirements we found for tools used by TUM-Rosie for food preparation. We additionally define the feature **Handle**, which the robot already has in the hand. We can describe the tools which we consider in this paper using mainly the transitive predicates `properPhysicalParts` taken from the KNOWROB system [127]. Additionally we use simplification for numeric relations like `min` and `max` to express a quantification that requires more than e.g. `exists` or for `all` quantifier. By the features and the predicates we can describe the tools we use here (see Figure 4.6) in the following way:

```
#Definitions:
Class: HandTool
  properPhysicalParts min 1 Handle
Class: Blade
  properPhysicalParts min 1 SharpEdge
  properPhysicalParts min 1 FlatSurface

#Tools used in this paper:
Class: Spatula
  SubClassOf: HandTool
  properPhysicalParts min 1 Blade

Class: Spoon
  SubClassOf: HandTool
  properPhysicalParts min 1 Concavity
  properPhysicalParts max 0 Hole

Class: Skimmer
  SubClassOf: HandTool
  properPhysicalParts min 1 Concavity
```

properPhysicalParts min 1 Hole

This ontology specifies how to relate tool names to sub-parts of the tool. To enable a robot to autonomously perform actions with such tools, some additional knowledge is required, which describes the subtasks in terms of the detected features or parts. In combination with this ontology, we can then derive actions from statements in natural language like the following: 'align the blade with the pan' or 'push the spatula down on the pan'. Such an explanation can be transformed into a control rule given we know what is the blade of a spatula. So it is required to extract features that can localize the features blade and concavity, to be able to understand such an explanation. These features might be able to express more tools than those three, but we had enough examples to evaluate the method on those three classes.

4.3.2. Visual Tool Analysis

We use in this context a Microsoft Kinect sensor as a 3D sensor. From this sensor we just take all points which are sticking out of the hand and are not the robot's hand. These points form the first hypothesis for the shape of the object. Then we try to fit a plane to it and analyze the structure of the distances of the points to the major plane. In this step we can distinguish planes from concavities and we get an estimation of the 3D position of the border points which we can use to integrate with extracted lines.

Sensors like the Kinect have problems with shiny objects, so we avoid them in the test, assuming that these problems can be overcome by future sensors or software for 3D reconstruction. Most current 3D sensors have additionally problems with geometric edges so we use additionally a camera with higher resolution, which is calibrated to the Kinect and the robot. We extract edges in the images of this camera and fit lines to them given they are nearby the borders of the 3D body. We project the start and end points of 2D edges onto the plane which was extracted in the 3D data.

With an object in the hand, the robot can control the background as well as the observation angles. So, in this work we assume both to be already optimal and we work only on this optimal frame. Optimality in the observation angles can be based on the largest visible area of the tool, which can be easily achieved automatically given a 3D sensor that can measure the tool.

4. Detection and Reconstruction

Flat Surface Extraction

In order to fit a plane, we perform first a simple thresh-holding in space to get points that belong most probably to the tool. The thresholds are known since we approximately know the relation of the potential tool to the robots hand.

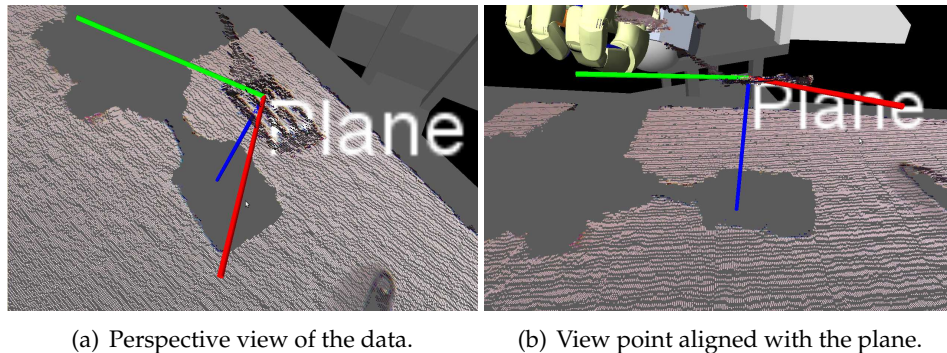


Figure 4.7.: A point cloud acquired with the Kinect sensor with a coordinate system showing the orientation of the extracted plane. The blue z-Axis represents the normal of the plane.

We take all points in the object and perform a SVD on the matrix containing all points as rows in the matrix in order to use the resulting first eigenvector as plane normal. This simple analysis resulted for all tools in a reasonable approximation of a plane that at least intersects the tool at a relevant position. For planar tools the detected plane is additionally very close to the actual surface.

In order to distinguish from a non-planar or an object with a concavity we perform then the analysis described in the following section. This analysis gives us a height map encoding how close the plane is approximating the actual point cloud. Given we classify the object as non-concave, we can use this distance in order to optimize the plane to fit better by adjusting the orientation by fitting a plane into the height map. A result for a final plane fitting can be found in Figure 4.7.

Concavity Detection

In order to detect a concavity, we have to find a point on the tool that is lower than the border around it. Therefore, we create an image that contains the averaged distances of a certain volume over the extracted plane. This image is a height map, which look like the images depicted in Figure 4.8.

The image that results plotting the distances to the plane should form a minimum in the center and an elevation on the borders to form a concavity. We consider a convexity as a

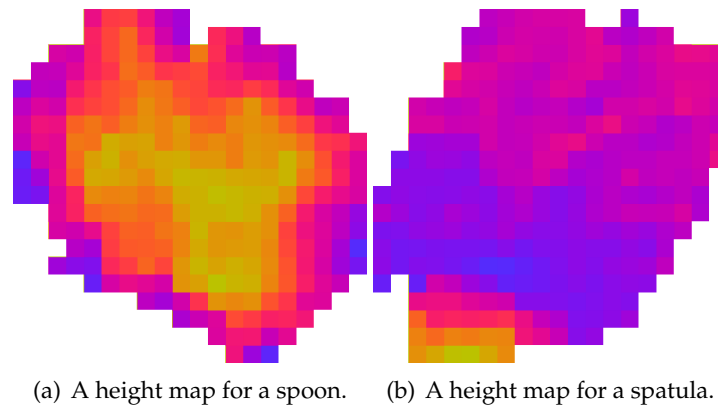


Figure 4.8.: Height maps of a tool with concavity in (a) and without in (b).

back-side view of a concavity, which just flips the sign of the comparison for concavities and will be treated as if we would have found a concavity on the other side.

To find the most probable point of a concavity, we search for the position and a radius of the two concentric circles that have the highest difference in their mean values under their border in the image we extracted. The circles are depicted in Figure 4.9(a), which also depict the three dimensions which are searched in order to maximize the difference in height between center and outer circle. This search can be done very fast because of the low resolution of our height map.

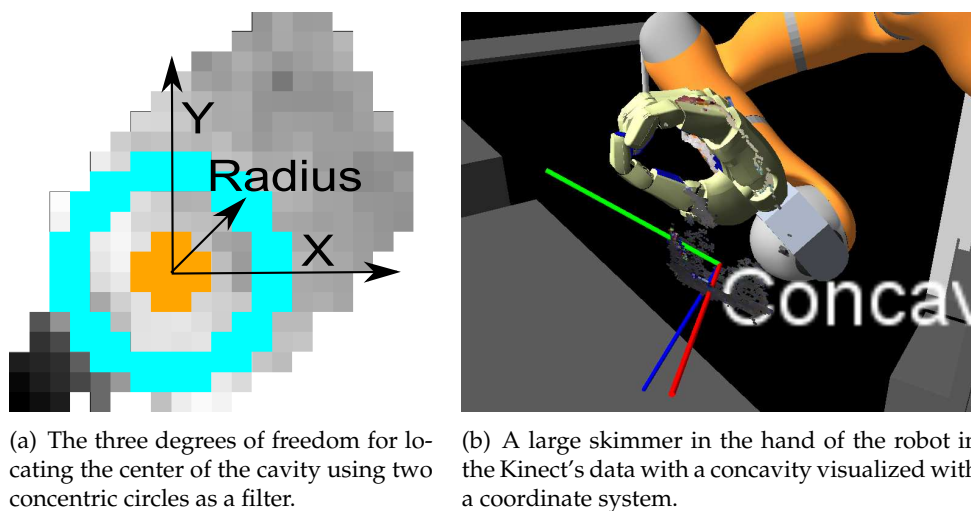


Figure 4.9.: Concavity Detection.

The position and radius with the highest difference is then analyzed if it is a real concavity: It must have a derivative in height from the outer to the inner circle in the same direction. This is done by comparing the maximal height in the center with the minimal height under the outer circle. This gives us a good measure if an object has a concavity, for a convexity the inner minimum has to be compared with the outer maximum. This comparison allows some

4. Detection and Reconstruction

outliers in the outer circle, given the thickness of the outer circle is higher than the number of outliers.

The position in space is then on the former plane surface at the new center of the concavity with the z-Axis pointing into the concavity. An example for this representation can be seen in Figure 4.9(b).

Edge Extraction and Hole Detection

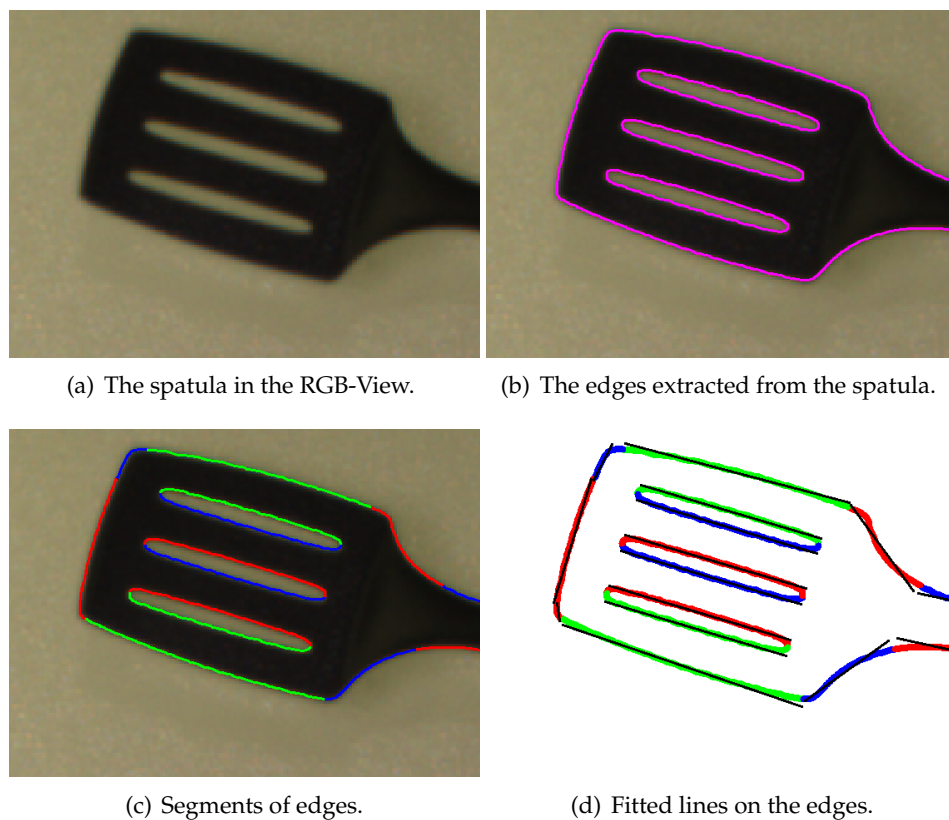


Figure 4.10.: The Line fitting.

In order to find sharp edges at the tool we search for straight edges of a certain length. The steps of the edge extraction are visualized in Figure 4.10. We use a standard sub-pixel canny-edge filter, which avoids additional forks at edge intersections, visible in Figure 4.10(b). These edges we split into connected segments. We merge two of those segments if their edge direction is similar enough and the closest points on the two segments are close enough. The contours we split again based on collinearity, meaning if the direction along such a contour varies too much, it is split again. The resulting segments are shown in Figure 4.10(c). On all remaining segments of a reasonable size we fit a line by a robust least square regression, depicted as black lines in Figure 4.10(d).

4.4. Transparent Object Detection and Reconstruction

All resulting lines can be intersected with the plane we extracted in the 3D data. In order to get a position in space, we can project the start and the end of a line onto this plane. The resulting 3D points define the line and its extreme points in 3D space.

By simple morphology and blob analysis we extract holes in the tool.

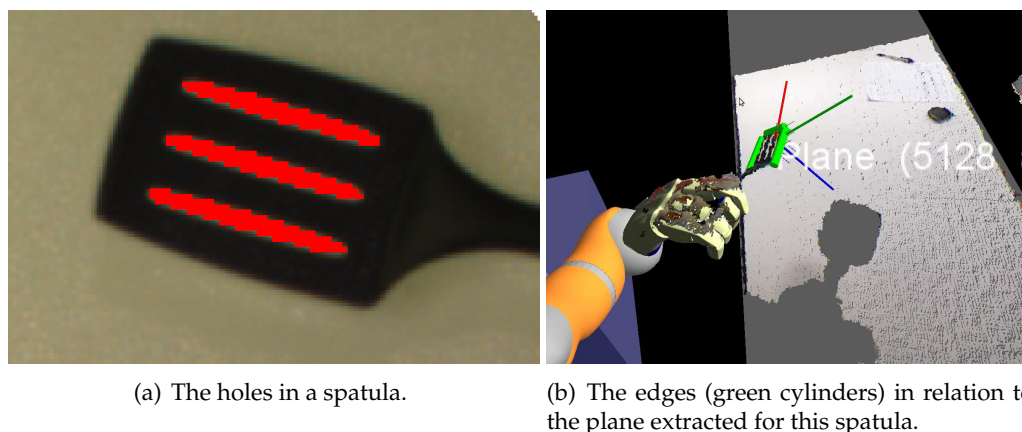


Figure 4.11.: Examples for results of hole detection and edge extraction.

The holes which we detect for the spatula can be seen in Figure 4.11(a). All lines around holes are not reported as results. The existence of holes is reported as a result as well as the position and length of the edges in space. Only Edges of a significant length are reported, which are in case of the exemplary spatula the three visible in Figure 4.11(b).

4.4. Transparent Object Detection and Reconstruction

The challenges that transparency implies to common sensors and algorithms seem simple [109]. Nevertheless, no general solution could be found [139]. Laser beams, e.g. emitted by LIDAR-Sensors, are usually partly reflected and refracted several times before they hit any surface, which leads to false or no 3D information at all [22, 146]. For normal camera systems transparent objects are almost invisible except for specularities that can be used to deduce shape information if well-defined preconditions are met [63, 64]. A 3D reconstruction using common stereo-vision approaches is difficult due to the lack of stable features on transparent objects [1, 11]. Tests with structured light [40] that were carried out at the beginning of our work showed only poor results on objects like drinking glasses or plastic bottles. On the other hand, some of the reconstructions were good enough to fit a 3D shape model into the point cloud. But the success of these approaches is still heavily depending on various factors such as lighting environment and object shape. Decent results could only be obtained under very constrained conditions. Therefore, recent studies concentrate on the development of algorithms which consider the special properties of translucent objects [50, 1]. Additionally,

4. *Detection and Reconstruction*

research into various sensors is being done, to maybe obtain more useful measurements [74, 92].

The approach presented here provides a foundation to expand available object recognition systems by transparency, leading to more robustness in the robot's environment perception. Apart from that, we took the household-robot as the use case for the actual system. Accordingly, our method is supposed to enable a robot not only to detect but also to manipulate transparent objects, which requires a reconstruction.

4.4.1. Related Work

A great deal of effort has been put into the modeling and detection of transparent objects in the past. Accordingly, there is a variety of publications that offer many different approaches. [50] offers a detailed overview on automated detection of transparency for various types of sensors. The authors conclude that many of the methods are promising but postulate certain constraints to environment, structure or object shape. In a household environment, however, we have to deal with random objects within arbitrary environments which is a reason for our novel approach. Nevertheless we want to present the research that contributed to this work. As an example, transparent objects were not considered in [109]. The reason for this is explained by Ezra and Nayar in [11]. Transparent obstacles lack stable features and therefore remain invisible for usual image processing algorithms. In this paper, Ezra and Nayar tried to perceive transparency by tracking features in digital images that were part of the environment and got reflected by the transparent object. The movement of the features on the surface of the object when the camera position was changed revealed information about the object's shape. Unfortunately, everyday objects like drinking glasses will not refract an image of the scene to the camera due to their complex shape and the less ideal material.

Wallace and Csakany proposed in [139] to newly interpret the photon detections in a time-of-flight system. They show that in a detection-number over distance histogram a peak occurred for every surface in the system's field of view. The peak for an opaque obstacle was just more intense than the peak for a transparent one because the emitted light was not entirely reflected but also transmitted and refracted. Depending on the object, multiple peaks would occur if the light had to pass several transparent surfaces, whereas an opaque surface resulted in only one distinct peak. Even multiple layers of transparent surfaces can be resolved with the proposed single-photon-counting but this method requires special sensors and access to the electronics in order to perform this kind of counting. However it shows that there is some kind of response to TOF sensors.

Another way to estimate the shape of a transparent object is presented in [22]. An object is heated with a Laser which is adapted to a wavelength that lies within an absorption maxi-

4.4. Transparent Object Detection and Reconstruction

mum of glass. After the heating the thermal radiation of the surface is being observed with a thermal-camera. Through the heating the transparent object becomes quasi-opaque to the thermal camera and the shape can be reconstructed. Although this method is not suitable for household applications, the publication offers important information about the absorption maxima of glass in infrared and ultra-violet wavelengths. As a matter of fact the SR4000 uses infrared light with a wavelength covered by the IR absorption maximum [40], so the response to glass that we discovered in the intensity images is reasonable.

The problems transparency poses for laser-range-sensors are depicted in [146] by Yang and Wang. They tried to overcome these drawbacks by adding ultra-sonic-sensors what made their robot capable of navigating in areas with windows and mirrors. Yet they showed that mere laser sensors are not sufficient.

The insufficiency of laser-sensor data when digitalizing transparent objects is also described in [63] by Steger and Kutulakos. This work proposes a method which reconstructs a shape by triangulating the light path between feature points in the environment and their mirror image in a transparent object which is seen by a camera. Here the object has to be shaped in a way that it reflects these features. Additionally the position of the features in 3D-space has to be known for which reason they used projected-light to create artificial features with known positions respective to the camera. However, the requirements for this method cannot be met by an arbitrary household scene.

The approach described in [64] uses specular reflection to optimize position estimates of objects. The method derives a light-map from specular reflections that move over shiny surfaces when the camera pose is altered. The displacement of the specular reflections is thereby significantly higher than the change in the texture seen by the camera. By iteratively comparing different views under consideration of the light-map the initial pose estimation is being refined. Despite its possible application to transparent object recognition this method is unsuitable because we cannot say for sure whether an object is shiny enough and whether there are sufficient light sources.

An adaption to the optical flow equations is specified in [1]. The equations are altered in a way that the shape of a translucent obstacle can be derived from the movement of the scene behind the object. In a household environment we sometimes have to deal with very homogeneous backgrounds such as monochrome tables or counter tops. Feature points are rare on these surfaces and therefore pose a big problem for this method.

[74] makes use of the polarization of refracted and reflected light rays for a shape recovery of translucent obstacles. Placed inside a spherical device the object is observed by a CCD camera through a polarization filter. When moving the filter the intensities of the light change in a certain way that enables this method to estimate surface normals. Due to the

4. Detection and Reconstruction

need for complicated devices this procedure is not viable for object recognition performed by a robot.

Hata et al. [40] reconstruct the shapes of translucent paste on a flat surface by projecting light patterns onto it and observing variations within the structure of the pattern. Unfortunately, the tests with structured light that were carried out at the beginning of our work showed only poor results on objects like drinking glasses or plastic bottles.

At large none of the available methods offers a solution to our use-case. Specularities and feature points are too dependent on the environment structure. Laser scanners and structured light do not grant valuable results. Nevertheless, the absorption of infrared and ultra-violet light lead us to the fact that the response of the SR4k ToF camera to transparency is viable and forms a basis for a novel approach.

4.4.2. Absorption based Inconsistency Analysis

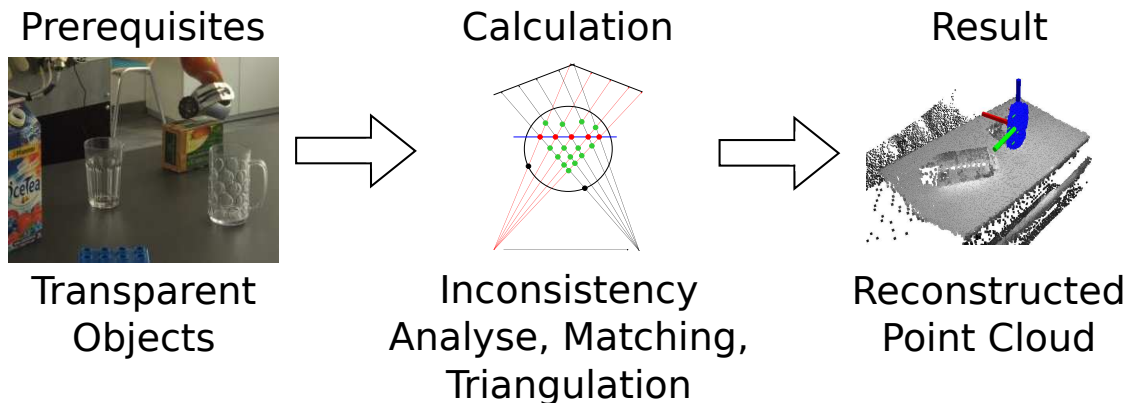


Figure 4.12.: Prerequisites, Calculation and result of Absorption based Inconsistency Analysis.

The method which is presented here is also described in [54]. The inputs we use for our method is the SR4k point cloud data which is also converted into a distance and an intensity map of the size 176x144. The system itself is divided into two steps with the robot's movement in between. In the first step the data from the SR4k is read, the obtained intensity images are optimized in their contrasts [34] and are segmented by extracting darker areas with an enhanced thresholding [82]. This procedure yields possible candidate areas for transparent objects, since darker areas can be caused by absorptions by transparent material. These candidates support not only 2D intensities but also 3D information corresponding to every image point. The defined candidate areas are then returned to the operating system together with the corresponding distance and intensity image as well as the 3D point cloud data.

4.4. Transparent Object Detection and Reconstruction

In the second step the robot performs a movement within a certain range which provides pose parameters that can be used for a 3D point transformation between the first and the second view on the scene. These parameters are acquired from the operating system ROS by comparing the two robot positions for each view which originate from an AMCL driven self-localization supported by two laser sensors. To ensure that the candidates remain in the field of view of the ToF camera, every candidate has an approximated world coordinate pose attached to it such that the platform can focus this pose and run the last step.

Now a second view is generated as well as a second segmentation with the same procedure as in the first step. The inputs are then complemented by the 3D transformation data and the ToF data from the first view. The method then processes every candidate and checks whether it has the characteristic of a transparent object when comparing the two views. In order to perform this check, we first establish 2D image correspondences by applying a perspective invariant matching in the intensity channels [46] for the respective candidate. In the next step the algorithm ascertains whether a candidate is a transparent object or not by checking for inconsistencies in its 2D and 3D points when comparing the two views. If the check is positive, a 3D reconstruction is carried out and the new 3D points are transformed into a suitable form for later grasping or path planning algorithms. More detailed information about each of the mentioned steps will be provided in the following sections.

Problem Discussion

To begin with, the available data will be described briefly. As shown in Figure 4.13(a) a transparent object can be seen in a usual camera image if the light environment fulfills certain conditions. Here we have no direct light from above and a distinct amount of ambient light coming from our ToF sensor. Figure 4.13(b) shows the same camera image where bright light is illuminating the scene from above. The translucent object is barely recognizable with human eyes. As a result, common stereovision approaches fail to even perceive the object.

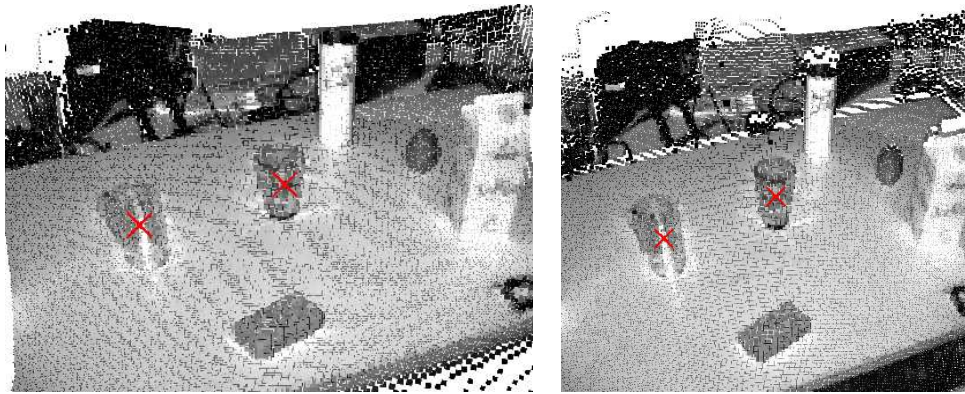


(a) Barely illuminated by ambient light from the side and Sr4k light. (b) Highly illuminated with bright light coming from above.

Figure 4.13.: Camera Images of the same scene with different illuminations.

4. Detection and Reconstruction

Accordingly a sensor to perceive those objects should be mostly invariant to the light environment. Here we propose the use of a SR4k ToF camera, which is mostly invariant to ambient light and therefore fulfills our demands. When a transparent object is positioned in the view of a ToF camera absorption of the IR-light is measurable in the intensity channel. Yet it provides improper 3D data for non-opaque objects. Figure 4.14(a) and 4.14(b) show two point clouds with the intensity values associated to each point that was generated by the SR4k while observing the same scenes with changing light environment described above. The scene shows a table with several opaque objects and two transparent objects in the middle and on the left.



(a) A scarcely illuminated scene.

(b) A highly illuminated scene.

Figure 4.14.: ToF point cloud colored with intensities showing different, yet not distinguishable, illuminations

These views seem to show a decent measurement of the transparent objects, which are marked with red crosses. As one can see, the glass's shape is perceivable as a darker area on the table. The reason for this is that the infrared light emitted by the SR4000 camera is partly absorbed by the glass [22]. The surrounding table surface on the other hand yields much higher intensities because the infrared light is reflected strongly. A very striking fact can be seen when observing the point clouds from a different angle. The 3D points of the glass are as flat as the table's surface on which the object was placed. This effect is illustrated in the Figures 4.15(a) and 4.15(b).

Moving the camera leads to a shadow-like behavior of the transparent objects, what means that the wrongly estimated 3D positions fall away from the light source that illuminates the scene onto the next surface which is the table in our setup. According to the results in [139] we suspect that one part of the infrared light is reflected because the intensity images contain specularities as well. After the light has propagated through the object it supposedly hits the first opaque surface, which is the table, and is reflected back to the camera. Since the light was mainly reflected by the table, the distance estimation of the ToF camera is in accordance to the surface behind the glass, explaining the flattened 3D estimation. The following set of

4.4. Transparent Object Detection and Reconstruction

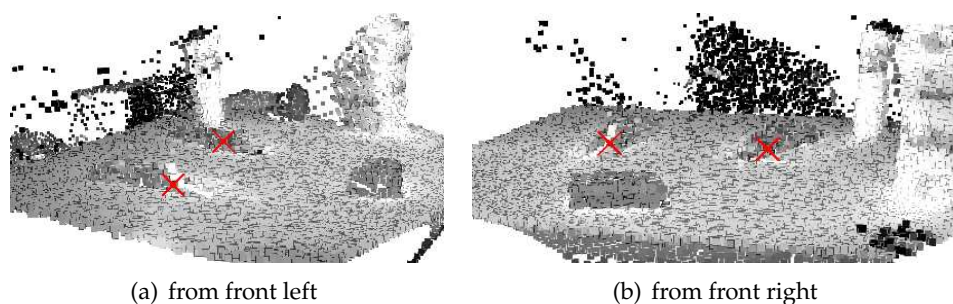


Figure 4.15.: The red crosses mark transparent objects that are as flat as the surface because the ToF camera cannot measure the shape correctly

images in Figures 4.16(a) and 4.16(b) depict the shadow-like behavior. The robot moves to the right and the point clouds of the transparent objects move into the opposed direction.

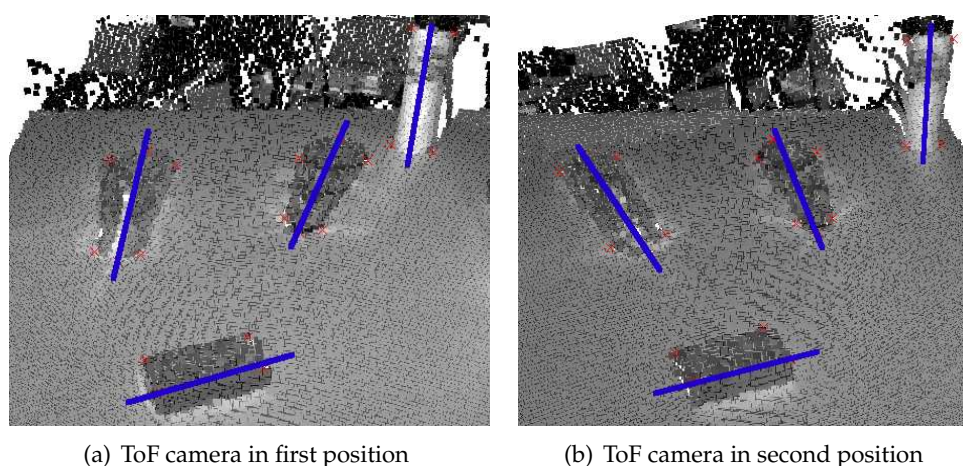


Figure 4.16.: Observed from a fixed point of view the transparent objects (marked with red cross) are moving like a shadow over the surface when the ToF camera is moved. The blue axes indicate the shadow-like movement.

All the objects in the Figures 4.16(a) and 4.16(b) are marked with red crosses to illustrate their alignment. The point of view is stable as well as the table on which the objects are placed. The only movement is performed by our platform such that the ToF camera position changes. The blue axes indicate that the opaque objects on the bottom and on the right keep their alignment whereas the two transparent objects in the middle and middle left of the picture perform a twist to the left. The aforementioned effects are taken into consideration by our method to distinguish between opaque and non-opaque objects. If a transparent object is found, the real 3D shape has to be recovered to enable our platform to manipulate it.

4. *Detection and Reconstruction*

4.4.3. **Segmentation**

In order to avoid calculating a complete image to image correspondence we apply segmentation as a preprocessing step. A complete image to image correspondence would lead to too many ambiguities in the matching since the features observed in the SR4k are usually weak. Fortunately, transparent objects absorb enough light of the ToF camera's emitted flash, that they appear darker than their background except from some specularities which are significantly brighter. Objects with a proper 3D response in the camera could be segmented using geometric cues [109] but this method is not functional for translucent obstacles. Accordingly our method performs segmentation in 2D in order to find areas with possible inaccuracies that have to be restored by a reconstruction procedure to support the reliability of platform functions such as grasp-position planning.

Intensity-based Initial Segmentation

In order to generate candidate regions that our method can focus on, the lowered intensities have to be found. First, we enhance the contrasts [34] between environment and transparent object which leads to emphasized boundaries between objects and background. The high contrasts allow the usage of an optimized threshold [82] which, applied to the intensity image of the ToF camera, extracts pixel-areas that represent lower intensities.

This extraction is carried out on a higher pyramid level as well, and after that, the results of both levels are combined what removes most of the false candidates. This leads to a better separation of the regions because thin structures in the environment do not appear anymore. The extracted regions are then refined with morphing operators to remove noise, separate areas and obtain solid regions.

Automatic Dark Background Adaption

In the course of the development we also considered a dark gray table as a surface. The segmentation procedure partly failed here because the table did not reflect as much of the IR-light anymore. The specularities actually seem to overcome the reflections of the table back through the objects which may originate from the heavily lowered reflectivity. However if the infrared light only passes one transparent layer the intensities keep their attribute of being darker than the table's surface. For two layers we have to deal with higher intensities now, as the specularities are brighter than the background. Nevertheless our segmentation adapts to darker backgrounds by assuming that the average image intensity complies with the background intensity. The average intensity is then interpreted as a dark or bright background.

Splitting and Merging

Occasionally, we found the regions of non-opaque objects still connected with opaque ones due to distinct ways of scene setups and view angles. As this leads to problems in the later matching phase, we added a second segmentation step that splits these regions. By generating a histogram that assigns column or row values to occurrences. After a smoothing procedure the function representing this histogram is derived for the dimensions of row and column to extract local maxima. Each maximum represents a new centroid for splitting candidates. All split candidates are tested against nearby candidates for a possible merge to avoid over-segmentation.

Candidate Omission

The resulting candidates are then evaluated based on their region convexity. If this factor is above a certain threshold, we calculate the deviations in x , y and z direction for the associated 3D points. We use the deviations as an approximation of the volume that is occupied by the candidate. From the number of points and this volume we may now calculate the point density. Applying this procedure we are able to omit candidates that do not yield a solid region or high density. Especially noise that occurs at the edges of the view has very low density.

The results of the initial segmentation are exemplarily displayed in Figure 4.17(a). The refinement of this segmentation using the splitting, merging and omitting steps is shown in Figure 4.17(b). Both images show regions overlaid over the intensity image in different colors for the each segment.

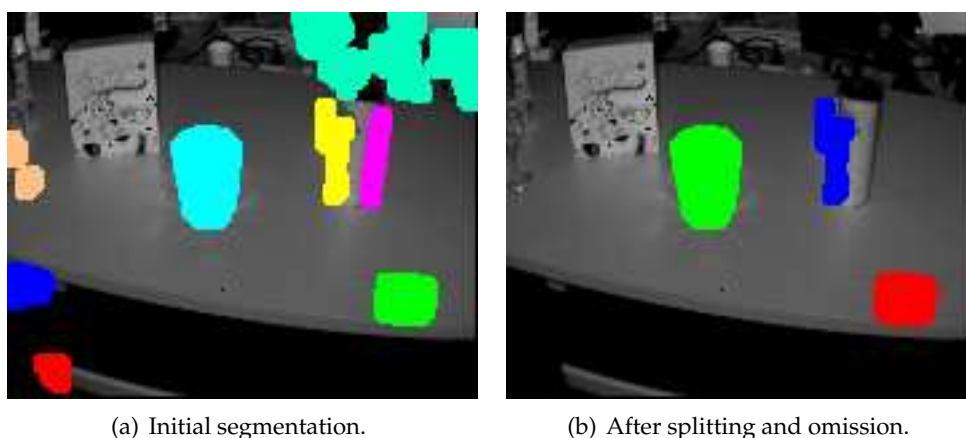


Figure 4.17.: Result of the whole segmentation phase with a solid transparent object in the middle.

4. Detection and Reconstruction

4.4.4. Matching

The matching procedure is crucial to find out where the candidate region moved to in the second view in order to be able to triangulate between the two views. We solve this correspondence question by introducing a planarity assumption for the images we observe. We assume the objects to be shadows, or more precisely projections of an object onto a flat surface. This assumption allows us to match the objects with a perspective invariant matching while introducing errors that are discussed later.

Initial Planar Matching

Thus, we use a perspective invariant matching, which is robust against noise and local deformations, presented by Hofhauser et al. in [46]. This matching builds a candidate based on a gray-value template and returns a planar homography for the best match from the first to the second view. The distance measure used for the matching score is based on corresponding image derivatives in direction. A prerequisite for the method is planarity of the template region. However, the matching is robust enough even if the background behind the transparent object is not planar. We use the candidate regions gathered by the segmentation in the first view as template images. The search is then applied to all parts of the second view that were found by the same segmentation step that are close enough to the predicted position. The prediction contains estimates of the maximal expected rotations and perspective distortions. On success, the matching results in a 2D homography which is applied to the candidate's image points. By transforming a region we get another region in the second view that covers the candidate's match, which leads directly to an initial point to point correspondence in 2D as well as 3D. In order to visualize this correspondence we introduce the following symbols: the segments in the first view will be represented by $S^1 = [s_0, s_1, \dots, s_m]$, the segments of the second view are $S^2 = [s'_0, s'_1, \dots, s'_m]$, each consisting of a set of measured points with a 2D coordinate p , a 3D coordinate q and an intensity value v : $s_i = [Q_{i,0}, Q_{i,1}, \dots, Q_{i,r}]$, $Q_{i,j} = [p_{i,j} = [x, y]^T, q_{i,j} = [X, Y, Z]^T, v_{i,j}]$. We estimate a planar homography H between the two views for each segment s_i and its resulting correspondence in the second view s'_k .

The best discrete pixel correspondence C defined by the indices j and l between the first and the second segment is then just the closest pixel to the transformed point using the homography:

$$C = \left[(0, l_0), \dots, (r, l_r) \parallel \arg \min_{l_j} \|p_{i,j} - Hp'_{k,l_j}\| \right] \quad (4.1)$$

$$p_{i,j} \in Q_{i,j} \in s_i, p'_{k,l_j} \in Q'_{k,l_j} \in s'_k$$

4.4. Transparent Object Detection and Reconstruction

Figures 4.18(a) and 4.18(b) visualize this relation.

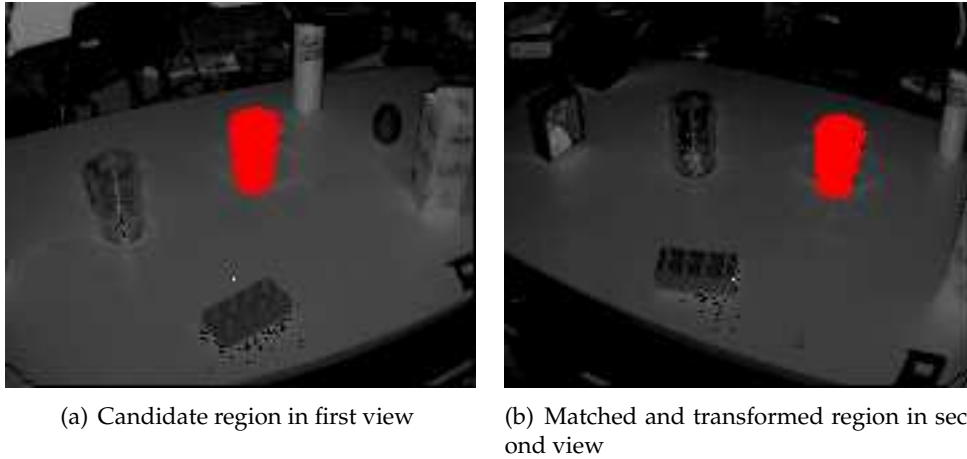


Figure 4.18.: The planar matching provides a perspective 2D homography that defines the pixel to pixel matching.

Introduced Errors by Planarity Assumption

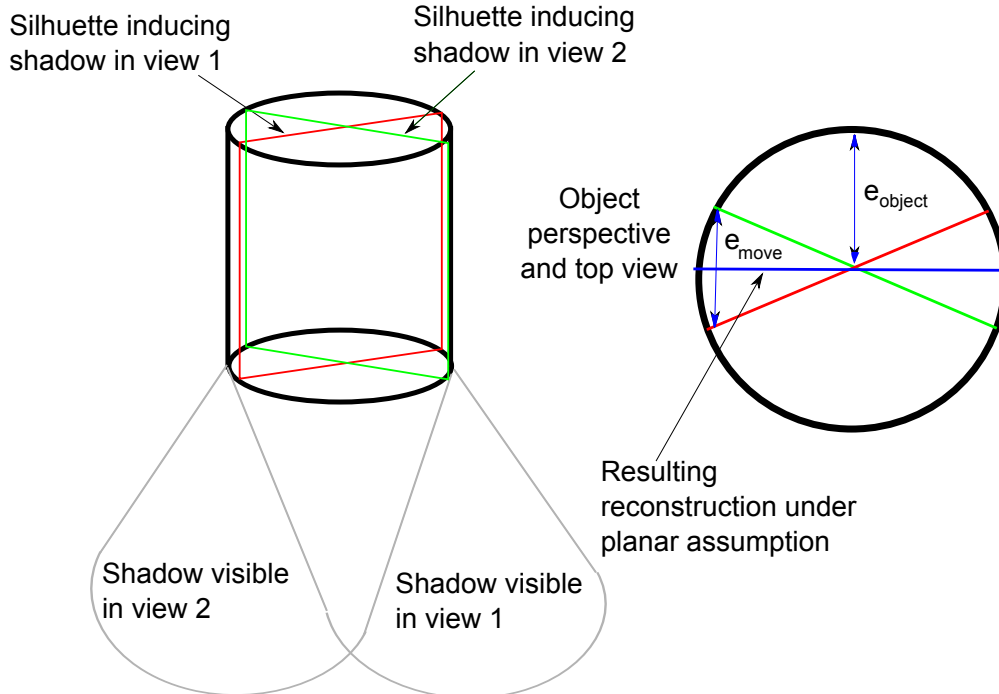


Figure 4.19.: Visualization of the errors introduced by the planarity assumption, left is a perspective view of the scene and on the right a top view of the object with two kinds of errors: e_{object} introduced by the planarity assumption, and e_{move} introduced by the changing silhouette due to the cameras movement.

4. Detection and Reconstruction

[46] also discusses prerequisites for this type of matching. Clearly, our method has to deal with a certain error as the transparent objects are not planar. Our results show that the triangulation of the center of the region is a valid approximation of the center of mass of the transparent object. Figure 4.19 shows the effect of the planarity assumptions: using the correspondences given by Eq. 4.1 a plane in space can be reconstructed (right part of the picture in blue). This plane approximates a mean plane of the two silhouettes observed in the two views as shadows on the table. Therefore, every correspondence bears an error e_{move} depending on the distance to the center of the reconstruction. Additionally, we underestimate the depth of the object with the planar assumption by e_{object} .

4.4.5. Inconsistency Check

The characteristics which we expect to distinguish transparent from opaque objects or dark patches on a flat surface are inconsistencies in their ToF 3D data to the 2D image data. To begin with, we depict the 2D related checks. Using the known 3D transformation P and the projection K , which consists of the known internal camera parameters, we predict the position of the set S^1 in the second view making it comparable to the set S^2 . Given a correct 3D measurement, the transformed 3D data should result in the same points $\hat{q}_{i,j} = [X, Y, Z]^T \in s_i \in S^2$ found by the matching procedure if the object is opaque. Then the transformed points $\hat{q}_{i,j} = Pq_{i,j}$ are projected into the image plane by applying K . The resulting 2D points \hat{p} should also be consistent with the 2D points p' of the candidate's match in the second view if the object is opaque. Here, we perform the first check that estimates the extent of offset within the 2D data which should be higher for transparent objects.

$$\sum_{\forall(j,l) \in C} \|\hat{p}_j - p'_l\| \tag{4.2}$$

This offset in 2D can be derived from the twist that a transparent object undergoes which was illustrated in 4.16(a) and 4.16(b). This twist not only causes an offset but also a rotation between the reprojected and the matched candidate region what leads to the second check that determines whether this rotation is above a defined magnitude. From the correspondence between p' and \hat{p} we deduce a similarity transformation in R^2 with 5 degrees of freedom (dof): the rotation R (2 dof), the translation t (2 dof) and the scaling s .

$$\arg \min_{R,t,s} \sum_{\forall(j,l) \in C} p'_l{}^T \left(\begin{matrix} sR & t \end{matrix} \right) \hat{p}_j \tag{4.3}$$

This similarity should result in an identity for correct 3D data. Otherwise the sheering introduced by the shadow effect can be measured in the R component. We can define a threshold

4.4. Transparent Object Detection and Reconstruction

depending on the movement performed by the camera between the two views which has to be exceeded by a transparent object. If the candidate data passes the threshold, the 3D data is tested for inconsistencies to further assure transparency. Given again the 2D point to point correspondence C , we can triangulate the actual 3D position of a point r in space. For the 3D rays given by the two image points q, q' and the camera movement P , the best triangulation can be expressed by the following minimization:

$$\arg \min_{m_1, m_2} (m_1 q_{i,j})^T P (m_2 q'_{k,l}) \quad (4.4)$$

where $m_1, m_2 > 0$ are scalar factors. From this optimization a new 3D point can be calculated using the gained factors m_1 and m_2 :

$$q'_{triangulated} = \frac{(m_1 q_{i,j}) + P(m_2 q'_{k,l})}{2} \quad (4.5)$$

The greater the difference of the two factors m_1 and m_2 from 1, the more likely there is an inconsistency between the triangulation and the 3D measurement. If we triangulate all points in the segments we approximately get a planar structure in 3D which is in correspondence with the measured 3D points, as already shown in Figure 4.19.

The planar body has certain attributes that are correlated to the scalar factors m_1 and m_2 which our first 3D related checks are based on. First the position of the triangulated points is analyzed which should be between the camera and the measured 3D points.

$$q_{i,j} - q'_{triangulated,i,j} > 0 \quad (4.6)$$

The second property is that the planar reconstruction should be nearly orthogonal to the flat shadow of a transparent object. We reassess this fact by fitting a plane into both the measured and the reconstructed points and calculate the angle α between.

$$\alpha = \arccos \left(\frac{n^T \cdot n'}{|n| |n'|} \right) \quad (4.7)$$

with the normals n and n' :

$$n = \arg \min_{a,b,c,d} (q_{i,j})^T \begin{pmatrix} a & b & c & d \end{pmatrix}^T \quad (4.8)$$

$$n' = \arg \min_{a',b',c',d'} (q'_{triangulated,i,j})^T \begin{pmatrix} a' & b' & c' & d' \end{pmatrix}^T \quad (4.9)$$

4. Detection and Reconstruction

After the shadow-like behavior was checked by means of its 2D inconsistency and its planarity, a last procedure checks whether the mentioned 3D movement of the shadow actually took place. As the movement should be determined by a rotation of the planar candidate point cloud, we identify its screw parameters by estimating the 3D transformation parameters that align the points q and Pq' . In order to obtain this information our method makes use of a single step of the ICP-algorithm [2] applying a least-squares-minimization to the following:

$$\arg \min_{R_{icp} T_{icp}} \sum_{\forall (j,l) \in C} q_j^T R_{icp} T_{icp} P q'_l \quad (4.10)$$

As a matter of fact the angles of rotation around the three axes are much bigger for the translucent obstacles than for opaque ones. From the rotation matrix R_{icp} we can deduce the 3D twist that the respective candidate has undergone. The overall magnitude of rotation between the point clouds q and Pq' indicates a transparent object with a high value and an opaque object with a very low value even if the measurement is erroneous. All together our method compares the two views of each candidate by means of image point distance and twist between the object's point clouds to determine whether an observed low intensity area is caused by an opaque or a translucent object.

Reconstruction

To finally be able to manipulate a transparent object, its real 3D occurrence must be recovered. As explained above, the initially reconstructed 3D points are located in a plane. Although the reconstruction mentioned in Section 4.4.5 is closer to an object standing upright it does not describe the real volumetric shape well enough to form a point cloud.

Based on the errors introduced in Figure 4.19, we can compensate for the error e_{object} and e_{move} by appending a set of additional possible intersections inside the object to the triangulation. The correct triangulation should occur in a set of reconstructions between $q_{k,k}$ and $P \cdot q'_{k,i}$ with k being a fixed number and i running from the first to the last column of the image row $p'_{k,i}$ that corresponds to $q_{k,i}$ and $q'_{k,i}$.

$$q'_{triangulated,i} = \frac{(m_1 q_{k,k}) + P(m_2 q'_{k,i})}{2} \quad (4.11)$$

Figure 4.20 illustrates such a triangulation performed on one horizontal slice of the objects shadow. The circle in figure 4.20 indicates a transparent object. The points r and r' satisfy the planar matching error explained in section 4.4.4 which also leads to the planar reconstruction displayed by the blue line and the red dots. In order to compensate this error we triangulate

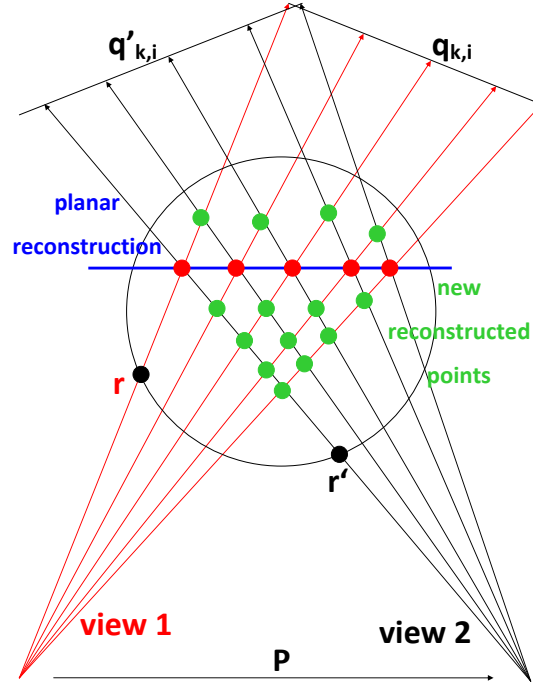


Figure 4.20.: The possible intersection points when triangulating between one image point of view 2 and all image points in view 1 that are located on a single image row.

new points shown as green dots using the matched points $q_{k,k}$ and $q'_{k,k}$ as well as the points in $q'_{k,i}$ that yield a point $q_{triangulated,k,i}$ that is closer to the second view position than the planar reconstruction. Yet the method does not intersect $q_{k,k}$ with all points $q'_{k,i}$ of the respective image row. The last point used fulfills the following:

$$|i| = 0.5 \cdot |p_{k,i}| \quad (4.12)$$

This approach yields a 3D blob shaped like the 2D region $p'_{i,j}$ that embodies an average point $\bar{q}_{triangulated,i,j}$ and the deviations around it which closely approximate the real position of the transparent object. The quality of this approach will be evaluated in the following section.

4.5. Summary

We proposed solutions to the problem of segmentation which can be applied to basic scenarios in a domestic environment. We can create object hypotheses and avoid obstacles without complete world knowledge. This enables a robot to act even before completely exploring

4. *Detection and Reconstruction*

its environment, which is a good base to continue with the recognition and categorization of objects. We also have described the problems in segmentation, and pointed out the challenges that occur in top down segmentation and reconstruction. The color based approaches are combinable with the 3D cluster based approaches. On the other hand, the method for transparent objects cannot directly combined with the other methods, since it already relies on certain properties of the intensity and 3D answers of specific objects, which partially contradict the assumption made in the other methods.

5. Object Recognition and Categorization

Assumed that the highlevel control has some valid object candidates for a semantic concept and wants to exactly localize a certain object in order to grasp it, the visual task can be seen as object recognition. All methods discussed here the recognition is either done by a localization of a model in 3D Space or by a classification of a segment in space. The approaches using classifications can be seen as a categorization, while the localization is more a recognition with an implicit categorization. We will start with approaches for general rigid and untextured objects. Then we continue to explore then the possibilities of texture on objects is giving us with feature-based approaches.

5.1. Rigid Object localization

If we have a model for an object it usually can be identified well. We present here methods for rigid objects that work under a planar assumption or assuming a CAD model being available. The property of an object to be rigid fits to a lot of objects in a kitchen environment: nearly all kinds of cutlery and dishes and tools are rigid, while they usually have little texture. So they are perfect candidates for the methods presented here.

5.1.1. Related Work

We are aiming on localizing a specific but previously unseen object in order to interact with it, which is a challenging task. A possibility to learn relevant information for grasping is to automatically estimate grasping points like it was proposed by Saxena et al. in [116] or to simply approximate an object by a primitive before grasping it [71]. Those approaches are limited by automatic segmentation which is still a problem, especially in range images, see [114]. Another approach is active vision that allows learning of shape or appearance of any object by automatic usage of the action capabilities of the robot for further inspection [60, 133, 30]. This approach involves necessarily changes in the environment, which is not always desirable.

5. Object Recognition and Categorization

A database of 3D objects enables a large set of new possibilities like highlevel motion planning [86, 76]. One possibility is to learn a specialized database; another is to select knowledge from a large database that was specified beforehand by humans, mostly available through Internet. Automatically using an Internet databases was already proposed in [28], but only using 2D-information, namely images. We are extracting 3D models from Internet-based databases. The selection of relevant objects is performed using clustering techniques based on the shape distribution function proposed by [87] and the distance measurement tested in [138] to find out the major cluster of the resulting models. Using a simple k-means algorithm, we are able to select inliers.

For calculating the most probable alignment in the world of such objects and if necessary to generate with morphing new models we align the models using a new Iterative Closest Point(ICP)-based algorithm which we called Volumetric ICP. The original version was introduced by [12]. We improved this ICP variant for our purposes. In literature, there are several morphing techniques most of them only applicable to closed meshes and are computationally very time consuming. Several of them are discussed in [65].

The last step for our problem is to match many 3D objects in 2D images. This problem was solved before and there are many different state of the art methods like for example [47] which is based on many views of a 3D object. We are using a state of the art 3D shape model matching technique, which is described in [143]. This method matches 3D-CAD-models in an image by simulating the 2D appearance in a model building phase. A range of position relative to the camera must be specified, which can only be done if information about the scene is available. In our kitchen scenario this is given, since we assume all static parts to be approximately known by earlier 3D-Laser scans or similar techniques and are semantically labeled via methods like [112].

5.1.2. CAD matching

One of the methods for localization in the system is the CAD matching, which allows to search through a complete position space for the best corresponding projection in an image. Several details of this method were published in already in [57] and [56], and the description here summarizes the methods described in those papers and describes newer extensions made after the publications. Given the CAD model describes the object exactly this matching results in an accurate pose estimation. The method we are using [143, 134] builds offline a search tree in scale space and finds online the best match of the projected CAD model and the current view's edges.

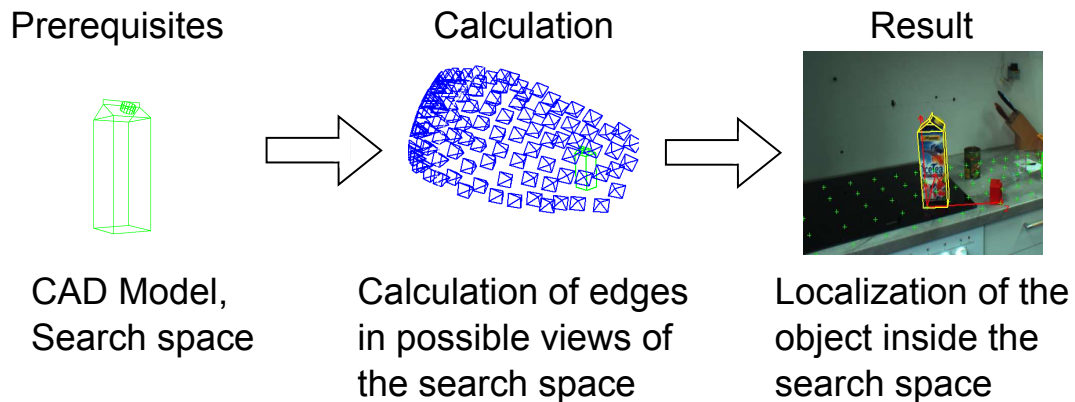


Figure 5.1.: Prerequisites, Calculation and Result of CAD matching.

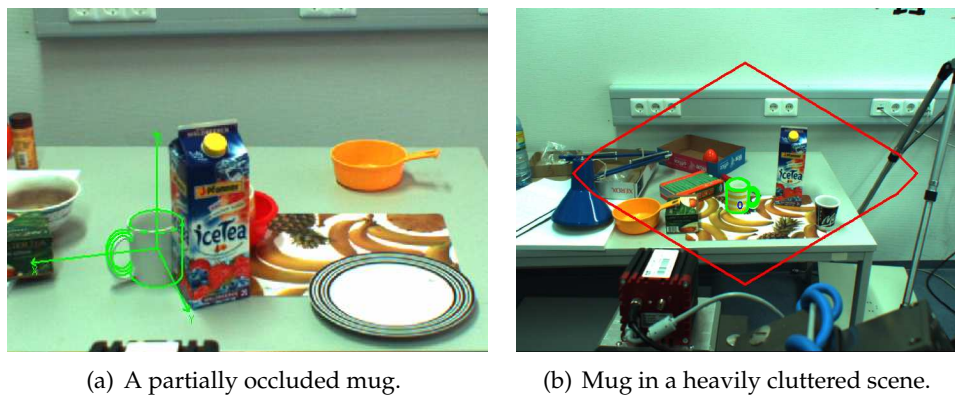


Figure 5.2.: Matching of a mug.

Search Space Calculation

Before propagating the point cloud data to the visual system we want to represent the search space in a compact form. Therefore, we describe the clusters C using their center points and their maximal extensions and possible rotations along the xyz coordinate axes. The extents are then converted to a 6×6 diagonal matrix, representing the estimated uncertainty of the cluster poses.

The matrix representation can be efficiently propagated through the different coordinate systems, to the final extents in the camera based spherical and image coordinates which we use to set up the search space for the CAD matching step. In our experiments we did not encounter any drawbacks from this compression scheme, since we have to overestimate the visual search space anyways because of a relatively large baseline between the RGB and the TOF cameras.

5. Object Recognition and Categorization



Figure 5.3.: Four different objects matched in a cluttered table setting. The projection of the matched CAD models is displayed in green.

3D Points to Search Space

Given that we have a point with its orientation in a 3D space, we interpret this point as an object to camera relation. To include such a point in the RGB camera search space we have to add its projection into a Region of Interest (RoI) image and transform it into the spherical coordinates. The latter enables modeling of the RGB camera image on a sphere with a radius r at a pose described by 3 angles: the longitude α , latitude β and camera roll γ . Given an already defined search space with $\langle \alpha_{min}, \alpha_{max}, \beta_{min}, \beta_{max}, \gamma_{min}, \gamma_{max}, r_{min}$ and $r_{max} \rangle$, we have to adapt it as soon as the point falls outside the RoI. For each point found in the adjacency of an already existing RoI, we extend the respective region to include the point by modifying its outer dimensions. Figure 5.4 shows an example of the 3D clusters estimated using e.g. the method presented in Section 4.2.1, here back-projected into the RGB camera image.

Finding the CAD Model

The CAD models of the objects that we want to localize are usually acquired as an a-priori step using precise laser sensors followed by a number of geometric processing steps (some-

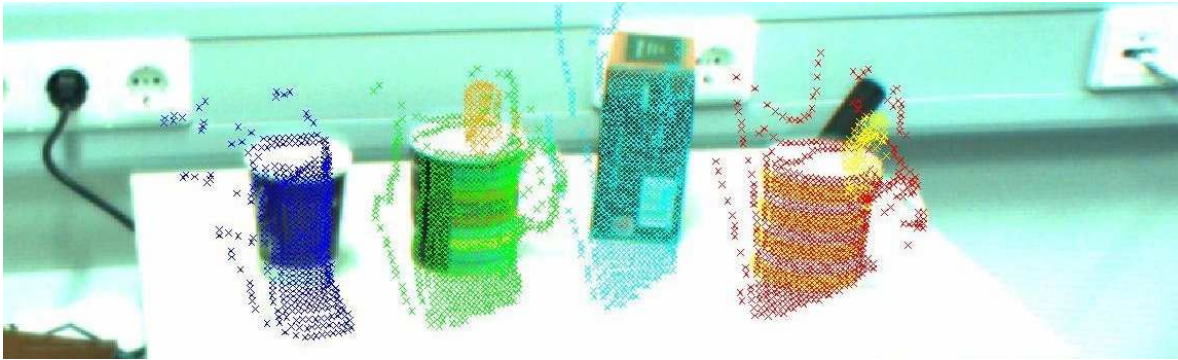


Figure 5.4.: Back-projection examples of the 3D clusters estimated using the methods in Section 4.2.1 into the RGB camera image.

times involving manual labor to clean the models), or can be alternatively downloaded from the Internet, see Section 6.2 and [57]. Most household objects available in the Internet databases can be found lying on supporting xy -planes in the 3D model coordinates. This is of course an assumption we do not rely completely upon, but we assume that the majority of the models are aligned in such a way. Since we additionally also align all selected inlier models to each other, we obtain the major “upright” direction. Following this we can assume that given a supporting plane we only have to account for one of the three rotational degrees of freedom. This constraint can be avoided by the highlevel system by specifying a larger search space in rotational space in order to detect also non-upright objects

This shape matching approach requires significant preprocessing of the given 3D model, whose complexity is polynomially increasing with the number of faces in the model and linearly with the number of views that have to be generated to get a complete search. The number of views in our application depends mainly on the given search space in the spherical coordinates system. Thus the constrained regions of interest reduce the search phase significantly.

To obtain fast results, we build a tree containing the projection of the model in different resolutions (image pyramids), with all leafs similar to their parents but at a higher resolution. The similarity of a match is measured in the 2D appearance of the projection. The information resulted from the segmentation step in the point cloud data gives a significant reduction in r (the distance to the camera) while α , β are only slightly affected by our search space restrictions. This is due to the search for the *same* classes of possible orientation on the table. If it was possible to reduce the RoI and the search range of γ , this would only affect the calculation time in the search phase. A very rough approximation of the estimated model calculation time is given as follows:

5. Object Recognition and Categorization

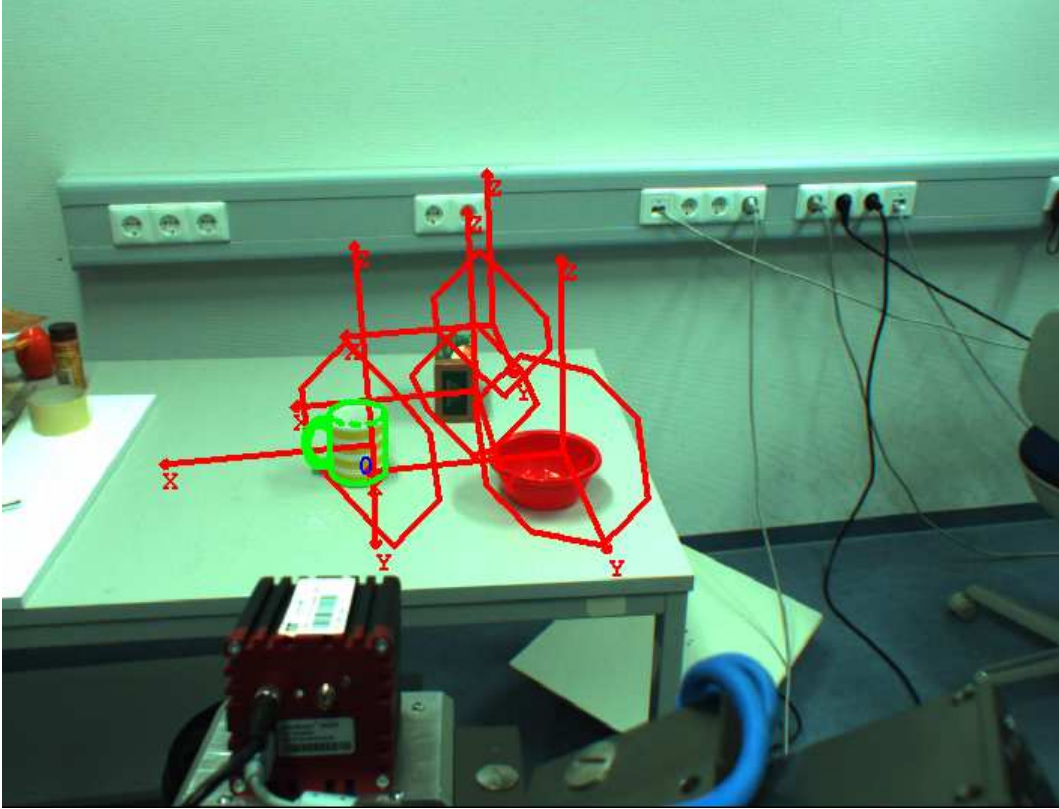


Figure 5.5.: A matched object on a table. The red polygons show the transitive hulls of the 3D clusters, while the projection of the CAD model is displayed in green. This search space is reduced to all clusters on the table.

$$step \sim \frac{\max(x_r^i, y_r^i)}{r_{min}} \quad (5.1)$$

$$t_{model} \sim N_f^v step(\alpha_{max} - \alpha_{min})(\beta_{max} - \beta_{min})(r_{max} - r_{min}) \quad (5.2)$$

where x_r^i and y_r^i represent the resolution of the image and N_f^v is the number of visible faces. Similarly, we can approximate the expected calculation time for the search phase by:

$$t_{search} \sim step(\gamma_{max} - \gamma_{min})N_{cand}^1 \quad (5.3)$$

where N_{cand}^1 is the number of candidate matches on the highest pyramid level, which corresponds directly to the size of the regions of interest used.

Figure 5.5 show an example the search space projected into the camera and a best match of CAD model of mug.

5.1. Rigid Object localization

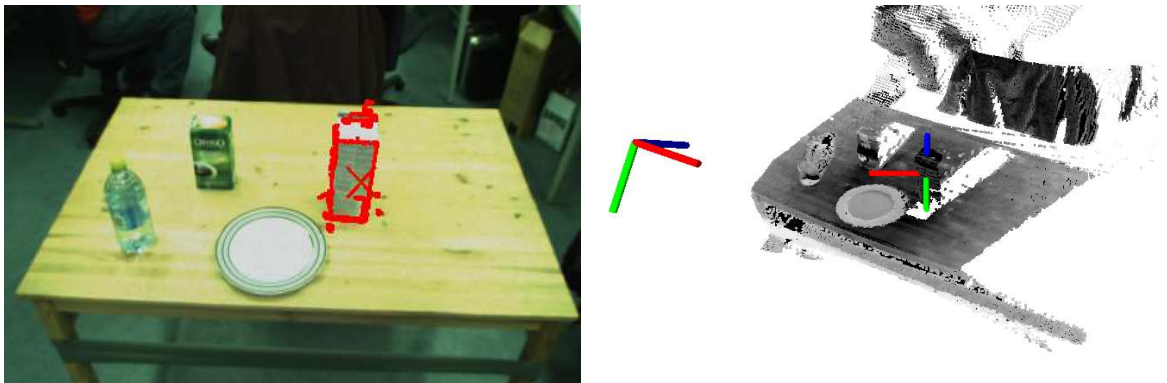


Figure 5.6.: An object matched using its shape model- The left part shows the matched shape template on the image and the right shows the resulting pose estimate in a 3D view of the same scene.

5.1.3. Planar Shape Model Matching

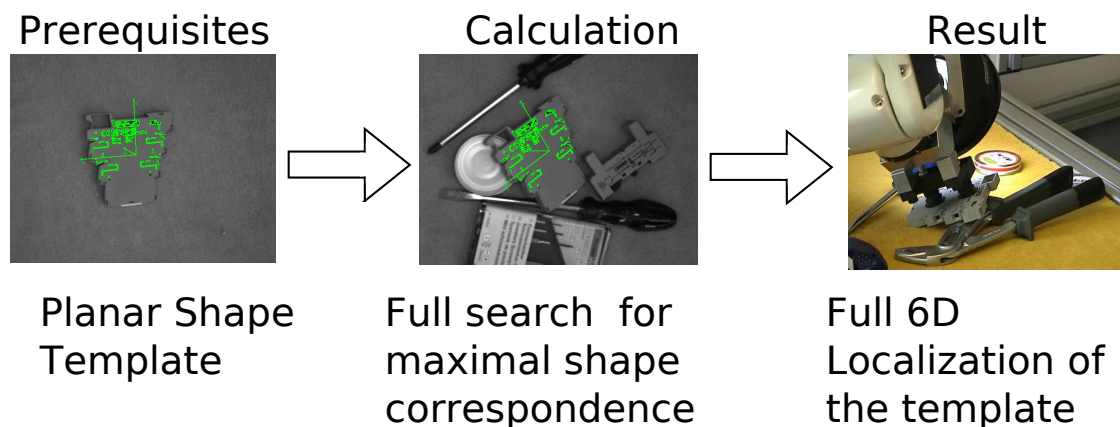


Figure 5.7.: Prerequisites, Calculation and Results of Shape matching.

Similar to the CAD based method, a shape model inferred from an earlier view of an object can serve to localize such an object. The distance measure used is similar to the method used for the CAD matching. Given a template image, we can calculate a homography estimating the best 2D-2D hypothesis between to a new image, [45].

This method is especially interesting due to the fact that it can return a 6D pose given the model is calibrated and the necessary models can be easily generated automatically, see Section 6.1.3. Those two facts allow this method to be easily applied to various scenarios like the detection of a pancake maker or the online calibration of tools.

Unfortunately, such a localization can result in the general case in a higher residual error, since in order to be able to describe a 6D displacement by a 2D-2D homography the template has to be planar. If this assumption holds we still get a lower accuracy in viewing direction

5. Object Recognition and Categorization

of the relevant sensor (similar to stereo measurements) and low accuracy in two of three rotational components. We performed a detailed error analysis for this method in the Results Chapter in Section 8.1.1.

Figure 5.6 shows such a shape model that is matched in an image. The better the assumption of the plane that was used for training was, the better we can derive the 3D orientation of the object from the result.

5.1.4. Surface Based Matching

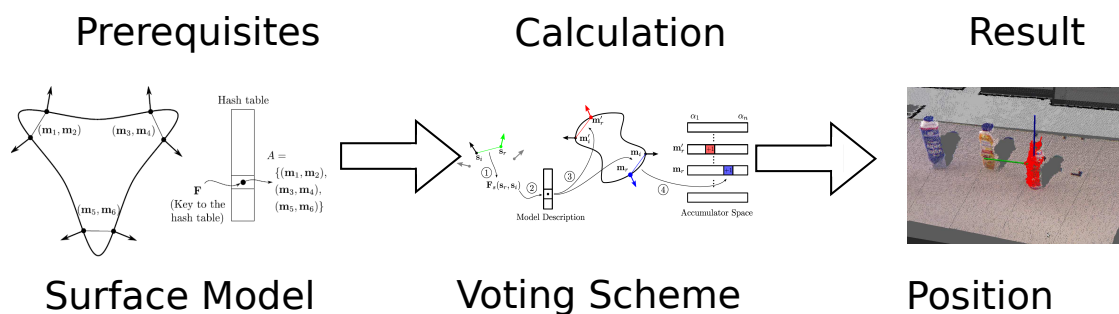


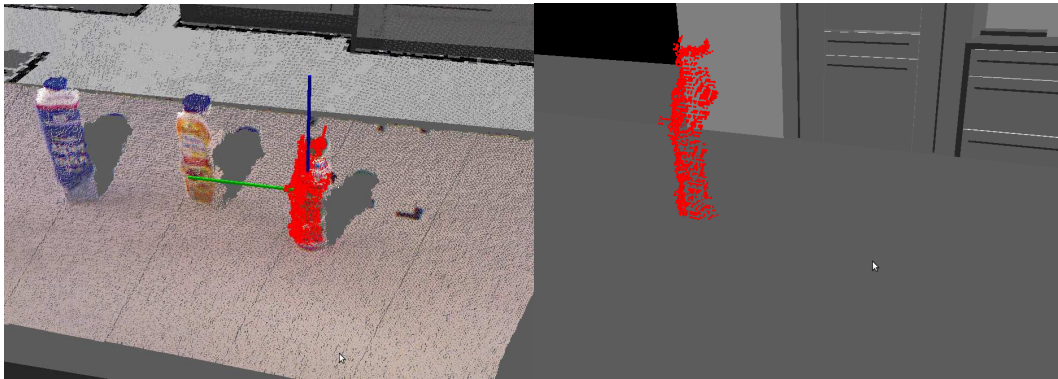
Figure 5.8.: Prerequisites, Calculation and Results of Shape matching, parts of the image are taken from [20].

Assuming the surface of the object is distinctive, we can extract from a segment of a depth image or a CAD model a surface model describing this object using the relations of normals on the surface. The relation of two identified points and their normals is already sufficient to estimate a pose of the object. By performing a voting in a new depth image about positions in space, the method presented in [20] can relocate the template again in further images, see Figure 5.9.

The pose winning the voting is refined with an ICP between the model points and the new depth image. The voting is performed in a discretized 4D space representing the possible locations the object can be located at, given a selected model point is at a certain position with a known normal. Such a point is called seed point and all other points of a sub-sampled version of the current scene are voting for a position with orientation in this space. After several seed points are tested, there should be a strong maximum for several of the seed points that corresponds to the best match in the image. This method will probably find a present object, given the sampling in the scene is dense enough and it has a unique shape regarding the background and clutter.

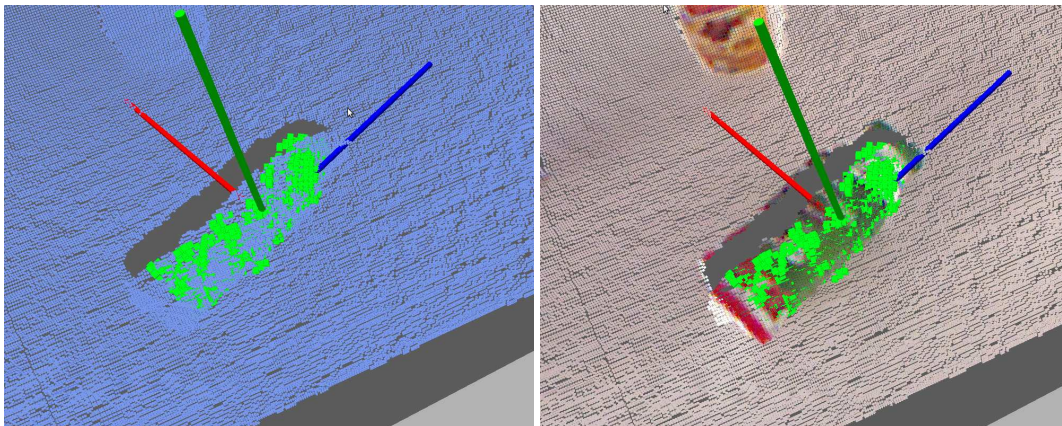
The execution time of this method scales with the 3D space that has to be searched, and comes to near real-time performance in our setup, if we reduce the search space in the depth image to the current working volume without known furniture.

5.2. Rigid Textured Object Classification



(a) A point cloud segment used for building a surface model.

(b) A point cloud segment used for building a surface model.



(c) A match and the transformed segment at a new position.

(d) The same match with a colored point cloud.

Figure 5.9.: Surface model for localization of Objects.

5.2. Rigid Textured Object Classification

The methods from the previous section allow of course the application to textured objects as well, but they suffer from drawbacks that are not necessary if it is known that an object is textured. The assumption if an object is textured can be easily tested, given a view of the object. But also the pure assumption is valid that most consumable products that can be acquired in packages in the supermarket are textured. This means that the class of textured object is important for many objects in a human environment besides tools and furniture.

Generally a good idea to distinguish textured objects is to count occurrences of typical such key points. This is the most common method to deal with identifying textured objects.

5. Object Recognition and Categorization

5.2.1. Related Work

The classification of objects is a field that is strongly influenced by the Pascal challenge, which is renewed yearly, see e.g. [23, 24, 25]. These challenges provide a database with a large set of annotated training and testing images, which include one or several objects. Those objects are annotated for each the image by a class name and the ROI of the object. The usual task is to localize the objects in the image and return the correct class. This challenge gives a good idea, what are the interests and the possibilities of the current techniques. For example we can name the work of Philbin et al. [95] which nearly solves one of the challenges ([23]). Examples of objects included into the newest data set ([25]) are music instruments, bikes or tools.

The standard approach to solve this task is to compute statistics over features in the ROI for the training objects and compare histograms over occurrences of those features. The best results are gained with techniques based on rotation invariant statistics based on the image gradients, like SURF [8] or traditional SIFT [68]. New developments presented in these areas are e.g. the so-called Nister-Tree presented in [83], which allow the usage of many different classes in the training data, while still being able to distinguish fast and relatively robust between the classes in the test phase.

An extension to this idea of the image classification is the information retrieval from Internet sources instead of annotated training data. Examples for such information retrieval can be found in [122] which uses patches from a frame of a video to search for more occurrences of the marked object in this and other videos. In a later work [119] proposes to use Internet databases with images to acquire training data for object classification. Performing this unsupervised allows the usage of a huge amount of data while forcing the approach to be able to handle wrong or wrongly annotated images in the databases.

5.2.2. Bag-of-Visual-Words-based Classification

In cooperation with J. Peters and K. Gleissenthal, [93, 136], a method was developed to distinguish object classes based on training images from an Internet image search, some parts of this work were presented in [128]. With this training data, SIFT feature for interest points were extracted, clustered them into so-called visual words and used a set of different classification techniques to compare the histogram of an image taken with the robot to the histograms generated from the Internet images. The method to train the classifier is separated into three steps which are discussed in the following.

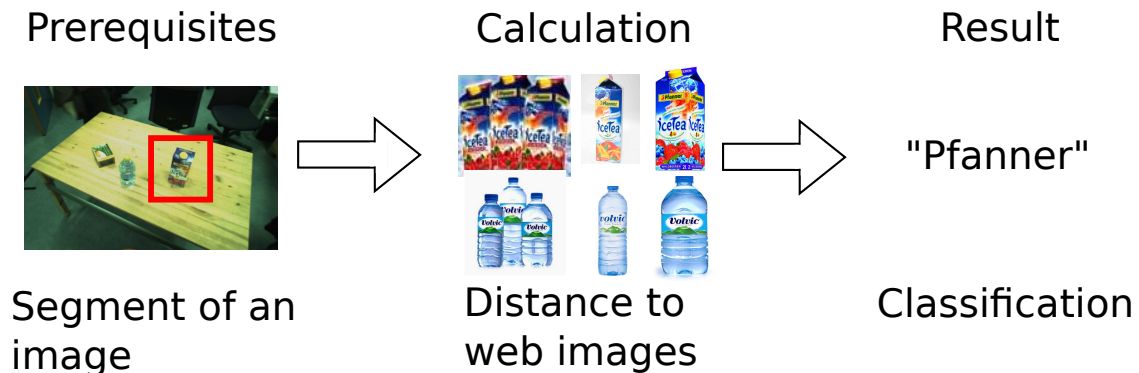


Figure 5.10.: Prerequisites, Calculation and Results of Bag-of-Visual-Words-based Classification.

Feature Extraction

The SIFT-vector is calculated from a histogram of gradient in the local neighborhood of point with high gradient changes (corners). Those histograms can be used to identify point in image, robust against different kinds of distortions. Different quantifications can be applied already on histogram level, so that a feature has a certain size, or it is extracted in different scale spaces.

This requires that this quantification in direction and scale are chosen carefully. Using images from an Internet search, the scale of the objects in the picture or the blur and noise level can only be guessed but not exactly estimated.

To overcome this restriction, we performed a search for optimal parameters regarding the possibility to distinct classes of objects by those features. The parameters which could be adapted for the feature extraction are basically the threshold how strong a corner has to be, the length of the SIFT-vector and the height of the scale space.

In tests with three classes as training data, we could achieve good results with a fixed parameter set. It seemed that this choice of parameters was also acceptable for new Internet searches for semantically similar objects. In the tests we used classes like 'Coke Can' or 'Volvic' or other brand names.

Quantification

A set of training images, which shows only a certain scenario, will most probably not cover the full space of SIFT vectors. In order to be able to have a compact and still dense representation, we try to extract local clusters of SIFT-vectors, which appear in the training set.

5. Object Recognition and Categorization

To perform this clustering fast enough on many classes, we took a k-means clustering applied on the set of features extracted from the images found for only one class. The resulting clusters are merged with the other classes using an approximated nearest neighbor search over all clusters for all classes. Two clusters were merged by using a distance threshold that is based on the in-cluster distribution. The resulting merged clusters define the so-called codebook.

Classification

The fused codebook is then used to create a histogram, how often an extracted SIFT features fall into a category of the codebook. A new SIFT-vector is sorted into the codebook by searching for the closest centroid in the codebook. As distance measurement we opted for the Euclidean distance, which is a valid but probably a non-optimal choice. This comparison with the centroids can be done in a tree-like structure and can be considered as relatively fast. The histograms are used as the feature to train a Support Vector Machine.

We used a *v*-SVC [118] by estimating the best parameters using again a subset of the training. The best set of parameters is chosen based on the best result on the rest of the training data and verified using a 10-fold cross validation.

Online Application

The resulting SVM with the selected parameters then can be applied online to segments of images:

We extract the SIFT features of this image and find the closest cluster in the codebook for each of them and build up a histogram that we can feed to the trained classifier. The resulting class and evaluation can then be used to identify of a certain object class.

Tests have shown that it is possible to distinguish a large number of classes given a good enough quality of the test images. It turned out to be difficult to create valid test data on TUM-Rosie, which is using lenses on its cameras with a too wide field of view to achieve the optimal resolution on an object. Some of the performed tests, which lead to the conclusions above, can be found in [93].

5.2.3. Randomized-Fern-based Classification

A descriptor of an interest point can be a statistical model instead of the normalized feature vector as for example SIFT or SURF. An example for such a descriptor was introduced by

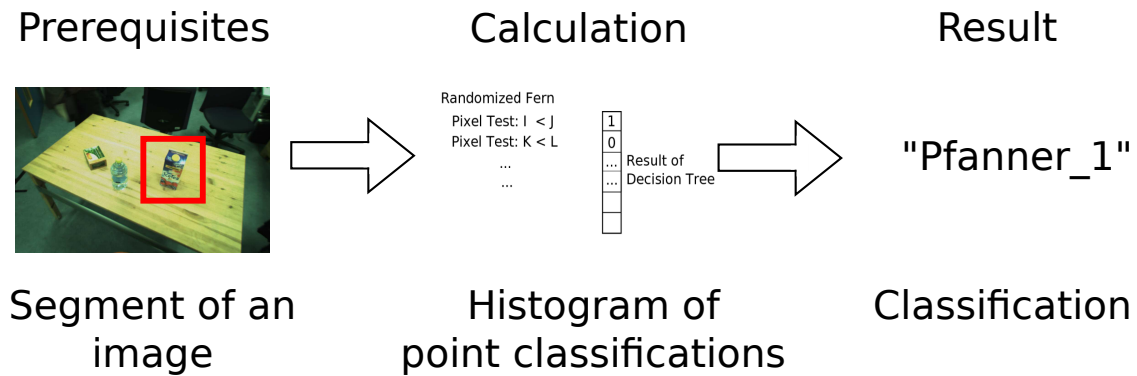


Figure 5.11.: Prerequisites, Calculation and Results of Randomized-Fern-based Classification.

Lepetit et al. in [66] and it is a classification using a simple decision tree to distinguish between classes of key points. Originally, this approach was used for wide base line matching, but with some modification it can work similarly on a bag of visual word approach as an object classifier.

Figure 5.12 shows an image and a zoom to a patch around a key point detected with the Harris key point detector [41]. This figure displays also how the feature is computed: For a list of randomly selected pixel an intensity comparison is calculated and the event is stored in a bit of a 2^h long variable u . h designates in this context the height of the classification tree. In a training phase a set of decision trees is randomly created and an a posteriori distribution is learned for each of them for a set of distinct key points. The learning of the distribution is performed by saving for every outcome of u how often a certain key point had this specific outcome under different views. The a posteriori distribution for all leafs is stored and used for classifying a new key point to be similar to one of the training key points.

Instead of a decision tree a decision fern gives the same results and is smaller to be stored [88]. A subsequent clustering of the distributions for the different classes of points gives information about similar patches, and allows reducing the number of points. The number of different distributions depends highly on the repetitiveness of the samples given.

Training Step

Independent of the samples, which were used to create the distribution, this classifier can be used to count occurrences of point classes in previously unseen objects. To create a recognizable object we observe the object in as many positions as we can and calculate statistics how often which point is classified as which of the previously trained classes. The training step uses then this histogram, usually normalized, how often which classes appears to reflect a

5. Object Recognition and Categorization

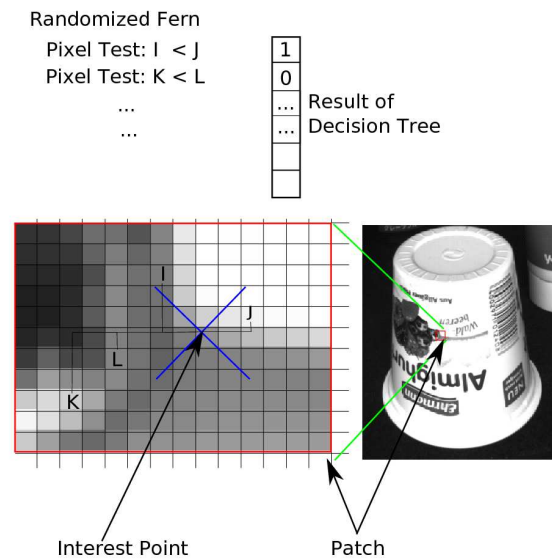


Figure 5.12.: A randomized fern descriptor

point on the surface of the object. Depending on the time which is available for training and matching, this can be extended by storing the information which interest points appear on the object and which are their most probable classification results.

Online Key Point Classification

Online we can apply then the same principle to create a similar vector again in order to create a comparable feature. This requires a similar segmentation as was used for the previous training step. The final classification uses then the previously learned classifier, e.g. a SVM, and returns the most probable class. False positives are rejected using the feature count and the classifier score, if existing.

Tests have shown, that it is possible without any long training to perform well enough to distinguish classes, with a tendency to false positives. The requirements on the quality of the images is a bit lower than in the bag of visual word approach, while also the number of distinguishable classes is lower.

5.3. Rigid Textured Object Localization

A well studied problem in computer vision is the localization of textured object using only one camera. The common assumptions in most of the work are that it is possible to find in

the texture of the object significant key points. Recognizing those key points and deriving information from this recognition is per se not reliable and requires consistency checks.

We propose here a method to localize 3D objects based on an interest point classification. This method has several new features compared to the existing work.

5.3.1. Related Work

Current solutions for detection of textured objects mostly rely on Gradient features like SIFT ([68]). Alternatively, they rely on statistical methods like the randomized fern approaches based on [66]. Both approaches build a set of Point to Point correspondences, either 2D to 2D, or 2D to 3D.

The problem to estimate movements and poses out of those correspondences was addressed in works about pose estimation like [120, 79]. But all those pose estimation methods are sensitive to outliers in the set of Point correspondences which required new techniques for the most popular outliers suppression method the Random Sampling Consensus (RANSAC). A comprehensive summary of current solutions for a RANSAC for this pose estimation problem can be found in [103]. Problematic with RANSAC solutions is, that starting from a certain outliers-ratio, tests to verify if a sampling produces and inlier are not stable anymore. So most approaches building on RANSAC can only support high quality data, which prevents it being applied to a large set of applications.

Alternatives for the robust estimation by RANSAC are the Natural 3D Markers proposed by [44]. Extending the ideas of which kind of errors might occur, besides wrongly annotated background are misclassified points on the object. This problem can be addressed given a good analysis of the expected error probabilities [49] and modeling the discriminativity of all tracked features.

The original techniques of SIFT by Lowe was improved by many different publications, e.g. its now possible to be used in real time on a mobile phone [137]. Some reformulations of the descriptor are simply faster to compute like e.g. the variants SURF and DAISY [7, 130]. Anyways, those solutions cannot provide significantly more stability in general applications [132], since they base on the same assumption about the repeatability of key points. All variants bear problems in representing text and repetitive patterns. Their major advantage on being computed locally is also their biggest problems. Some of those problems can be avoided by using different image scales [83], but not all objects which humans are considering as textured can be used with SIFT-based or SIFT-similar-based approaches. Still those methods have a speed advantage with a relatively short global search phase caused by the locally limited matching.

5. Object Recognition and Categorization

The range of application is as rich as the research on the methods: robotic manipulation based on SIFT-like features was done by several groups [106, 61, 5, 39]. It is often used to localize the camera in respect to a world that is assumed to be fixed, mostly in the context of Visual SLAM [100, 98, 59, 124]. Similar to Visual SLAM, [90], build 3D meshes of objects observed over time in real time.

5.3.2. Descriptor 3D

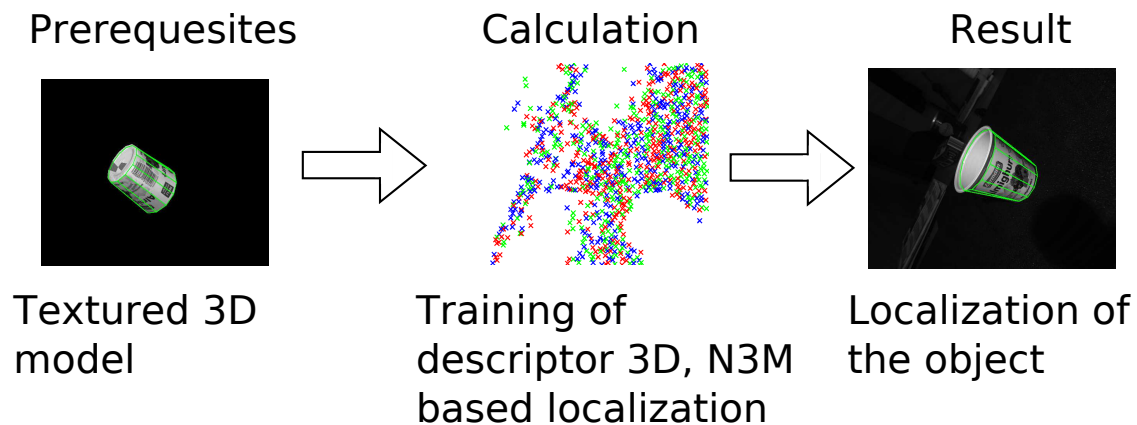


Figure 5.13.: Prerequisites, Calculation and result for Detection by Descriptor 3D.

Beginning with the methods introduced in Section 5.2.3 we can extend this approach to localize 3D Objects. Starting from a textured 3D model of an object, we can simulate all its views which helps to extend the ideas of the classification to a direct localization. By extracting and training key points from the simulated views, a model can be built that is able to detect the object in cluttered scenes. A method which does acquire textured models can be found in Section 6.3.

Model Simulation

A fast way to render a textured model is using OpenGL on a graphics card. This also allows rendering additional information in eventually unused color channels like face indices. Such information can be used to speed up processes of the selection of key points or the training of descriptors for those key points.

Keypoint Selection

The selection of a key point should result in the most repeatable points on the object from different viewing directions. Usually, Harris Points are relatively well repeatable, unless an

detected corner is geometrically caused.

In order to find out the most stable and repeatable key points, a large number of different view of the model is rendered. For every view the key point detector is applied and a re-projection of all points is performed back on the 3D model. On the model, the points can be easily compared with earlier points, and if a newly detected Harris point falls on the same 3D point on the model, this point gets an increased score. Finally, a limited number of points is selected based on their measured re-occurrences. Points, which are in neighborhoods with few other stable points, are kept preferred.

Additionally, the modeling we performed bears the risk of having inaccuracies nearby face borders. Since the reconstruction of the model is done face-wise, on borders of faces, intensity differences can occur based on fusion of differently lighted view serving for the texture of two neighbored faces. This leads to especially strong Harris Points on such model edges, which might be purely artificially introduced. So we exclude all feature point if they are too close to a corner point of the CAD model. This limitation is only applied to models with relatively large faces compared to the camera resolution. To determine this parameter, we estimate the average face size in pixel of the model in the mean training distance. From this we can estimate how much of the area will be removed, how close a point have to be to a 3D model corners to be removed.

Key Point Training

The training is performed by simulating the new object in all rotations and all allowed distances. For some viewpoints, the training data does not contain enough information. This is often the case, when the model was created by putting an object on a table and moving a camera around it.

The rendered views are sampled in 4 dimensions: Depth, and all three Rotations. In every view we project the selected key points from the model in the scene and test their visibility. If they are visible we the outcome of all randomized trees is calculated. Every Tree accumulates an a posteriori distribution to connect the tree outcome with the index of the key point on the model.

The visibility test contains two elements: The projection must be on the same face as it was trained (prevent to learn self occlusions) and the point must look relatively frontal to the camera.

5. Object Recognition and Categorization

Clique Calculation

The training finally results in an a posteriori distribution of the random fern. We estimate an algebraic distance between the estimated a posteriori distributions for all classes. If two of those distributions are too similar, the classes are fused into a clique.

This clique has the effect, that if a candidate point is classified as a member of this clique, all others appear as possible classification result as well. The a posteriori distributions $D_i, i = 0 \dots \|\text{Points}\|$ resulting from the training are compared using the algebraic distance:

$$d(D_i, D_j) = \sum_{a=0}^N \|D_i[a] - D_j[a]\|_1 \quad (5.4)$$

The distances between all i and j are used to estimate a normal distribution over the distance. Any pair of points is checked whether its distance varies from the mean distance $\text{mean}_{\forall i,j}$ more than three times the covariance $\sigma_{\forall i,j}$:

$$d(D_i, D_j) > \text{mean}_{\forall i,j} (d(D_i, D_j)) + 3 \sigma_{\forall i,j} (d(D_i, D_j)) \quad (5.5)$$

In the case this distance is lower, the class j will be assumed as part of the clique of the class i and vice versa. We keep the list for every class that contains all classes that are close enough. This will be necessary for the fast lookup in the online phase, whether a class, which is resulting from an applied randomized tree, is significant or not, and if other classes have to be considered, too.

N3M Selection

An N3M is following to Hinterstoisser et al. [44] a Natural 3D Marker, that consists of 3 or 4 key points defining a marker and one additional checker point that can be used to verify a matching of the marker points. This is a concept allows to create a verified hypothesis of an objects position.

We want to demonstrate here the case of a 4-key-points marker, since the planar case is well described in [44]. In the non-planar case we need to select 5 key points, which are relatively close together. Of those five points, four are describing a marker and one is the checker point. This checker point p_c is predicted as using the measurement of the other points in linear combination:

$$p_{ch} = p_0 + c_1 v_1 + c_2 v_2 + c_3 v_3 \quad (5.6)$$

5.3. Rigid Textured Object Localization

where p_i are the marker points in 3D space and v_i their difference of p_i and the first marker point p_0 :

$$v_i = p_i - p_0 = \begin{pmatrix} v_{i_x} \\ v_{i_y} \\ v_{i_z} \end{pmatrix} \quad (5.7)$$

$$p_0 = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} \quad (5.8)$$

Eq. 5.3.2 is only valid if the four point are not coplanar, which would mean that they can be used as a 3-key point marker. Solving this equation for the parameters c_1, c_2 and c_3 results in the following and can be calculated while learning the descriptor:

$$c_1 = -\frac{-v_{2_x}p_yv_{2_z} + v_{2_x}v_{3_y}p_z + v_{3_x}v_{2_z}p_y - v_{3_x}p_zv_{2_y} + p_xv_{2_y}v_{3_z} - p_xv_{3_y}v_{2_z}}{-v_{1_x}v_{2_y}v_{3_z} + v_{1_x}v_{3_y}v_{2_z} + v_{2_x}v_{1_y}v_{3_z} - v_{2_x}v_{3_y}v_{1_z} + v_{3_x}v_{1_z}v_{2_y} - v_{3_x}v_{2_z}v_{1_y}} \quad (5.9)$$

$$c_2 = \frac{-v_{2_x}v_{1_z}p_y - p_xv_{3_y}v_{1_z} - v_{1_x}p_yv_{3_z} - v_{3_x}p_zv_{1_y} + p_xv_{1_y}v_{3_z} + v_{1_x}v_{3_y}p_z}{-v_{1_x}v_{2_y}v_{3_z} + v_{1_x}v_{3_y}v_{2_z} + v_{2_x}v_{1_y}v_{3_z} - v_{2_x}v_{3_y}v_{1_z} + v_{3_x}v_{1_z}v_{2_y} - v_{3_x}v_{2_z}v_{1_y}} \quad (5.10)$$

$$c_3 = -\frac{-v_{1_y}p_zv_{2_x} + v_{1_y}p_xv_{2_z} - v_{1_z}v_{2_y}p_x + v_{2_x}v_{1_z}p_y - v_{2_z}p_yv_{1_x} + v_{1_x}v_{2_y}p_z}{-v_{1_x}v_{2_y}v_{3_z} + v_{1_x}v_{3_y}v_{2_z} + v_{2_x}v_{1_y}v_{3_z} - v_{2_x}v_{3_y}v_{1_z} + v_{3_x}v_{1_z}v_{2_y} - v_{3_x}v_{2_z}v_{1_y}} \quad (5.11)$$

Those parameters can be applied directly on a frontal parallel projection of the 5 points to predict the position with the same Equation just replacing all 3D points by 2D points. Assuming that we can create circumstances to have a projection of our object of interest and a possibility to localize the 5 points in this image of the projection, we can measure an error telling if the prediction about the location of the 5th point is correct regarding the relative position to the others. This can be expressed by the following equation using p'_i, p'_{ch} as the projection of the points and v'_i as the differences of the projected points from p'_0 :

$$\varepsilon = \|p'_{ch} - (p'_0 + c'_1v'_1 + c'_2v'_2 + c'_3v'_3)\|_2 \quad (5.12)$$

with:

$$v_1 = (1, 0, 0)^T \quad (5.13)$$

$$\theta_1 = \frac{p_2 - p_0}{\|p_2 - p_0\|} \cdot \frac{p_1 - p_0}{\|p_1 - p_0\|} \quad (5.14)$$

5. Object Recognition and Categorization

$$\theta_2 = \frac{p_3 - p_0}{\|p_3 - p_0\|} \frac{p_1 - p_0}{\|p_1 - p_0\|} \quad (5.15)$$

$$\theta_3 = \frac{p_3 - p_0}{\|p_3 - p_0\|} \frac{p_2 - p_0}{\|p_2 - p_0\|} \quad (5.16)$$

$$v_2 = (\sin(\theta_1), \cos(\theta_1), 0) \quad (5.17)$$

$$v_3 = (\sin(\theta_2)\cos(\theta_3), \sin(\theta_2)\sin(\theta_3), \cos(\theta_2)) \quad (5.18)$$

In a real camera we have of course not a frontal parallel projection but more an approximated pinhole camera projection.

This allows us to approximate the error introduced by the assumption of a frontal parallel camera, which is contradicting the camera matrix C , under the relative position of the object in the camera P and a projection of the model point:

$$p'_i = CPp_i \quad (5.19)$$

$$\delta_\varepsilon = (p'_0 + c_1v'_1 + c_2v'_2 + c_3v'_3) - (C(Pp_0 + P(c_1v_1 + c_2v_2 + c_3v_3))); \quad (5.20)$$

First we can eliminate the single appearance of p_0 using Eq. 5.3.2 and replace all projected points with it:

$$\delta_\varepsilon = (p'_0 + c_1p'_1 - c_1p'_0 + c_2p'_2 - c_2p'_0 + c_3p'_3 - c_3p'_0) - (C(P(p_0 + c_1p_1 - c_1p_0 + c_2p_2 - c_2p_0 + c_3p_3 - c_3p_0))); \quad (5.21)$$

If we replace all 2D points with 3D points, we come to the following equation:

$$\begin{aligned} \delta_\varepsilon = & (c_1CPp_0 + c_1CPp_1 - c_1CPp_0 + c_2CPp_2 - c_2CPp_0 + c_3CPp_3 - c_3CPp_0) \\ & - (C(P(p_0 + c_1p_1 - c_1p_0 + c_2p_2 - c_2p_0 + c_3p_3 - c_3p_0))); \end{aligned} \quad (5.22)$$

Without loss of generality we can assume that P contains already the translation induced by p_0 that we can set $p_0 = (0, 0, 0)^T$:

$$\delta_\varepsilon = ((1 - (c_1 + c_2 + c_3))CP(0, 0, 0)^T + c_1CPp_1 + c_2CPp_2 + c_3CPp_3) \quad (5.23)$$

5.3. Rigid Textured Object Localization

$$-(C(P(c_1p_1 + c_2p_2 + c_3p_3)))));$$

If we consider now P as an Euclidean transformation matrix parameterized with three angles α, β, γ and a translation vector $(t_x, t_y, t_z)^T$, we can write Pp as (by using abbreviations for $\sin(\alpha) = s_\alpha, \cos(\alpha) = c_\alpha, \sin(\beta) = s_\beta, \cos(\beta) = c_\beta, \sin(\gamma) = s_\gamma, \cos(\gamma) = c_\gamma$):

$$p_t = Pp = \begin{pmatrix} c_\beta c_\gamma p_x - c_\beta s_\gamma p_y + s_\beta p_z + t_x \\ (s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma) p_x - (s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma) p_y - (s_\alpha c_\beta) p_z + t_y \\ -(c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma) p_x + (c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma) p_y + (c_\alpha c_\beta) p_z + t_z \end{pmatrix} \quad (5.24)$$

If we additionally take advantage of the fact, that for a pinhole camera the distance to the camera has a linear correlation with the observed error in row and column. Or more mathematically, we can replace t_z by 1 in this matrix and multiply the resulting instance with $1/t_z$. Speaking more intuitively: the error of the N3M prediction (or any observed distance) correlates linear in both dimensions with the distance of the object with the camera. The distance of the camera can be assumed as larger than zero, to allow the camera to see the object. This allows us to define the translation vector in the camera as $(t_x, t_y, 1)^T$ given we are only interested in a relative comparison of the errors regarding different N3Ms.

$$p_t = Pp = \begin{pmatrix} (c_\beta c_\gamma) p_x - (c_\beta s_\gamma) p_y + s_\beta p_z + t_x \\ (s_\alpha s_\beta c_\gamma + c_\alpha s_\gamma) p_x - (s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma) p_y - (s_\alpha c_\beta) p_z + t_y \\ -(c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma) p_x + (c_\alpha s_\beta s_\gamma + s_\alpha c_\gamma) p_y + (c_\alpha c_\beta) p_z + 1 \end{pmatrix} \quad (5.25)$$

If we assume now that we have a simple pinhole camera with an exactly quadratic sensor size we can additionally eliminate t_y and all terms with $\cos(\gamma)$. This is usually not exactly true, but can be simulated given a good camera calibration. For calculating good thresholds, we also have to consider discretization errors, which are not relevant for the error approximation. Intuitively spoken, with a rotation around the cameras principal axis, we do not change any distance in the image plane. This allows us to rotate the virtual camera frame as long as the component t_y of the object's position becomes 0 and define this new position as $\gamma = 0$. This simplification results for a single transformation in:

$$p_t = Pp = \begin{pmatrix} c_\beta p_x + s_\beta p_z + t_x \\ s_\alpha s_\beta p_x - c_\alpha p_y - s_\alpha c_\beta p_z \\ -c_\alpha s_\beta p_x + c_\alpha p_y + c_\alpha c_\beta p_z + 1 \end{pmatrix} \quad (5.26)$$

5. Object Recognition and Categorization

If we stick to the pinhole assumption with a quadratic sensor size depending on the camera constant f we can replace Cp_i with:

$$Cp = \begin{pmatrix} \frac{-fp_x}{p_z} \\ \frac{-fp_y}{p_z} \\ p_z \end{pmatrix} \quad (5.27)$$

If we combine those simplifications and substitute them in Eq. 5.3.2 we get the following term:

$$\frac{\delta_\epsilon}{ft_z} = \begin{pmatrix} (1 - c_1 - c_2 - c_3)(-t_x) + c_1 \frac{-(c_\beta p_{1,x} + s_\beta p_{1,z} + t_x)}{-c_\alpha s_\beta p_{1,x} + c_\alpha p_{1,y} + c_\alpha c_\beta p_{1,z} + 1} + \\ c_2 \frac{-(c_\beta p_{2,x} + s_\beta p_{2,z} + t_x)}{-c_\alpha s_\beta p_{2,x} + c_\alpha p_{2,y} + c_\alpha c_\beta p_{2,z} + 1} + c_3 \frac{-(c_\beta p_{3,x} + s_\beta p_{3,z} + t_x)}{-c_\alpha s_\beta p_{3,x} + c_\alpha p_{3,y} + c_\alpha c_\beta p_{3,z} + 1} \\ c_1 \frac{-(s_\alpha s_\beta p_{1,x} - c_\alpha p_{1,y} - s_\alpha c_\beta p_{1,z})}{-c_\alpha s_\beta p_{1,x} + c_\alpha p_{1,y} + c_\alpha c_\beta p_{1,z} + 1} + c_2 \frac{-(s_\alpha s_\beta p_{2,x} - c_\alpha p_{2,y} - s_\alpha c_\beta p_{2,z})}{-c_\alpha s_\beta p_{2,x} + c_\alpha p_{2,y} + c_\alpha c_\beta p_{2,z} + 1} + \\ c_3 \frac{-(s_\alpha s_\beta p_{3,x} - c_\alpha p_{3,y} - s_\alpha c_\beta p_{3,z})}{-c_\alpha s_\beta p_{3,x} + c_\alpha p_{3,y} + c_\alpha c_\beta p_{3,z} + 1} \end{pmatrix} - \begin{pmatrix} \frac{-(c_\beta(c_1 p_{1,x} + c_2 p_{2,x} + c_3 p_{3,x}) + s_\beta(c_1 p_{1,z} + c_2 p_{2,z} + c_3 p_{3,z}) + t_x)}{-c_\alpha s_\beta(c_1 p_{1,x} + c_2 p_{2,x} + c_3 p_{3,x}) + c_\alpha(c_1 p_{1,y} + c_2 p_{2,y} + c_3 p_{3,y}) + c_\alpha c_\beta(c_1 p_{1,z} + c_2 p_{2,z} + c_3 p_{3,z}) + 1} \\ \frac{-(s_\alpha s_\beta(c_1 p_{1,x} + c_2 p_{2,x} + c_3 p_{3,x}) - c_\alpha(c_1 p_{1,y} + c_2 p_{2,y} + c_3 p_{3,y}) - s_\alpha c_\beta(c_1 p_{1,z} + c_2 p_{2,z} + c_3 p_{3,z}))}{-c_\alpha s_\beta(c_1 p_{1,x} + c_2 p_{2,x} + c_3 p_{3,x}) + c_\alpha(c_1 p_{1,y} + c_2 p_{2,y} + c_3 p_{3,y}) + c_\alpha c_\beta(c_1 p_{1,z} + c_2 p_{2,z} + c_3 p_{3,z}) + 1} \end{pmatrix} \quad (5.28)$$

The final error measurement can now be sampled using this formula out the possible occurrences of this N3M on the model and the average error we get gives a very good approximation of the quality of this N3M. The quality can be used to select stable N3Ms in favor of unstable, which results in a significant quality boost in the N3M selection. Just taking the absolute distances of the points and their relative location following the idea of Hinterstoisser et al. resulted in the 3D case in far too few or too many N3Ms, with a large number of unstable N3Ms.

N3M Detection and Evaluation

Online, a Boolean matrix for all detected Harris is calculated to decide later in a fast manner, if two candidates can be part of the same N3M. This can be efficiently done since the decision is based on a hierarchy of comparisons that requires usually no calculation. The preliminary classification into model points has to be calculated before.

The first step in this hierarchy is a minimal and maximal distance in pixel can be defined that allow two points being part of the same N3M or not. This is directly derived from the selection mechanism described before 5.3.2, that does only allow non-collinear and nearby

5.3. Rigid Textured Object Localization

points. The same threshold can be projected into the camera, which gives a constant maximal and minimal distance, which serves as a simple decision criteria for a possible N3M.

The next decision is a look up in the neighborhood table, which was calculated in the model generation phase in order to select possible N3M candidates. This table contains for every class a list of other classes that were initially close enough. This calculation is done after the unrolling into clique members, so all ambiguities by cliques are supported.

The last criterion tests, if the range of possible projections of the point and the specific neighbor contains the measured pixel distance. This test is only calculated a few times, so this two values are not pre-computed to save memory (requires two doubles per $n_{classes}^2$).

Pose Refinement



Figure 5.14.: Matched positions of a yogurt cup.

5. *Object Recognition and Categorization*

The pose refinement is based on an iterative approach using the pose estimation presented in [120]. The pose estimation uses first the five N3M points and then iteratively adds all points that have a projection of a model point close enough to a corresponding classification of a candidate point.

If the process produces enough inliers, also all Harris points are compared with predictions of model points. In case the prediction of a model point is close enough to any Harris point, this point is additionally used for a final guided matching.

This process is done until either a stable candidate is derived or the maximal allowed number of tests on N3Ms is performed. Figure 5.14 shows a textured object that was matched in a simple scene. Results of tests using this method can be found in the Results Chapter in the Section 8.1.3.

5.4. Summary

We described in this chapter methods that are capable to recognize certain properties or positions of known objects, which can lead to a more precise world belief state. We discussed visual properties that lead to clues that can be used to get more reliable results with a specialized method.

Strong geometric distinctness or certain texture properties can be used to describe objects in a way that they are either recognizable in a cluttered environment or that they can at least be classified as a part of a certain class of objects. Some of the methods are more sensitive than others to environment changes, like changing outside light sources or another sensors setup than others. We have developed here a set which allows a robot to robustly find this object, by just providing the right model.

Unfortunately, creating and giving the perception system the right models is the biggest problem here. All of the yet presented methods require training data or a model in order to be able to recognize something. The so far remaining question for those approaches is how to create those models. This is discussed in the next chapter.

6. Learning Perceptual Models in Domestic Environments

Most of the discussed localization methods require some model information. An example for required information is a CAD model. If this kind of information is not present in the robotic system, some localization methods are not applicable. In order to be able to profit from the variety, we present here a list of methods to infer further models.

Starting from a segmentation, we can learn color and shape models. From a cad model we can learn a texture model, or other visual representations. Additionally we present here methods which acquire models from the World Wide Web.

6.1. Simple Model Inference

After a successful object localization, a new task can be triggered that returns instead of a new pose a new model that will be connected with the semantics that were used for the search. Examples for actually implemented methods for deriving new models are for example the learning of an color model, or a template learning of the texture and shape for planar patches. Using such simple approaches, it is easy to create examples for certain objects and object classes in an semiautomatic way.

6.1.1. Related Work

This modeling of objects by examples is a widely discussed problem and is often done semi-autonomously in order to achieve better performance. There are currently several challenges setup in this area, which are interesting to compare with the problem we impose here. E.g. the "Solutions In Perception Challenge" was proposed and discussed in a recent IROS Workshop [14]. It asks to be able to recognize objects in cluttered scenes. In difference to the presented approach here, the data which is provided of objects are assumed to be well segmented, rigid, textured and lambertian and views are available from all directions.

6. Learning Perceptual Models in Domestic Environments

Similar to this challenge, but less applicable to our scenario, while already releasing the assumption of texture, is the widely known PASCAL Visual Object Classes challenge [26], which trains classes of objects, but also gives a lot of training data beforehand for one object class to be recognized. In another direction goes the Semantic Robot Vision Challenge ¹, which requires a semantic description instead of an object instance to start from. Such data, fulfilling the above assumptions, cannot always be provided in an unstructured environment. Less assumptions are taken in the work presented in [142], which only expects that a robot has an object in its hands in order learn a textured 3D-model. Unsupervised Learning 3D Models, [108], assumes segmentation and validation over multiple views, which results on nice 3D models, but unfortunately requiring a static world. A bit different is the approach proposed in [75] which derives parameters for predefined model that are applied on movable objects to extract basic properties. This approach requires a model to explain all possible effects, and need also means to measure those effects, which both is not intrinsically verifiable.

6.1.2. Any to Color

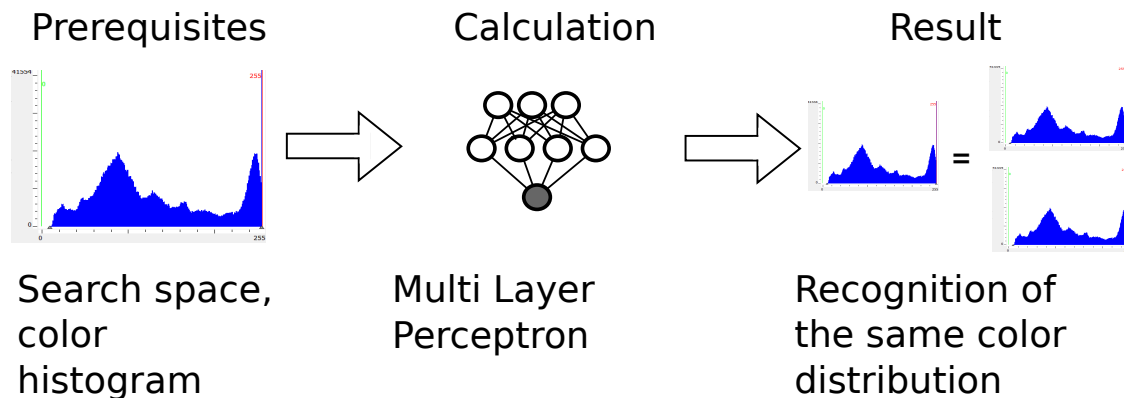
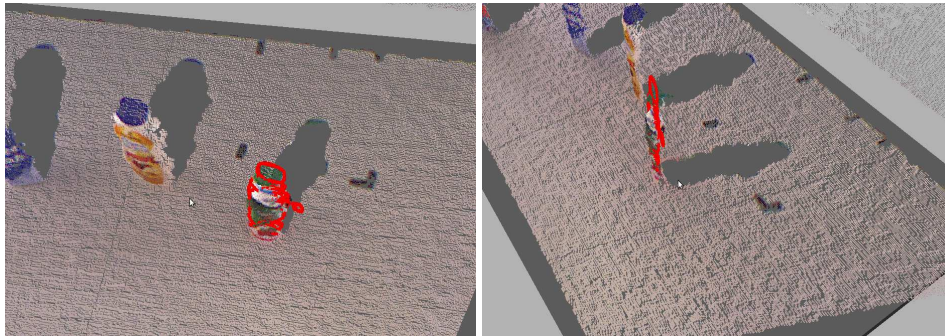


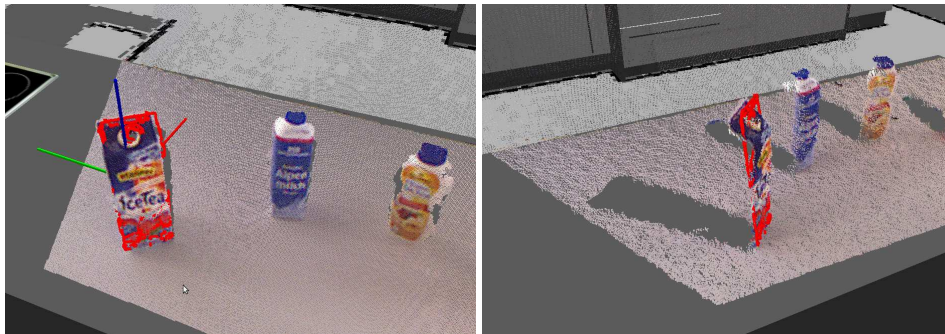
Figure 6.1.: Prerequisites, Calculation and result for Color Classification.

Given we have a context aware (illumination state: sun, artificial or indirect sun plus artificial illumination) pixel wise color classification, we can find out the most significant color class (we use currently only few like: red, green, blue, yellow, white, black). The precondition is that we can segment an object of interest in a color image, this we do usually by projecting a 3D cluster to the image. Even a point distribution like we get in from the candidate detection is enough for this purpose, if we only take the region covered by points inside σ instead of 2σ , which is our usual measure for search space sizes. For color classification we use a multilayer perceptron ([107], MLP) that was trained for several examples on images from

¹Website: <http://www.semantic-robot-vision-challenge.org/>.



(a) The contour which is projected on the assumed plane. (b) The same contour from a side view to see the difference between object and assumed plane.



(c) A contour for another object, frontal view. (d) A contour for another object, side view.

Figure 6.2.: Planar Shape model for localization of Objects.

all periods of the day. The annotation of colors was done manually, for simpler semantic connection.

This color classification can be applied to any geometric segmentation and a comparison with the learned color model by a distance over the small histogram of classified colors. This helps to distinguish objects without difficult matching or type resolution.

6.1.3. Any to Planar Shape Model

If we assume partial planarity of an object with sufficient size, we can learn a shape model of the object. It is helpful if this area has texture, but a prominent contour can be enough for a good recall. By saving a 3D position of the estimated planar substructure and image regions centroid, this template can be re-localized in 3D, see Figure 6.2. For extracting the main planar substructure we just calculate the SVD of segmented 3D points, and use the cross product of the first two Eigenvectors as a plane normal through the object's center. To get more a more robust segmentation of the object into different planes, methods like presented by [99] or [111] will result in better assumption.

6. Learning Perceptual Models in Domestic Environments

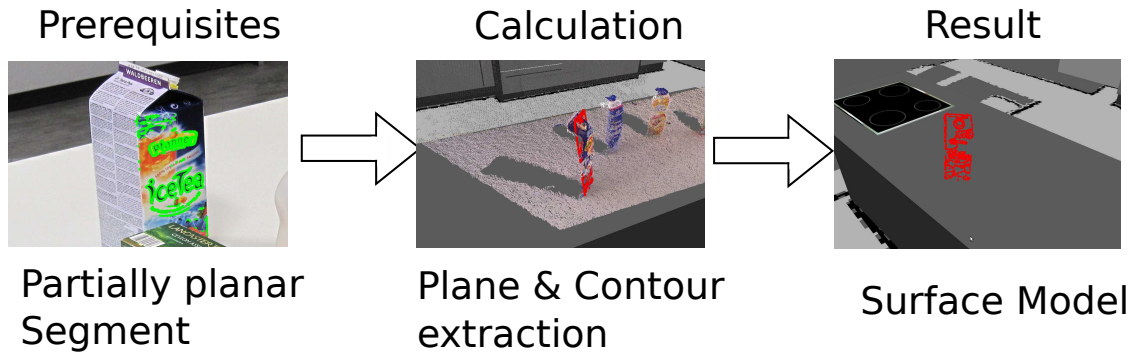


Figure 6.3.: Prerequisites, Calculation and result for Planar Shape Model Extraction.

A shape model, which is inferred from an earlier view of an object, serves to re-localize such an object. With such a template we can calculate a homography estimating the best 2D-2D hypothesis between the training image and the final image, using the method described in more detail in Section 5.1.3 which is based on [45].

6.1.4. Segment in 3D to Surface Model

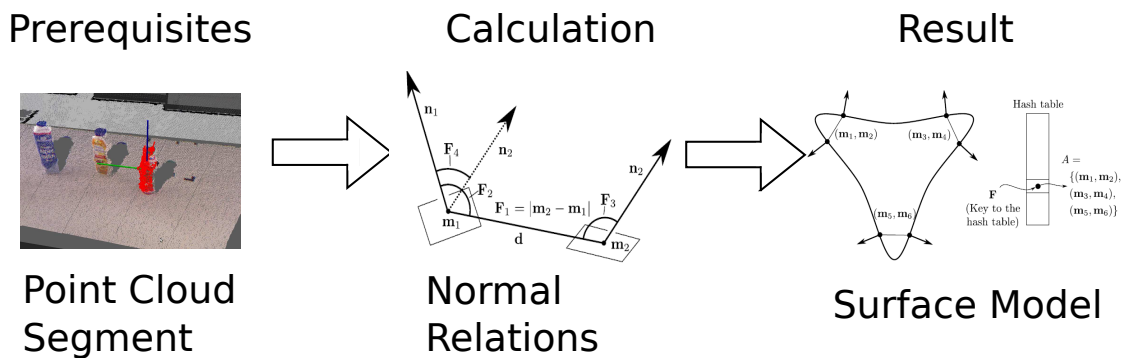


Figure 6.4.: Prerequisites, Calculation and result for Surface Model Extraction, parts of the images are taken from [20].

In order to automatically generate surface models that can be matched with the method described in Section 5.1.4, we have to extract a clean and smooth segment of a point cloud, from one or several views. The combination of views will not be discussed here, but it is possible to efficiently and robustly align several segment of point clouds to a nice description of an object, see [108] or [110].

If we have such a good segment with the normals on top of it we could directly learn a surface model representation. Unfortunately, the data is often too noisy to create really nice models. Smoothing of the normals helps sometimes, but the degree of smoothing has to be carefully adapted. Currently the implemented version only supports a fixed smoothing on

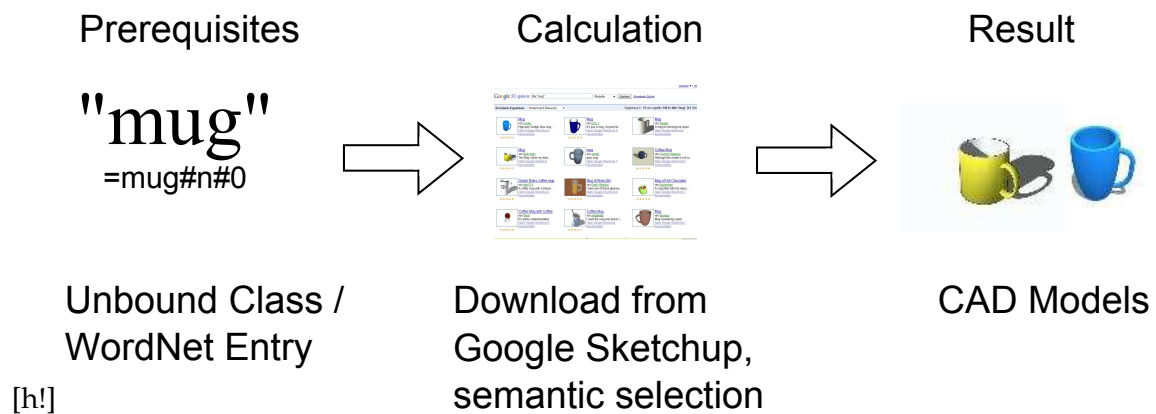


Figure 6.5.: Prerequisites, Calculation and result for acquisition of CAD models from Google 3D warehouse.

the points, which is tuned to the sensor noise in the expected working distance. The automatic estimation of such smoothing parameters is future work. Additionally, it is important to avoid the inclusion of supporting planes and background into the model.

6.2. Acquisition of CAD models

The deployment of service robots into human living environments imposes great challenges on their object recognition capabilities. Such robots will know abstractly which objects they have to manipulate namely cups, plates, bowls, pots, etc. But because the appearance of these objects varies substantially it will not know how they look. Also, in case of long-term operations, it will be inevitably for a robot to be confronted with situations in which it has to manipulate new kinds of objects. Moreover, since the robot's task does not end with the detection of the objects' existence but rather its purpose is manipulating them, in particular to pick them up and place them, having fairly accurate geometric models of them is a valuable resource for reliable and efficient robot operation. The main purposes of such models is the accurate localization of objects in world models, guessing their weight and center of mass, and determining appropriate grasp points for the objects. This vision method was presented in a first version in [57] operates as follows.

1. Given abstract instructions for the tasks the robot is to perform, all types of objects that the task refers to can be extracted, such as cups, plates, etc.
2. For each object type it looks up a library of geometric 3D object models (in this work we use Google 3D warehouse for this purpose) and retrieves prototypical 3D models for these object types.

6. Learning Perceptual Models in Domestic Environments

3. Given the prototypical 3D object models, which will only match the objects in the environment to a certain degree and typically lack accurate size specifications, the robot looks for objects in the environment that sufficiently match these 3D models.
4. The vision system then matches the object model to the image in order to select one of the prototypical models as an accurate one for the objects in the environment.

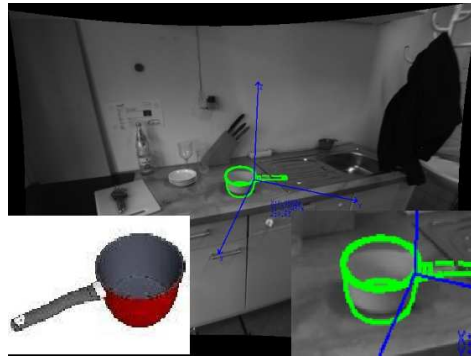


Figure 6.6.: A scene in that a model from Internet was found. The lower right corner zooms in on the match. In the left corner the model from the Internet that is used to calculate the shape model on the right.

This method is especially interesting for a learning phase after deployment of a service robot in a new environment. The method uses also a morphing algorithm that generates models that better fits variations of objects found in the environment. We show that the specialized models allow state-of-the-art shape matching algorithms with the additional use of some physical and semantic constraints in a known environment to efficiently detect and localize those objects. Fig. 6.6 shows a “cooking pot” that was found on a known counter in a kitchen environment.

6.2.1. Semantic-aware Selection of CAD Models

Query results from Google 3D warehouse will not always contain models with the intended content. This can have various reasons: First, the database might be annotated wrongly, which we can assume to happen not too often. More often, annotations are inaccurate or too general or ambiguous. A simple example would be results for the search term ‘cup’, which might point to coffee cups as well as to trophies.

The more information we have on the search context, the easier it is to overcome such problems in the query’s results. Given we have a WordNet synset as the search term, which we can assume inside COP, see Section 2.2.1, we can use the complete set of annotations, which Google 3D warehouse provides with the resulting model (the so-called `tags`), to infer a likelihood for a model in the results that it fits to the search term as it was meant. This

method for the selection based on those tags was implemented and tested by Pavel Mihailov in his Diploma Thesis [70] based on the previous version of the selection process presented in [57].

Distance Measurement

We base the likelihood on WordNet using different distance measurements. Those measurements are based on the relations defined for WordNet which were introduced by [91]. Most of those distance metrics are based on the *is-a* relation, which basically relates all nouns on WordNet on the basis of one entry being a specialization of another.

Such a distance measurement of two words can be applied on the search term's synset and all existing synsets for tags and titles of the model found on Google 3D warehouse. The smallest value distance is then a measurement approximating the relevance.

This assumption is supported by the results of an inquiry performed with 10 subjects asking them for relation between kitchen words. This results were put into correspondences with the results of the different distance measurements available for WordNet. The usage of such an inquiry to measure correlations of words was proposed by [73] and set into correlation with distance measurements on an early WordNet corpus by [104]. The results of the inquiry can be found in Tables 6.1 to 6.5.

Table 6.1 shows the results from [73] and a reproduction of the results asking the 10 subjects. It also contains the six distance measurements from WordNet for comparison. Tables 6.1(a) and 6.1(b) contain the correlations of the human judgments with the distance measurements of WordNet. They show that the several of those distances measurement correlate strongly with the human judgment.

This experiment was then applied to words from the kitchen domain, to find the distance measurement that is most valid for the domain we are interested in. The results from the inquiry are stated in Tables 6.3 and 6.4, and the correlation of the results with the WordNet distances are shown in Table 6.5. It turned out that there are at least three candidates which have a relatively high correlation with the collection human judgment. The definitions of the distance measurements can be found in [70].

The results were convincing, so that given the tag is correlated with model, a correlated tag will make the correlation of the model with the search term more probably than an uncorrelated tag. This leads to the conclusion, that it is helpful to use the tags and the distances to the search term.

Unfortunately the measurements themselves will not give a valid conclusion if the tags we are using are correlated with the search term or not. First the distance is only helpful in a

6. Learning Perceptual Models in Domestic Environments

Word 1	Word2	Miller Charles means(0-4)	Repl. means (0-4)	res (0-11,77)	jcn (0-inf)	lin (0-1)	lesk (0-inf)	vec (0-1)	hso (0-16)
car	automobile	3.92	3.75	7,0	inf	1	3576	1	16
gem	jewel	3.84	3.25	9.82	inf	1	816	1	16
journey	voyage	3.84	3.5	7.46	0.35	0.83	41	0.8	4
boy	lad	3.76	3.25	6.74	0.29	0.8	50	0.79	5
coast	shore	3.7	3.5	8.1	1.62	0.96	51	0.64	4
asylum	madhouse	3.61	3.5	10.67	2.47	0.98	42	0.77	4
magician	wizard	3.5	3.75	11.07	inf	1	145	1	16
midday	noon	3.42	3.75	9.57	inf	1	46	1	16
furnace	stove	3.11	3	2.49	0.06	0.23	52	0.58	5
food	fruit	3.08	1.25	1.78	0.09	0.16	34	0.3	0
bird	cock	3.05	0.75	6.94	0.27	0.79	115	0.66	6
bird	crane	2.97	0.5	6.94	0	0	18	0.36	5
tool	implement	2.95	3	6.31	0.85	0.91	125	0.36	4
brother	monk	2.82	1.25	10.16	0.07	0.21	147	0.43	4
crane	implement	1.68	0	3.45	0.08	0.33	6	0.16	3
lad	brother	1.66	0.5	1.9	0.08	0.24	10	0.42	3
journey	car	1.16	0.5	0	0.07	0	19	0.39	0
monk	oracle	1.1	0	1.9	0.06	0.18	4	0.13	0
food	rooster	0.89	0.25	0.06	0.07	0.08	8	0.12	0
coast	hill	0.87	0	6.14	0.22	0.73	11	0.23	4
forest	graveyard	0.84	0	1.17	0.06	0.11	7	0.09	0
monk	slave	0.55	0	1.9	0.07	0.2	13	0.25	3
coast	forest	0.42	0	1.17	0.06	0.12	10	0.16	2
lad	wizard	0.42	0	1.9	0.08	0.22	2	0.05	3
chord	smile	0.13	0	3.07	0.08	0.33	1	0.08	0
glass	magician	0.11	0	1.87	0.06	0.14	3	0.07	0
noon	string	0.08	0	0.78	0.07	0.09	2	0.09	0
rooster	voyage	0.08	0	0	0.05	0	1	0.03	0

Table 6.1.: Results from [73], [104] and a reproduction of the results.

6.2. Acquisition of CAD models

(a) Experiment of Miller and Charles [73]		(b) Reproduction by Mihailov [70]	
Distance Measurement	Correlation	Distance Measurement	Correlation
res	0.8	res	0.8
jcn	0.46	jcn	0.46
lin	0.73	lin	0.34
lesk	0.34	lesk	0.34
vector	0.89	vector	0.89
hso	0.66	hso	0.66

Table 6.2.: The resulting correlation of the inquiry and the WordNet relatedness measurements

relative way, meaning only a comparison is a tag is more correlated than another to the same word will help. But if a word is in WordNet in a more connected clique than others, it will have the significantly different distance measurements. This means we have to introduce some kind of normalization to the values we get from the distance measurement.

Here it is helpful, that we get a relatively large variety of objects from Google 3D warehouse for the kind of queries we are interested in. E.g. searching for "cup" resulted in 2593 at 01 December 2010. For most of them we get a set of tags and at least a title. So we have far over 2593 candidates of words which we can use to estimate the random guess for.

So we estimate the intrinsic deviation σ for the current search term. This per se is not yet a normalization factor, and does not suffice the requirements we have to a normalization term.

Threshold Learning

In order to get the relation between the deviation of the similarity measurement for a word and a judgment, if a term is part of our inlier set or not, we try to estimate this relation by manually annotating several examples.

For queries for knife, mug and spoon, a manual annotation has been performed. This results in the a posteriori distribution visualized in Figure 6.7 for the `res` similarity measurement. This helps to visualize the best value of the `res` similarity to decide if a tag is consistent with the search term or not. The graphics on the right side of the Figure show that there are different optimal decisions for different search terms. The optimum is visualized with the crossing of the red and the blue line. The first two rows show an optimal decision between values of 0.4 and 0.6 for the `res` similarity measurement, while the last rows would have the optimal threshold be between 0.6 and 0.8.

6. Learning Perceptual Models in Domestic Environments

Word1	Word2	Human Judgments means (0-4)	res (0-11,77)	jcn (0-inf)	lin (0-1)	lesk (0-inf)	vec (0-1)	hso (0-1)
spoon	architecture	0	2.49	0.06	0.22	46	0.06	0
cup	art	0.25	2.49	0.09	0.32	71	0.14	2
spoon	art	0.25	2.49	0.07	0.27	67	0.09	0
mug	beer	0.25	0.61	0.05	0.06	32	0.25	0
cup	beer	0.25	0.61	0.06	0.07	52	0.25	0
mug	beverage	0.5	0.61	0.06	0.07	45	0.39	0
cup	beverage	0.25	0.61	0.07	0.08	83	0.39	0
cup	breakfast	0.25	0.61	0.06	0.07	42	0.29	0
mug	breakfast	0.25	0.61	0.05	0.06	22	0.29	0
cup	car	2	5.32	0.19	0.67	151	0.14	4
spoon	car	0	5.32	0.12	0.57	160	0.25	0
cup	coffee	0.25	1.37	0.06	0.07	88	0.4	0
mug	coffee	0.5	1.37	0.05	0.06	52	0.4	0
cup	coffee cup	3.25	8.99	0.48	0.9	231	0.49	16
mug	coffee mug	3.5	11.77	0	0	67	0.61	16
mug	cup	2.75	5.32	0.1	0.51	55	1.0	3
mug	dishes	1.5	5.32	0.1	0.51	66	0.19	3
cup	dishes	0.75	8.43	0.71	0.92	978	0.19	5
cup	food	0	0.61	0.08	0.08	116	0.2	0
mug	food	0	0.61	0.06	0.07	52	0.2	0
cup	glass	2.25	8.1	0.13	0.59	937	0.35	5
mug	glass	1.5	5.32	0.1	0.51	58	0.35	3
cup	handle	0.25	1.17	0.06	0.13	252	0.38	0
mug	handle	0.5	1.17	0.05	0.11	404	0.38	4
cup	kitchen	0.25	2.49	0.08	0.29	24	0.06	0
mug	kitchen	0.25	2.49	0.07	0.25	20	0.06	0
spoon	kitchen	0.5	2.49	0.07	0.25	24	0.07	0
frying pan	kitchen	1.0	2.49	0	0	18	0.07	0
cup	logo	0	0	0	0	11	0.04	0
mug	logo	0	0	0	0	12	0.03	0

Table 6.3.: An inquiry capturing semantic relation of kitchen objects, Part I, [70].

6.2. Acquisition of CAD models

Word1	Word2	Human Judgments means (0-4)	res (0-11,77)	jcn (0-inf)	lin (0-1)	lesk (0-inf)	vec (0-1)	hso (0-1)
cup	metal	0.25	0.61	0.07	0.08	111	0.13	0
mug	metal	0	0.61	0.06	0.07	56	0.13	0
spoon	metal	0.25	0.61	0.06	0.07	129	0.24	0
frying pan	metal	0.25	0.61	0	0	57	0.24	0
cup	plate	0.5	8.1	0.29	0.82	77	0.25	4
spoon	plate	0.75	8.1	0.16	0.72	81	0.16	0
cup	plastic	0.25	0.61	0.05	0.06	98	0.09	0
spoon	plastic	0.25	0.61	0.05	0.05	93	0.23	0
cup	racing	0.75	0	0.05	0	20	0.03	0
spoon	racing	0	0	0.04	0	16	0.03	0
cup	soup	1.0	0.61	0.06	0.06	68	0.14	0
spoon	soup	0.5	0.61	0.05	0.06	94	0.1	0
cup	tea	0.5	1.37	0.06	0.06	74	0.39	0
mug	tea	0.25	1.37	0.05	0.06	41	0.39	0
spoon	tea	0.5	1.37	0.05	0.06	71	0.08	0
cup	utensil	1.25	3.45	0.1	0.4	58	0.06	3
cup	warm	0.25	0	0	0	0	0	0
mug	water	0	2.49	0.06	0.22	67	0.13	0
knife	hunt	0.5	1.37	0.05	0	77	0.09	0
knife	car	0.25	3.45	0.1	0.41	270	0.18	0
knife	dagger	2.5	3.45	0.07	0.32	239	0.37	5
knife	sword	1	3.45	0.08	0.36	489	0.27	5
knife	weapon	2	3.45	0.11	0.44	216	0.19	0
knife	blade	2.25	8.43	0.08	0.36	786	0.35	6
knife	handle	0.25	1.17	0.06	0.12	629	0.35	6
knife	dinner	0.25	0.61	0.06	0.07	23	0.06	0
knife	food	0	0.61	0.07	0.08	141	0.26	0
knife	fork	0.25	7.49	0.15	0.69	85	0.34	3
knife	plastic	0.25	0.61	0.05	0.06	123	0.13	0
knife	tool	0.5	7.49	0.4	0.86	253	0.19	5

Table 6.4.: An inquiry capturing semantic relation of kitchen objects, Part II,[70]

Distance Measurement	Correlation
res	0.5984
jcn	0.2777
lin	0.3583
lesk	0.2543
vector	0.5745
hso	0.66

Table 6.5.: The Correlation of the inquiry results with WordNet distance measurements, [70].

6. Learning Perceptual Models in Domestic Environments

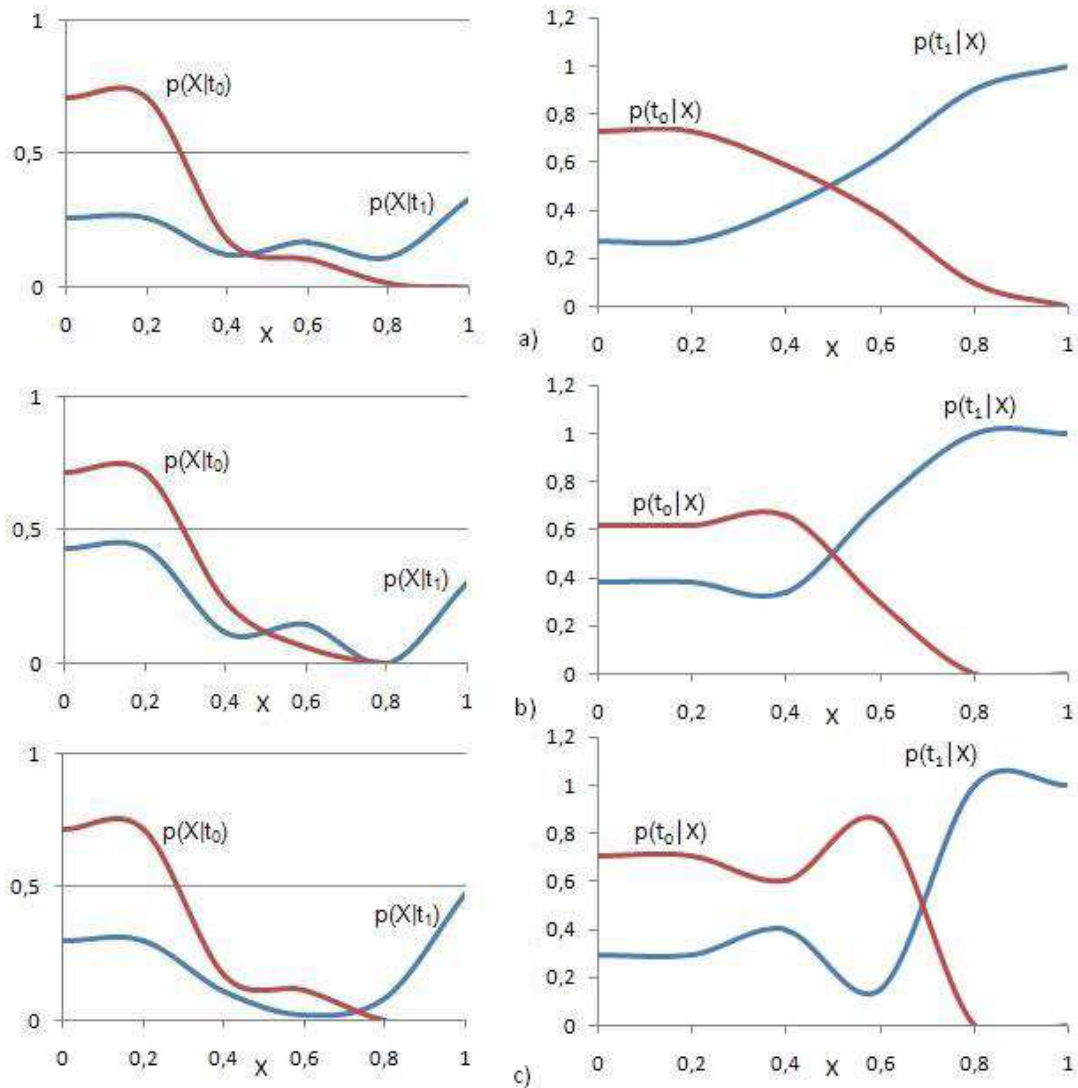


Figure 6.7.: The res -similarity given a related (t_0) or an unrelated term (t_1), $p(X | t)$. the a posteriori distribution of the relatedness given the res -similarity, $p(t | X)$. The first line shows the data for knife, the second for mug and the last for spoon [70].

The graph was generated the following way: any value returned from res was normalized and sorted into $N = 5$ buckets representing the values from $[0.0 - 0.2, 0.2 - 0.4, \dots, 0.8 - 1.0]$. For all tags found a res similarities to the search term was calculated and a class was annotated, t_0 for an inconsistent tag or t_1 for a consistent tag. This allows the representation shown in Figure 6.7 to visualize the probability density of the res similarities given a class $p(X | t)$ and the density of a class given a res similarity $p(t | X)$. If we introduce a threshold d we can estimate the expected error with the following equation

$$E[d] = \sum_{n=1}^N \begin{cases} p(X_n | t_0) & \text{if } X_n < d \\ 2p(X_n | t_1) & \text{if } X_n > d \end{cases} \quad (6.1)$$

If we want to get a threshold which is optimal for more than one search term, we applied a minimization for the following term for all search terms in the set W :

$$\arg \min_d \sum_{w \in W} E[d] \quad (6.2)$$

This results in a decision criterion d , which generalizes better for further search terms.

6.2.2. Model Selection with a Shape Distribution Function

Besides the semantic selection of models, we apply a selection mechanism for relevant models by performing a clustering techniques based on the shape distribution function proposed by [87]. On the shaped distribution we apply a distance measurement tested in [138] to find out the major cluster of the resulting models. Using a simple k-means algorithm, we are able to select inliers.

Assuming that there are more inliers than outliers in the results we want to select the largest cluster as inliers. To find similarities between 3D models we use the shape distribution function. This feature is calculated by randomly selecting points on faces of a model and describes the shape of the model. We are using k-means introduced by [29] for clustering. Caused by the outliers ratio we use $k = 4$ clusters, since we expect maximally 3 types of outliers. This step is also performed unsupervised and results in a set of 4-9 models.

The results from the World Wide Web search normally contain 20-40 percent outliers. We want to recognize them, before starting any further post-processing, or at least group the models in order not to align to different models. Actually, there are three different types of outliers in the results. First, ambiguities of the search string that result in different objects than meant. For example "cup" could mean a trophy or a liquid container. Second, assembled scenes containing the right object, for example an oven with a "cooking pot" on it. And third, wrongly labeled objects or semantically close models additionally showed by the search engine if there were no better matches. An example for this is a fork found searching for "spoon".

Figure 6.8 visualizes shape distribution for several classes of objects to show the interclass differences in this measurement and the intra class similarities.

6. Learning Perceptual Models in Domestic Environments

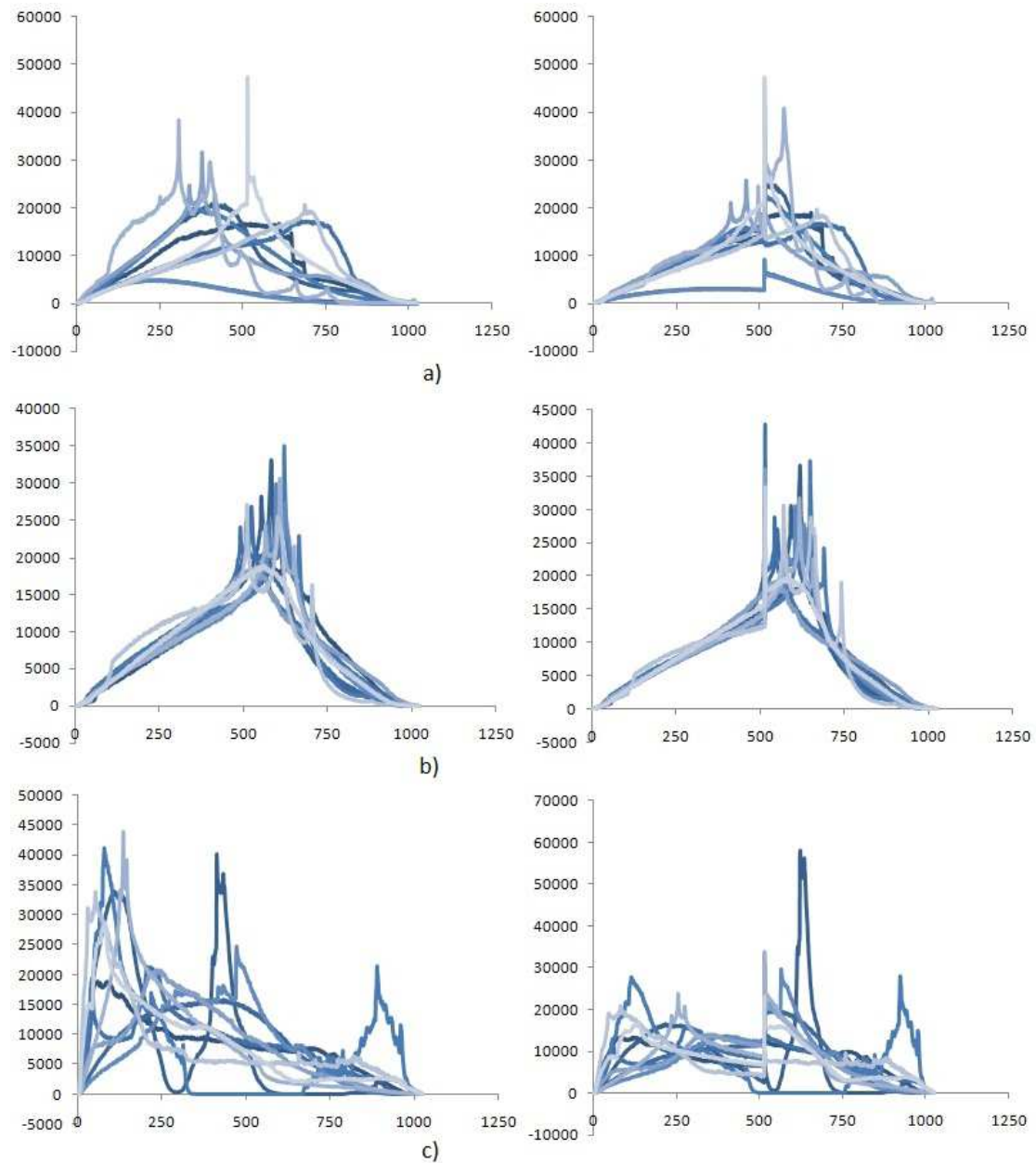


Figure 6.8.: Shape distribution histograms for 10 objects for the search terms *cup* (a) *mug* (b) and *fork*. All figures show histograms with 1024 bins with a sub-sampling of the objects using 4096 points. The left side is normalized using the maximum the right side using the mean. The image was taken from [70].

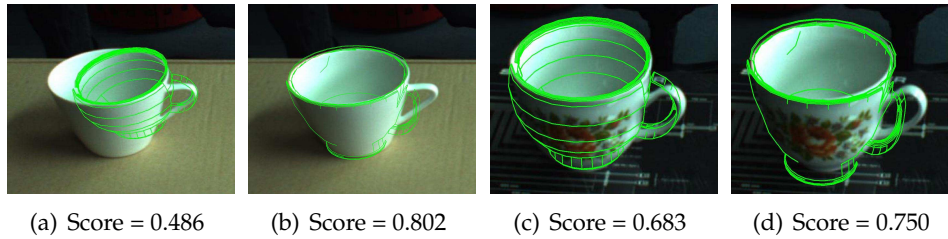


Figure 6.9.: Two real cups in our kitchen which better match one of the in-between models ($t = 0.25$) in Fig. 6.10, as compared to the original model ($t = 0$).

6.2.3. Morphing between CAD Models

Often we do not get enough models to perform a successful matching in a scene. Thus we create new models from the ones we already possess, for which we utilize morphing between models, and choose in-between models. This method was previously published in [147] and is summarized here.

Alignment of Models

Morphing process usually requires the presence of a human operator to perform an appropriate scaling, and alignment to make both models of same size and aligned[65]. We use the technique presented by [87] to form a histogram of distances between randomly selected points on the model-surface, and then pick the most commonly occurring distance (actually the middle point of the most frequent bin) and scale that distance to a constant value and the whole model is scaled isotropically. A similar technique is used for finding a statistical centroid and translating both the models so that the centroid lies at their origin.

Next we need to register the models against each other before we can start the morphing process. For this, one technique of interest is the Iterative Closest Point Algorithm (ICP) [12]. However, ICP works for only dense point-clouds; while the 3D models have only vertices (of the triangulated faces) which can be directly used for ICP. These vertices are densely packed at places where a high curvature is present in the model; and very few of these are present for comparatively flat surfaces. Thus, even if the curved surface has a small area, its alignment is given more weight by the ordinary ICP, as compared to a more planar surface with large area - which should not be the case. Thus, we used the technique presented by [87] to form such a dense point-cloud which has a distribution of points proportional to the surface area of the faces. This enables us to run a ICP with equivalent weight to all parts of the object.

6. Learning Perceptual Models in Domestic Environments

Table 6.6.: Morphing Algorithm

Step 1: Translate along z-axis and scale making the extent on both side equal.
Step 2: For every vertex in the source model, find the nearest point on the surface of the destination model.
Step 3: Introduce this nearest point as a new vertex in the triangulated model, and divide the triangle containing the vertex into three triangles.
Step 4: Store this as a mapping from the vertex in source model to the new vertex in destination model.
Step 5: Repeat step 2-4 reversing the two models.
Step 6: All vertices in both models now have an one-to-one mapping between them, and they are equal in number.
Step 7: Interpolate between the corresponding pairs of points using linear interpolation based on the parameter t .

Morphing



Figure 6.10.: A Morphing sequence - the first and the last models are obtained from Google 3D Warehouse.

Morphing is a technique that finds common use in computer animation. It involves transforming from one image or 3D model to another controlled by a parameter t going from 0 to 1. At $t = 0$, the resulting model is the same as the initial source model, while at $t = 1$ the resulting model is the same as the original destination model; whereas at values of $0 < t < 1$, we have a resulting model that is visually “in-between” the source and the destination model. Thus it is different from the original models, while maintain similar basic geometry in case the morphing is performed between objects of similar shape (which is the case of interest to us). We exploit this technique to generate new models when we fail to obtain models that fit the objects in the scene well enough.

Our approach yields particularly good results when the models have rotational symmetry around a line (detecting symmetry in 3D models is a well-studied problem [125]). Many of the models that we find useful for our kitchen environment have such a geometry that we

may assume the z-axis to be the principle axis of the models. In the following therefore, we assume that the axis of symmetry is the z-axis. The extent of both models is made equal on both sides of the x-y plane by translating and scaling slightly. In the following, we refer to one of the two models (between whom we perform the morphing) as the source model (corresponding to the morphing parameter $t = 0$) and the other model as the destination model (corresponding to $t = 1$). For each vertex in the source model, we take the x-y plane on which it lies, and intersect the triangles in the destination model with this plane, to obtain the nearest point on the surface of the destination model to the current vertex of the source model (we choose the nearest point out of the nearest points contributed by each plane). This is done by finding the intersection points of the x-y plane on the edges using Plücker lines [43]. The reason behind this major step is that the source and destination models will usually have different levels of detail. If we have two vertices A and B which form an edge, the edge is represented by L ; and p represents in homogeneous coordinates the x-y plane on which the source vertex s lies, then

$$L = \begin{bmatrix} 0 & A_x B_y - B_x A_y & A_x B_z - B_x A_z & A_x - B_x \\ A_y B_x - B_y A_x & 0 & A_y B_z - B_y A_z & A_y - B_y \\ A_z B_x - B_z A_x & A_z B_y - B_z A_y & 0 & A_z - B_z \\ B_x - A_x & B_y - A_y & B_z - A_z & 0 \end{bmatrix}$$

$$p = \begin{bmatrix} 0 & 0 & 1 & -s_z \end{bmatrix}^T.$$

Then the point of intersection u is given by $u = Lp$, and with another intersection point v on another edge of this face, we can find the nearest point on this line segment from the source vertex using parameter θ (obtained by taking equating the derivative of the distance to zero). If the points u , v , and p are represented in non-homogeneous coordinates, then c gives the desired closest point.

$$\theta = \frac{(u-p)(u-v)}{|uv|}$$

$$c = u + \theta v$$

This nearest point is introduced into the mesh of the destination model as a vertex by dividing the triangle that contains it into three (one new vertex in destination model for each original vertex of source model). We repeat this procedure, reversing the roles of the source and destination model; and store this one-to-one mapping. The introduction of new points into the models can be thought of as increasing the degrees of freedom of the model to take the shape of the other model and still looking pleasant. The last step is to perform a linear

6. Learning Perceptual Models in Domestic Environments

interpolation of the vertices from their position in source model to a their final position in the destination model. The complete algorithm is summarized in Table 6.6.

6.2.4. Selection by Best Match

Applying the method already presented in Section 5.1.2 to the image using several models we can use the score to compare the resulting score of different CAD models on the same part of the scene.

This gives a good approximation of the correspondence of a CAD model and the real object. The less cluttered the current scene is, the better this will work.

6.3. Texturing of Object Models

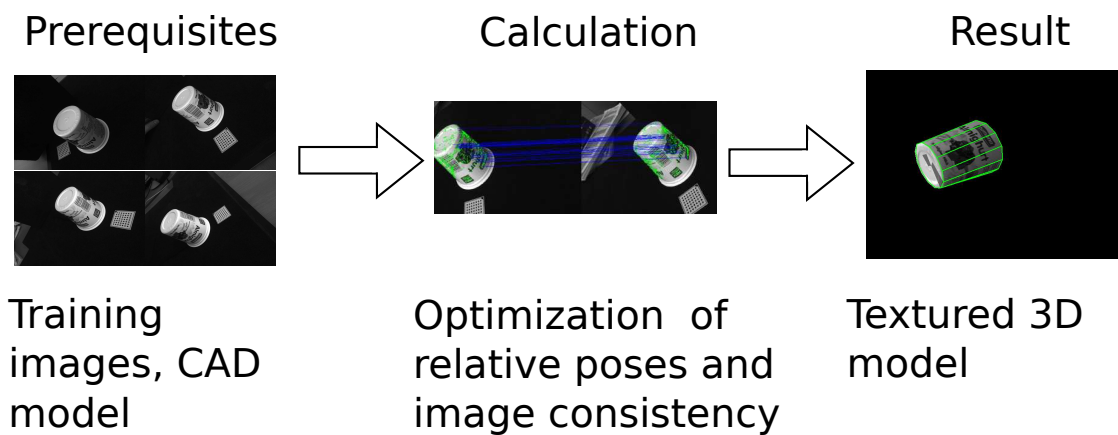


Figure 6.11.: Prerequisites, Calculation and result for texturing of 3D CAD models.

Given a known pose for several views we can create a texture models. A simplified setup, which does not necessarily have to take place on a robotic platform is depicted with some sample images in Figure6.12.

The assumption to have a CAD model will be used here for simplification of the explanation. The approach was also tested on reconstructed 3D data, and does bring results in similar quality, given the reconstruction is good enough.

6.3.1. Related Work

Reconstruction including texture was already tackled by [90] in the attempt to build with a single camera a globally consistent model. It uses image to images correspondences to

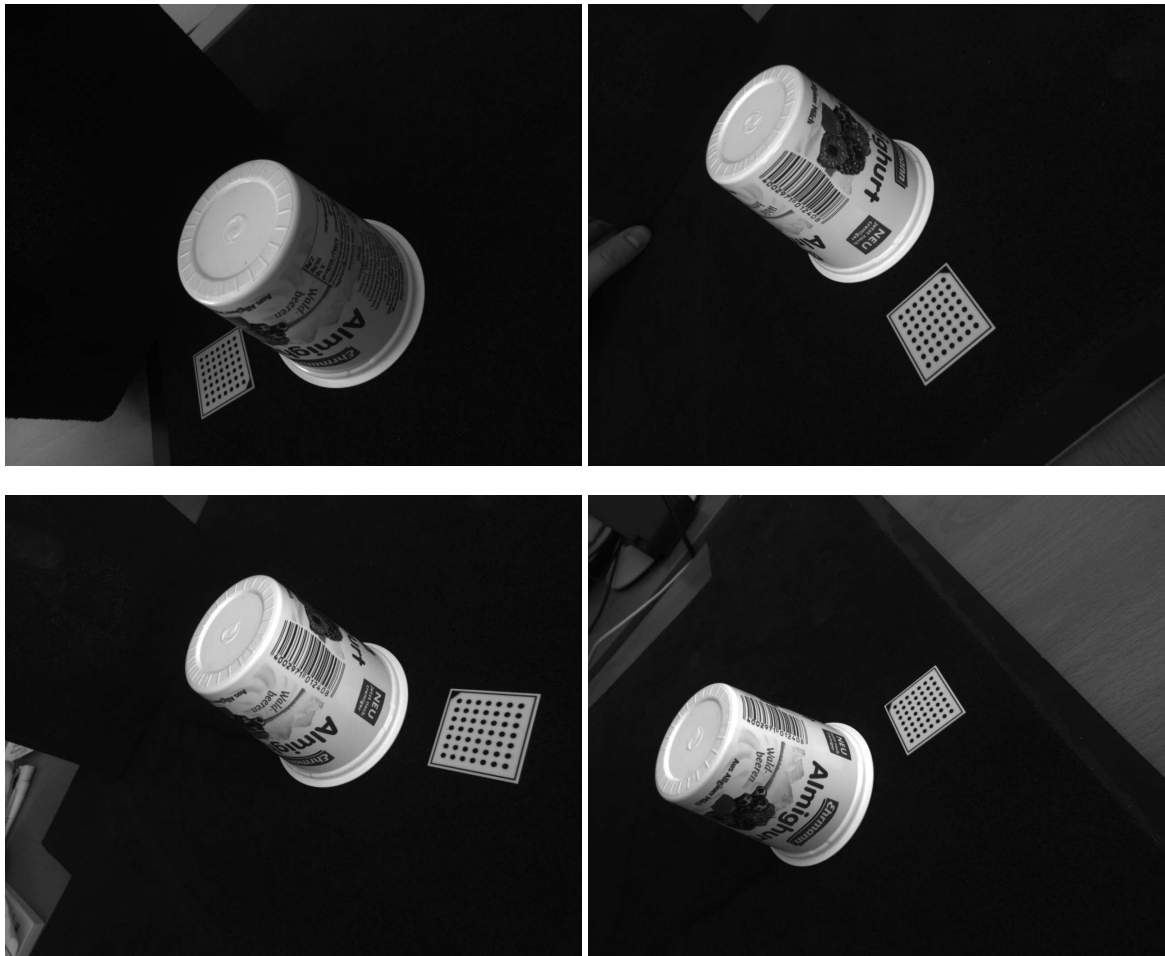


Figure 6.12.: Training images of a yogurt cup.

triangulate in an affine context points that are connected with triangles unless there is evidence that the occupied area is an overestimation. In contrast, we try to avoid the reconstruction step, given a CAD model is already available. Different approaches were presented by [15], [89] or [97], which all have in common, that they do not create a mesh but try to stay on a point cloud level while trying to reconstruct a full scene. A more accurate textured mesh is extracted by the work of [67] as well as [31] which tries to create a complete triangulation of a compact scene visible from several cameras.

6.3.2. Acquisition of Training Data

Knowing the positions of the images and the position of the object in at least one of them is necessary to get a good texturing of the CAD model. This information can be acquired using either a manual adaption of the position of the object or use a method like the edge

6. Learning Perceptual Models in Domestic Environments

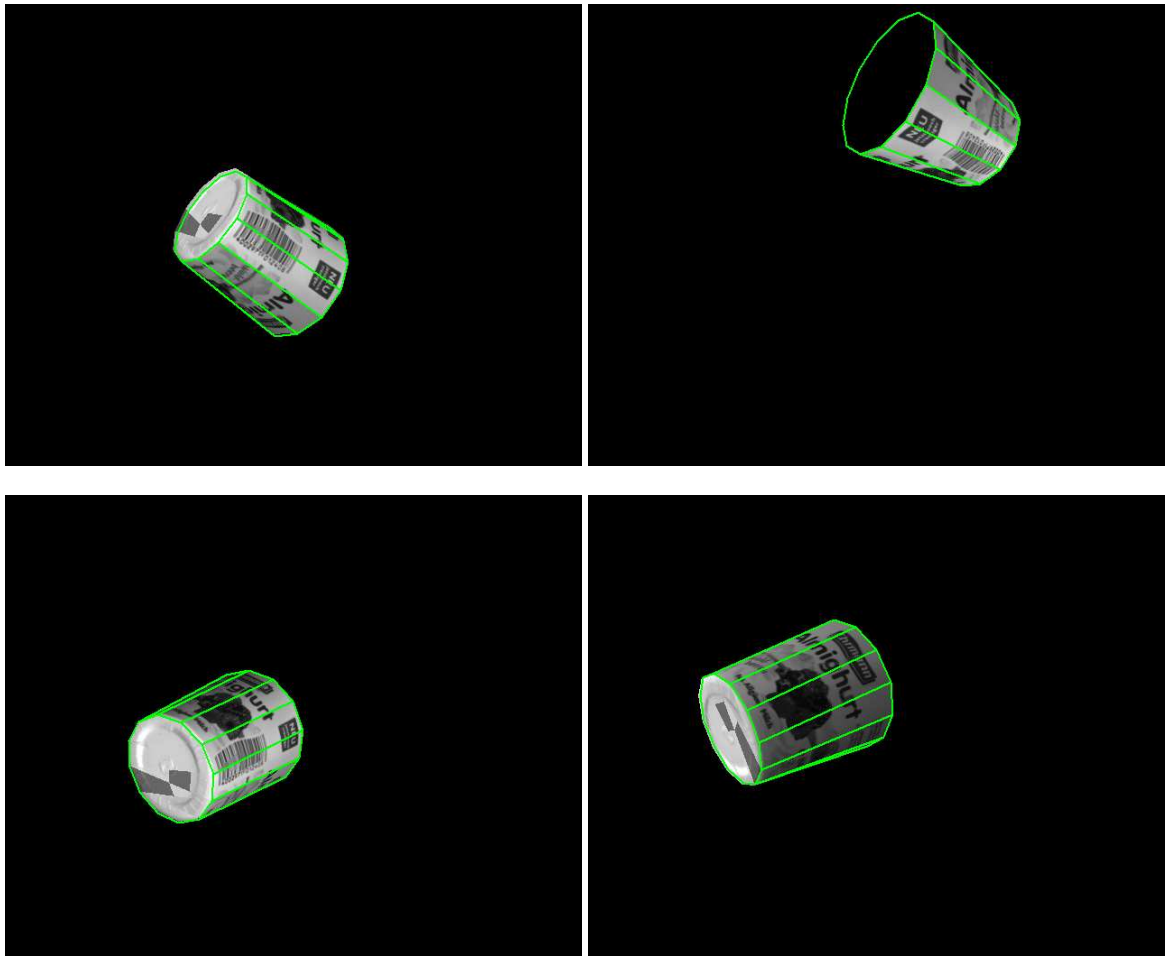


Figure 6.13.: Simulated views of a yogurt cup.

based CAD matching to get a good initial alignment of the images with the objects.

6.3.3. Conflict Resolution via Bundle Adjustment

From all training images, the interest points on the model can be extracted. For all those interest points in one image correspondences can appear in all other images, given the face the points are most probably on is visible in this image, too. The visibility is checked using a usual OpenGL rendering of the CAD model at the known position with the camera parameters of the real camera. Instead of a texture, the face id is rendered coded in the color. This gives for every pixel the most probable face given the initial pose estimate is good enough. Note that this might fail, if the faces get too small.

Interest Point Comparison using NCC

Two interest points can form a correspondence if they are on the same face in two different images. Additionally their NCC on the simulated front view of their face in both image around the interest point must be high enough. The NCC ε for two image f and t is defined as follows:

$$\varepsilon = \frac{1}{n-1} \sum_{x,y} \frac{(f(x,y) - \bar{f})(t(x,y) - \bar{t})}{\sigma_f \sigma_t}$$

with \bar{f} and \bar{t} being the mean intensity of the patches and σ_f and σ_t being the intra-variances of the patches.

The front view can be calculated using a perspective transformation warping the image part covered by the face to a quadratic patch of higher resolution. The area around the face is transformed also using the same transformation, in order to have a valid border treatment.

Calculating the Optimal Position

The optimization of the pose is performed with a Levenberg-Marquardt algorithm [78] on a the problem of finding the best positions for the objects in the cameras and the best position of the correspondences on the model.

All positions have 6 degrees of freedom and they depend on all correspondences that depend on one Harris point that was extracted in this image. The Harris points are considered as stable and are not recalculated immediately. The correspondences have two degrees of freedom each, and provide one error measurement per image point. This means that only correspondences with more than three image points introduce additional information and the minimal number of correspondences is at least 6 per image.

To evaluate the consistency of a correspondence regarding an underlying CAD model we need to reproject a point in an image to the Model to get its coordinates on the model. The position and orientation of the objects center is denoted with H_n . The faces of the CAD model defined by three points are denoted with $F, f_i \in F, f_i = [p_1, p_2, p_3]$. For any face f we can calculate the transformation from the model origin to a virtual coordinate frame having z pointing along the normal and having X oriented along the first edge $p_1 \rightarrow p_2$, which will be denoted as H_f .

Given an image point $[r, c]$ on the object visible at H_n the relative coordinates to the first point p_1 on the correct surface f_i is:

6. Learning Perceptual Models in Domestic Environments

$$t = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix} = H_f^{-1} P_0 + \frac{-(P_0 \circ p_1 p_2 \times p_1 p_3 + d)}{H_n \begin{pmatrix} fx \\ fy \\ f \end{pmatrix} \circ p_1 p_2 \times p_1 p_3} \quad (6.3)$$

The correct surface can be estimated by rendering the model at the specific position using the known camera parameters using instead of a texture the face index. Alternatively the point can be checked for a being inside a triangle, which is usually more costly if there are many points to test.

Formalizing the minimization which is performed using the correspondences C consisting of the structure $c \in C : [p_i, p_j, \dots, p_k]$ with a point of one image $p_i \in P_{n_1}$ and the other points $p_j \in P_{n_2}$ with $n_2 \neq n_1$ and $p_k \in P_{n_3}$ with $n_3 \neq n_2 \neq n_1$ in different images. for image t .

$$\varepsilon = \sum_{c \in C} \sum_{p_n \in c} \|KH_n H_{f_c} t_c - p_n\|_2 \quad (6.4)$$

While H_n is parameterized with six parameters, which corresponds to the number of degrees of freedom for a Transformation in space.

The resulting t_c give the positions of the interest points on the model and are not used later on. The transformations H_n give a more exact measurement were the object is located in the image. This information is exact enough to reconstruct the texture, given the CAD model is exact.

6.3.4. Reconstructing the Texture

For getting the final texture, we choose for every side of each face of the CAD model an optimal texture. This is decided calculating the NCC between all occurrences of a face in an image being visible. The one image patch with highest NCC multiplied with the minimal size of the two patches that are compared with the other patches is selected as the optimal texture. If there is more than one possible candidate with a similar score, the two images are fused.

The resulting images parts are stored as a texture and can be displayed using OpenGL or can be exported to formats like *.obj format for further usage.

6.4. Summary

In this chapter we presented methods that either can be used to model objects online or to use web resources to build a model that allows to localize and categorize objects. We also discussed the preconditions that have to be fulfilled, in order to be able to learn certain models, which are either given by the objects appearance or already available models which are required to build a more precise one.

Our perception system is capable to autonomously acquire new knowledge of objects and can so adapt online to new tasks. With the feedback mechanism described earlier in Section 2.5 it can even test on not fulfilled preconditions which cannot be measured on model creation like the uniqueness of a shape or a texture. Results for this capability can be found in Section 8.3.2.

6. *Learning Perceptual Models in Domestic Environments*

Part IV.

Applications of CoP

7. Perception Guided Robotic Manipulation

Most of the perception mechanisms which were discussed earlier in this work are meant to enable robotic manipulation. The most critical part in a perception-cognition-action loop, which we want to implement, is the actual action. This part is invasive to the world and it anyhow contains much more potential dangers and problems to the outside than perception or cognition. Any invasive action can be considered as potentially dangerous, since most unwanted effects are much more difficult to undo than to do.

Perceptual mechanism cannot influence any other robotic system besides taking cpu time and perhaps heating up the environment. On the other hand the manipulation mechanism can easily interfere with the perceptual mechanism by adding motion and additional clutter to the scene. This system requires a very careful design and high accuracy and predictable behavior.

7.1. Accuracy and Robotic Manipulation

Accuracy in the sensor data and the robot model is required for several obvious reasons. First, any sensor data must be accurately localized in respect to the robot, as soon as it should result in any action of the robot. Second, even if this localization is done, the accuracy should be persistent with any movement of the robot. And if it is not persistent, the implied inaccuracy should be trackable. Third, any internal world representation of the robot should be as close as possible to the reality.

In order to achieve the necessary accuracy, several calibration steps are required. Depending on the sensor set and the actuators of a robot, a calibration procedure can be achieved autonomously or not. E.g. a robot like the PR2 can see its actuators well enough to get an internal calibration of the cameras in the head and the arms. Meanwhile, it can be a problem to calibrate the base laser with the other sensors, since they have only a small overlapping field of view.

The calibration procedure we developed for TUM Rosie will be described here, which includes additional requirements for the SR4k calibration.

7. Perception Guided Robotic Manipulation

7.1.1. Camera Calibration

Camera calibration is necessary or at least very helpful to gather any kind of 2D-3D relation from images. This work just uses state of the art calibration methods in order to be able to use cameras in 3D manipulation setups. This section tries to give useful hints regarding potential problems for camera calibration, which is in robotics still an issue, even if a modern robotic platform like the PR2 are coming with a (nearly) complete auto calibration, including

7.1.2. Hand Eye Calibration

Fast and reliable manipulation requires a system with the minimum amount possible of systematic errors in the position estimations. In the real world there will be always uncertainty in sensing and acting, but it makes sense to minimize the error on the variables that are under control of the robot developer. This starts with the kinematic description of our robot. With this goal, we employ a method to calibrate all the kinematic chains involved: from the camera to the hand of the robot.

For the setup on TUM-Rosie, it is especially important to know the pose of the 3D sensors as exactly as possible. A direct hand-eye calibration is not feasible, since the measurements of e.g. a ToF camera are not reliable enough to get the necessary data in a good quality. An easier way is to use a RGB-camera mounted on the same pan-tilt unit. But this approach imposes another problem regarding the underactuated pan-tilt unit, while solving the issue of inaccuracy of the calibration. The task of calibrating the 3D position of a ToF using a camera is relatively simple. Calibration of the camera to the hand has also been investigated by other researchers in the past, but usually in a way that does not allow segmenting the actuated chain into parts.

Also, since the pan-tilt unit only has two degrees of freedom, it cannot produce all the data that is normally needed and we had to tackle this separately.

From Camera to Hand

We can describe the transformation hand (h) to object (o) $H_{h \rightarrow o}$ with the following transformation chain:

$$H_{h \rightarrow o} = H_{o \rightarrow c} H_{c \rightarrow pt} H_{pt \rightarrow ptb} H_{ptb \rightarrow ab} H_{ab \rightarrow h}$$

All H describe here transformations between the single parts of our system. The abbreviations besides h and o mean: c is the camera, pt the mounting of the head on top of the pan

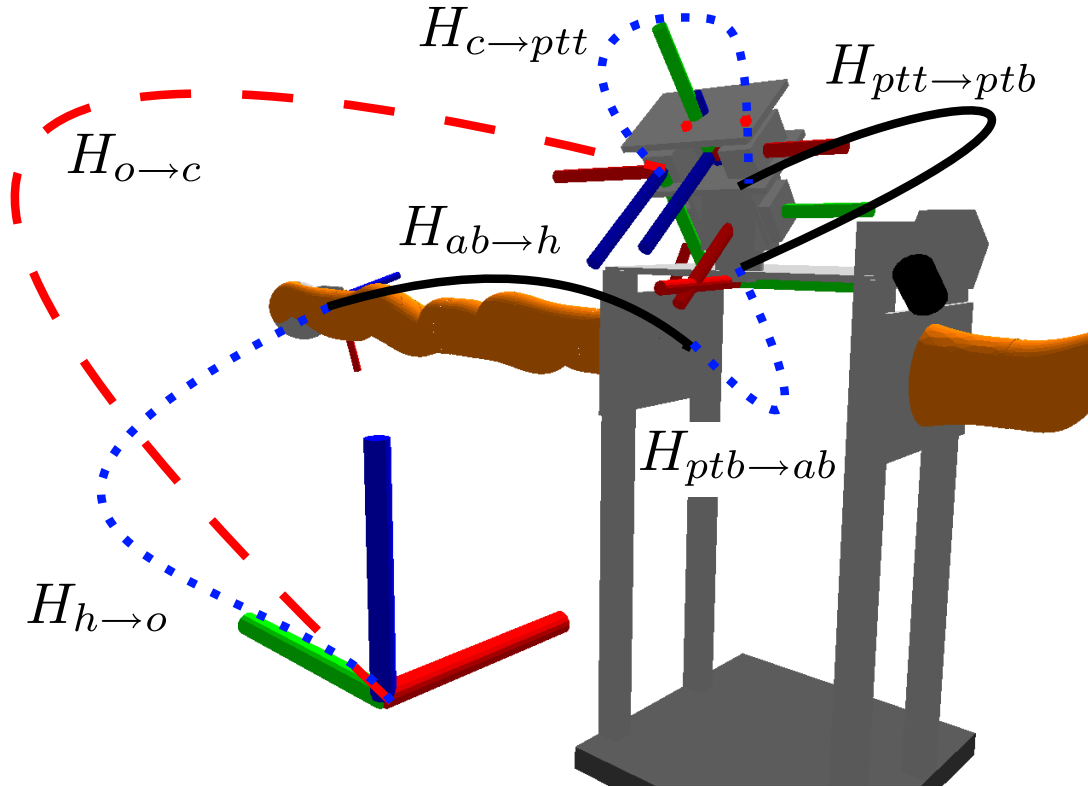


Figure 7.1.: A model of the robot including the frames involved in the calibration. Red dashed lines are perceived, the solid black lines are known and dotted blue lines are calibrated frames. The calibration object, represented by the bigger axes, was mounted on the flange of the arm, but it was drawn in front of the robot for clarity.

tilt unit, ptb the mounting of the pan tilt unit on the robot platform, ab the mounting of the arm on the platform. All of those frames are labeled in the model in Figure 7.1.

Of those transformations, some can be assumed to be known, given certain conditions. We can assume that we have an accurate forward kinematic for the arm as well as for the pan-tilt unit, which will give us $H_{ptt \rightarrow ptb}$ and $H_{ab \rightarrow h}$ depending on the current joint angle state. Inaccuracies in the mounting to the next part are considered in the respective next transformation.

Given we have a known object, namely a calibration body, we can also estimate very accurately the internal calibration of the camera and synchronously the relative position of the object to the camera ($H_{o \rightarrow c}$), given we have different views of the object. This leaves us with three unknown transformations $H_{c \rightarrow ptt}$, $H_{ptb \rightarrow ab}$ and $H_{h \rightarrow o}$.

If the robot's hand grasps the calibration body we can assume all those offsets to be static. This enables us to measure several joint transformations using state of the art hand eye calibration techniques. By moving the arm in the 6 dof of space and measuring the 6 dof of

7. Perception Guided Robotic Manipulation

the objects to camera we get 12 dof observations what enables us to estimate also two 6 dof offsets in the system.

By keeping the pan-tilt unit stable and moving the arm with the calibration body in space to K positions, we can solve while calibrating also for $H_1 = H_{c \rightarrow ptt} H_{ptt \rightarrow ptb} H_{ptb \rightarrow ab}$ and $H_2 = H_{h \rightarrow o}^{-1}$:

$$err = \arg \min_{H_1, H_2} \sum_{k=1}^K f(k, H_1, H_2);$$

$$f(k, H_1, H_2) = \sum_{n=0}^{|X|} \|x_n H_1 H_{ab \rightarrow h, k} H_2 H_{o \rightarrow c, k}^{-1} - x_n\|$$

where X is the set of known points on the calibration body. $H_{ab \rightarrow h, k}$ and $H_{o \rightarrow c, k}$ are the measurement of the forward kinematic and the calibration body localization after the k -th movement. If we use here a nonlinear optimization (e.g. the Levenberg-Marquardt algorithm), we come easily to the transformation H_2 which can be used to define a new virtual end effector: Given the calibration body is well formed and its we placed its frame at a position that is well placed inside of the robots hand we can directly use H_2 as the new end effector offset. H_1 contains still the pan tilt offset ($H_{ptt \rightarrow ptb}$) and two unknown offsets $H_{c \rightarrow ptt}$, $H_{ptb \rightarrow ab}$.

From Camera to Shoulder

If we now solve for those two remaining unknowns we could theoretically the same process with a stable arm and a moving pan tilt unit. Unfortunately, our pan tilt unit has both rotational axes in the same point which moves the camera on a sphere with an fixed radius. This means this process could result in a manifold of correct solutions while distributing one orientation and two offsets randomly between before and after the actuated elements. But in our case we also want to calibrate the ToF to the hand we have to solve this ambiguity, too. This means that a nonlinear optimization like the one applied for the arm is here not applicable for the full search for H'_1 and H'_2 which would be in this case $H'_1 = H_{c \rightarrow ptt}$ and $H'_2 = H_{ptb \rightarrow ab} H_2$ If we reduce the problem by inserting the already acquired information from the last step, we only have to solve H'_1

$$H'_2 = H_1 H'^{-1}_1 H^{-1}_{ptt \rightarrow ptb}$$

If we assume that we have a good approximation for the rotations of the camera on the pan tilt unit and we modeled it already in the transformation $H_{ptt \rightarrow ptb}$ we can optimize the translation of the camera on the pan tilt top x_0, y_0, z_0 :

$$err = \arg \min_{x_0, y_0, z_0} \|X_n^{ptb} - (X_n^k - (x_0, y_0, z_0, 1)^T) H_{ptt \rightarrow ptb}\|$$

7.2. Dealing with Uncertainty in a Robot System

where X_n^{ptb} is the n -th points on the calibration body in pan-tilt base coordinates while X_n^k is the same points in Camera coordinates after the k -th movement of the pan-tilt unit.

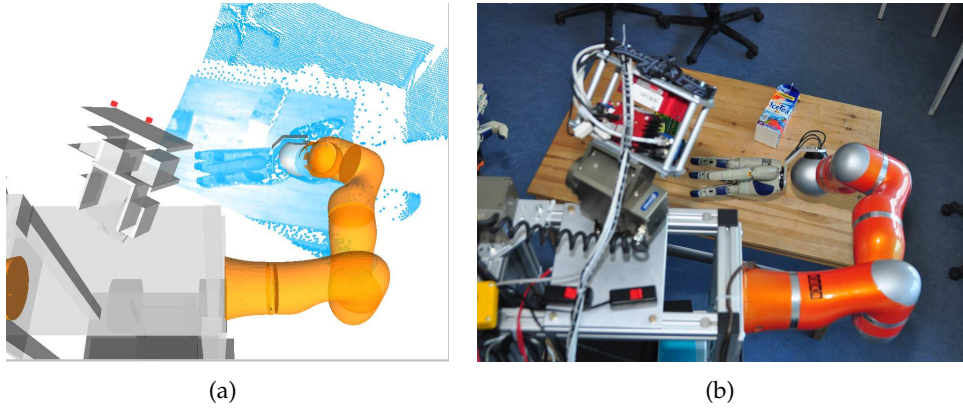


Figure 7.2.: Visual evaluation of the calibration: Overlay of ToF data and the robot model on the left. The model does not include the robot hand, but it is visible in the sensor data at the correct place. Please compare with the photo on the right of the same scene.

This leads to solution that ignores three rotational degrees of freedom, but given metal plate supporting our sensor head, those rotation are small and are compensated partially by the inverse error that is then contained in H_2' . It is eliminated completely in the pan-tilt position used for the initial calibration state with only the arm moving. With a good choice of this pan-tilt state in the center of the actual working space, this proposed simplification has the smallest error. The accuracy of the process is very well illustrated in the Figure 7.2 (left), where we depict the robot arm using the kinematic robot model, and the hand is seen in the 3D sensor data. Notice that the hand pose accurately reflects the picture on the right, which shows the correct calibration. All frames in figure 7.1 are involved.

7.2. Dealing with Uncertainty in a Robot System

A robotic system is even with a very good calibration not perfect. But for most of the systems there exists an estimation about the uncertainty which is introduced by this system. Self localization or calibration are good examples for procedures that are aware of the intrinsic residuals. We want to have a robotic system can approximate and model its own imperfections. So we want to introduce a system that monitors the positions of all parts of the robot and of all perceived elements with the respective uncertainties between all systems. In theory the propagation of errors is important for relevant control. The standard procedure was described in [13].

7.2.1. Covariance Propagation

An error in position space SE3 can be expressed by a covariance matrix describing an Gaussian approximation of the uncertainty of a position. This matrix has than 6 times 6 entries which describe the distribution in which the real position is much likely to be expected.

The uncertainty in position is quite easy to display and understand while the uncertainty in lacks good means to display it and so lacks also intuition to be understood. So we want here to give a simple example to explain the understanding and the visualization used in this work: Given an detection of an object which was done in a 2D RGB data has a very accurate localization of an object in the row-column plane of the camera, while it has only a rough idea about the distance of the object from the camera. This gives us an estimation of errors in the single axis which we can represent in a 3x3 covariance matrix C_{pos} with the errors in e_x, e_y (row-column) and e_z (distance):

$$C_{pos} = \begin{pmatrix} e_x & 0 & 0 \\ 0 & e_y & 0 \\ 0 & 0 & e_z \end{pmatrix} \quad (7.1)$$

If we only fill the diagonal we assume an independent error with a Gaussian distribution for each dimension. if we want to express this error estimate in terms of the robotic hand for example which might look in the same direction like the camera but has a different inclination which can be represented by a rotation around the cameras y-axis we would get an influence of the error in x and the error in z.

$$C_{hand} = \begin{pmatrix} e'_x & 0 & e'_{xz} \\ 0 & e_y & 0 \\ e'_{xz} & 0 & e'_z \end{pmatrix} \quad (7.2)$$

with a possible calculation of e'_{xz}, e'_x, e'_z given the rotation.

$$e'_{xz} = \frac{(\cos(\alpha)e_x \sin(\alpha)e_z) + (\sin(\alpha)e_x \cos(\alpha)e_z)}{2} \quad (7.3)$$

$$e'_x = \cos(\alpha)e_x \sin(\alpha)e_z \quad (7.4)$$

$$e'_z = \sin(\alpha)e_x \cos(\alpha)e_z \quad (7.5)$$

Which is just makes use of the simple case of the covariance transformation in 3D space which can be displayed by:

$$C_{pos} = \begin{pmatrix} c_1 & c_2 & c_3 \end{pmatrix} \quad (7.6)$$

7.2. Dealing with Uncertainty in a Robot System

$$C_{hand} = \sqrt{\sum_{i=0}^3 (H_{hand}^{pos} c_i) (H_{hand}^{pos} c_i)^T} \quad (7.7)$$

with H_{hand}^{pos} representing the transformation necessary to come from the coordinate system of the camera into the coordinate system of the hand.

This system allows to express positional error in another frame. If we extend the problem to errors in rotational space, we get a slightly different formulation of an error in six dimensions:

$$C = \begin{pmatrix} e_x & 0 & 0 & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 & 0 & 0 \\ 0 & 0 & e_z & 0 & 0 & 0 \\ 0 & 0 & 0 & e_\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & e_\beta & 0 \\ 0 & 0 & 0 & 0 & 0 & e_\gamma \end{pmatrix} \quad (7.8)$$

In order to transform this covariance by a full 6D pose to express the uncertainty in another frame, we need to decide on a valid representation of the rotations.

7.2.2. Located Object Tree

Expanding this problem to the representation of a perceived object, the error we want to calculate is usually in hand coordinates and should include a robotic movement before. This movement can only be measured under uncertainty, which also can be modeled. This implies that we have a chain of transformation from the object to the former robot position to the new position continued to the hand where we have at least two transformations which bear significant errors. To represent this, we need a tree of transformations from all located events to all others with the error which will be introduced to express the one in the other frame.

Located Objects

We consider the perceived object as a frame in space, which represents the orientation and position relative to the perceiving sensor at the position when the respective image was taken. This position represents an orientation and position of the respective sensor relative to the position of the robot at this point in time when the image was taken. Both of them have additionally a distribution stored which represents the Gaussian distribution. See 7.3.

7. Perception Guided Robotic Manipulation

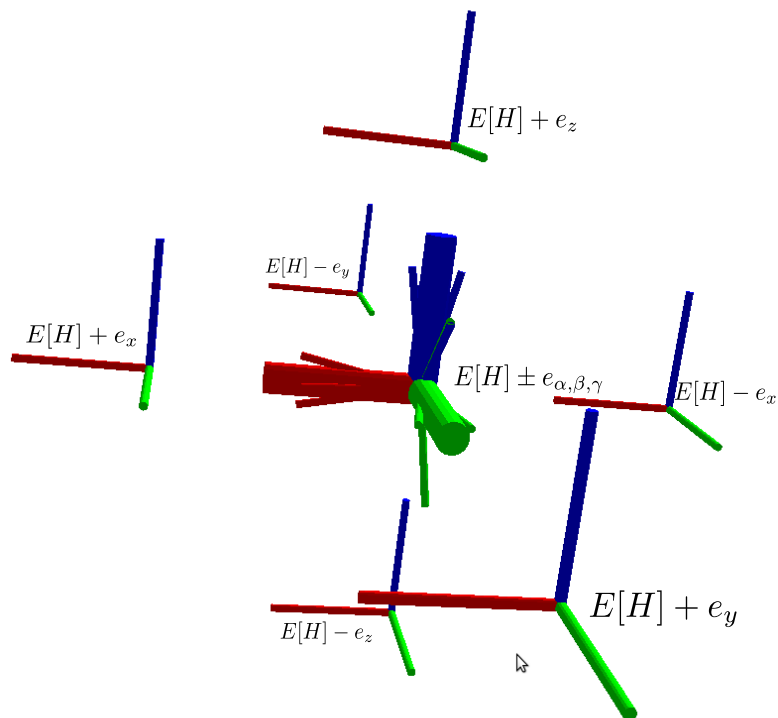


Figure 7.3.: A frame in space and the major components of an attached covariance displayed by the samples from the distribution that are displaying an approximation of possible samples of this distribution.

Event versus Time as Synchronization measure

In order to synchronize global position information over a system, there are different paradigms possible: The Located Object framework chose to use an event that changes a motor or measures a new position to be applied as soon as this event is reported to a central service. Alternatively, the information can be distributed, given a global information measurement, like synchronous time.

The moment the information arrives at the Located Object server decides on the time it is applied. This might be problematic in systems which have a high delay to communicate with the server. On the other hand it relaxes the requirement on synchronous clocks, like it is e.g. necessary for the usage of the `tf` framework¹.

Additionally, the service based approach reduces the communication requirements to a minimum, given the assumption that the measurement of movements are happening more often than the consumption of this information. This is usually true as soon as there is an abstraction from the real time control loops.

Interpolation Scheme

For getting positions that were measured in the past, both paradigms proposed in Section 7.2.2 have troubles. While the event based will lose all data which is not connected with an event, the time-based method can only store a certain period of the history.

For the event based scheme, the solution is to create an event for all relevant actions to avoid losing necessary information. This is especially interesting for the moment when a camera image is taken or similar events of sensor captures. With such an event created, even processing of sensor data which requires a long time (e.g. over 20s) will result in a result that is in a valid reference frame. This reference frame can be even used in a local context.

This mechanism assumes a constant interpolation in time, implying that no incoming events mean no movement (static world assumption). Alternatively, with a time-based synchronization, there will be a previous and a following measurement to any point in time in the near past, which can be used to interpolate the position at a certain point in time. This assumes the measurement of the position to be well derivable at any point in time. Usually, this assumption does not hold for direct and not filtered encode measurements. So this interpolation scheme has to be applied carefully, especially for non regular events.

¹<http://www.ros.org/wiki/tf>

7.3. Grasp Planning

Our goal is to build a general and competent control system for mobile manipulation robots performing pick-and-place tasks in human environments. We can expect to have information and detailed models of many of the objects present, and this should be employed to the greatest possible advantage; but in any open environment the robot will certainly encounter unknown objects, or due to sensor uncertainty, it might fail to recognize known ones.

To deal with these issues we are developing a pick-and-place control system that employs two general classes of grasping strategies: (1) informed grasping strategies that use models of the objects to plan or infer adequate grasping actions using the model as an information resource and (2) general methods that can grasp objects successfully without having model information and just relying on single view sensor data.

In this paper we investigate the second class of grasping strategies, the ones applicable without having prior models information. An overview of this grasping strategy is shown in Figure 7.4: The robot is ordered to grasp an object on a table, so it moves its time-of-flight camera and obtains a point-cloud of the table and the object. The object of interest is segmented from the point cloud, and a Gaussian point distribution representation is calculated (Sec. 4.2.1). This is the simplest model that can represent the position, size and orientation of the object including the perceptual uncertainty. We then use a simplified model (Fig. 7.6) of our robotic hand to find a good hand orientation and approach vector considering obstacles.

The grasp action consists of: (1) Moving the hand to the approach position/orientation while avoiding obstacles (2) Move to the grasp-pose while detecting collisions with the object and adjust the grasp accordingly. (3) Execute a 3-finger-pinch that holds the object using the robotic hand. (4) Detect problems (if the object slipped out of the hand). (5) Lift the object from the table.

The main contributions of this paper are: (1) A perception system based on time-of-flight range data that represents objects as Gaussian point distributions. (2) A grasp pose optimization algorithm. (3) A method to incorporate torque sensors in the fingers to detect collisions and improve grasping. A correct hand-eye (robot hand/cameras) calibration is an important pre-requisite, so we explain the used method in section 7.1.2. Briefly said, we present a working system that uses 3D perception with a time of flight (ToF) camera and torque sensors in the fingers to reliably manipulate a large set of unmodeled objects.

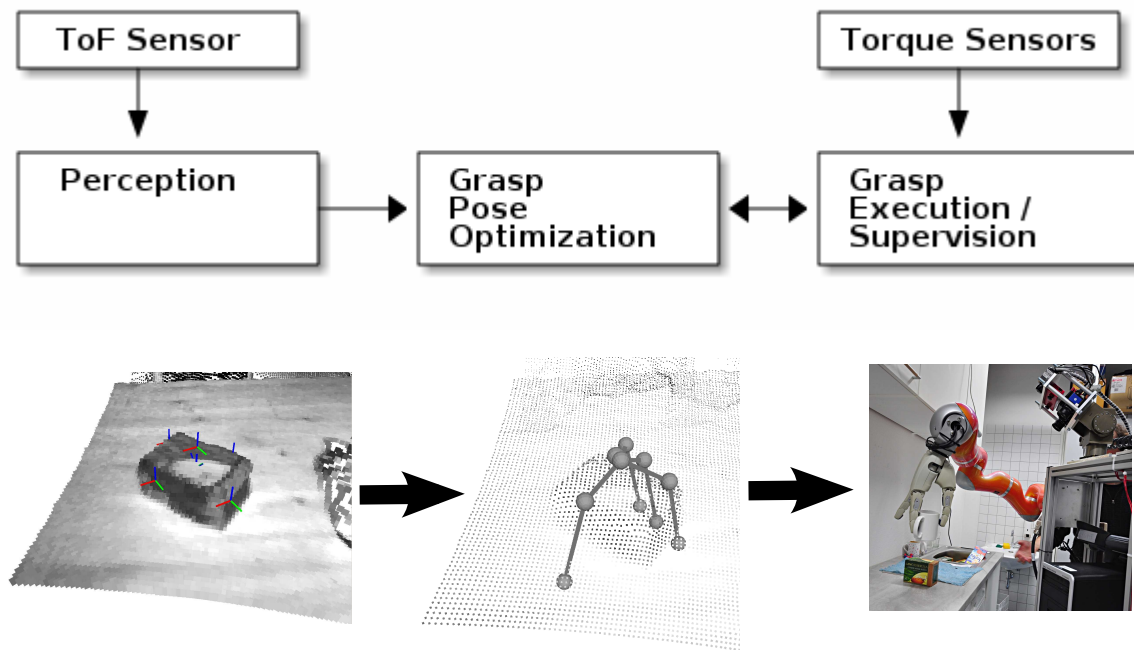


Figure 7.4.: Overview of the proposed system.

7.3.1. Related Work

The dexterous ARMAR-III platform [3] showed planned manipulation with a five-fingered hand for everyday objects in [77]. While this work still relies on objects being marked with colors or significant textures, other work extended the perception modalities (like the work with STAIR in [94]) to perception by touching for manipulation of unknown objects. Also demonstrated on the same platform were grasp point estimation for a bipedal manipulator and manipulation of unknown objects based on vision in [116].

An example for 3D sensor based object localization and manipulation was presented on a PR-2 in [113], which could also be done with our sensors. We currently also use the robotic middleware ROS of Willowgarage [102].

Natale et al [80] present a system that deals with great uncertainty in the position of the manipulator by using tactile feedback to explore the objects. This is inspiring work for incorporating sensors in the robot's hand. We calibrate our robot to be able to act quickly and accurately when the sensors allow it, and follow the same idea of exploration when the estimates of position and/or size were not precise. Another very interesting work shows manipulation of textured objects based on a single view [105]. A very advanced system was presented on a HRP-2 in [85]: it additionally includes the verification of a manipulation task by vision in daily life environments, this system also uses previously known objects and visual modalities and a simple hand, while it uses the full body for manipulation.

7. Perception Guided Robotic Manipulation

Saxena et al describe an advanced system to grasp novel objects using visual [117] and ToF [116] information. One important difference is that our system does not need training data and that it uses the torque sensors from the hand to monitor and improve the grasp.

A simple grasp planning was shown by Vahedi and Stappen presented in [135] for a three fingered hand could encage a polygonal region, and could approximate a grasp in a computationally complexity in $O(n^3)$ (n = length of Polygon). In contrast we present here a fast probabilistic encaging of an unmodeled object including a set of unmodeled obstacles.

The work of Geidenstam et al. [32] has shown grasp planning of previously unmodeled objects, which unfortunately requires on the one hand high quality input measurements and on the other hand relatively high computation time. Geometrical-simulation grasp planners like GraspIt [33] require object meshes or CAD models to work, or at least point clouds with approximated faces and normals. Both are not necessary for our approach.

A simple grasp planning was shown by Vahedi and Stappen presented in [135] for a three fingered hand could encage a polygonal region, and could approximate a grasp in a computationally complexity in $O(n^3)$ (n = length of Polygon). In contrast we present here a fast probabilistic encaging of an unmodeled object including a set of unmodeled obstacles.

The work of Geidenstam et al. [32] have shown grasp planning of previously unmodeled objects, which unfortunately requires on the one hand high quality input measurements and on the other hand relatively high computation time. Physical-simulation grasp planners like OpenRave [18] or GraspIt [33] require object meshes or CAD models to work, or at least point clouds with approximated faces and normals. Both are not necessary for our approach. For our experiments we calibrated our system using hand eye calibration. This technique was introduced by [131] and improved by [17] and solved for two unknown offsets in an robotic system. We show in this work a pragmatcal extension to calibrate another unknown component in our specific system.

7.3.2. Grasp Pose Optimization on a Gaussian Point Distribution

Given our simple representation of the position and shape of the object, our grasp planner has to find an appropriate pose where the robot can execute a general force-closure grasp, like a 3-finger-pinch.

Our strategy is to find a hand pose that brings the center of the palm as close as possible to the object, while avoiding collisions by maximizing the distance of the object to the points representing the fingers.

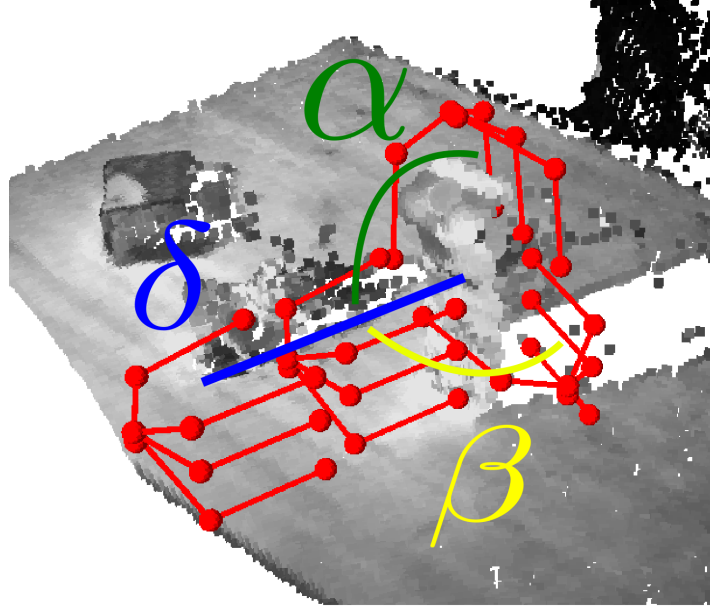


Figure 7.5.: We parameterize the grasp with three values: α influences the steepness of the grasp, β influences the direction of the grasp, δ defines the final distance between hand center and the object center.

These contradicting goals can be jointly solved using our representation of measurements of unknown objects as a point distribution: We model all objects as a Gaussian distribution of material around the estimated object center μ with a covariance Σ , see section 4.2.1.

We can approximate the probability of a collision of a finger tip position $P = [x, y, z]^T$ with the object at $Q = [\mu_x, \mu_y, \mu_z]^T$ by

$$d(P, Q) = [x - \mu_x, y - \mu_y, z - \mu_z]^T \quad (7.9)$$

$$f(P, Q, \Sigma) = \frac{1}{c\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}d(P, Q)\Sigma^{-1}d(P, Q)^T\right) \quad (7.10)$$

based on the standard Gaussian probability density function for joint random variables with $c = (2\pi)^{3/2}$. The random variables are here the three components of Q . Those components are not independent since we measure the end effector position in terms of the arm, while measuring the objects in terms of the camera.

Given this function f we can now evaluate a certain grasp and we can calculate an optimal grasp regarding this criterion. Figure 7.5 shows how we parameterize the position of the hand with the angle out of the x-y plane α (approaching angle, defined positive) and one angle around the z-axis β (hand rotation) we get the following function for calculating a Point $P_\delta^i = [x_i, y_i, z_i]^T$ on the approaching phase toward Q at distance δ :

7. Perception Guided Robotic Manipulation

$$P_{\delta}^i(\alpha, \beta, Q) = Q + \begin{bmatrix} (\delta - z)c\alpha s\beta + c\beta x_i - s\alpha s\beta y_i \\ (\delta - z)c\alpha c\beta + s\beta x_i + s\alpha c\beta y_i \\ (\delta - z)s\alpha + c\alpha y_i \end{bmatrix}, \quad (7.11)$$

where $c\alpha$ denotes $\cos(\alpha)$ and $s\alpha$ denotes $\sin(\alpha)$. If we introduce now a special distribution case of objects, that have a clean major axis along the robot's Z axis (equal to the table's normal), we get this special case for the covariance:

$$\Sigma_z = \begin{pmatrix} s_x & c & 0 \\ c & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \quad (7.12)$$

This special Σ allows a general minimization for an optimal α . Which can be expressed by

$$\arg \min_{\alpha} f(P_{\delta}(\alpha, \beta), Q, \Sigma) \quad (7.13)$$

This minimum can be estimated by set the derivative of f equal to zero, while setting $Q = [0, 0, 0]^T$ without loss of generality:

$$\frac{\Delta f(P_{\delta}(\alpha), [0, 0, 0]^T, \Sigma_z)}{\Delta \alpha} = 0, \quad (7.14)$$

which results in two solutions solving for α independent from all other parameters: $\alpha_1 = 0, \alpha_2 = \frac{\pi}{2}$. The result is intuitive: if an object is placed upright on a table, we only have to evaluate if we better grasp from the top or from the side. This does not hold for any inclined objects or any objects placed on a ramp, but it is valid for most household items and all our test objects.

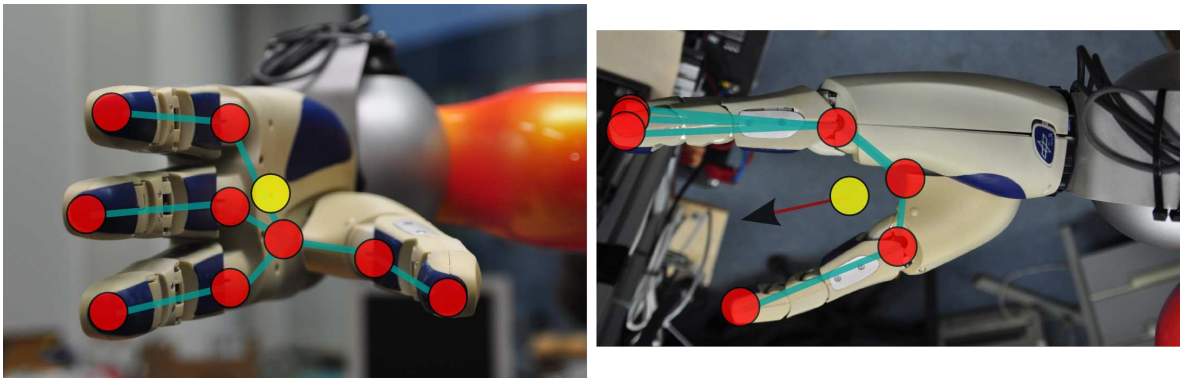


Figure 7.6.: Simplified hand model. Red circles indicate the points used in the kinematic model. The yellow circle shows the origin of the hand reference frame. The left hand is shown.

In order to characterize the hand properties we define a set of points $P_{\delta}^{0..9}$. They are shown

in Fig. 7.6. P_δ^0 and P_δ^1 describe the connection between thumb and the other fingers, which is a critical collision point. $P_\delta^{1..9}$ are the positions of the other fingers. All P_δ^i are just defined like P_δ with a constant offset in hand coordinates, which can be easily derived from Q , α and β . We defined the z-axis of the hand so that it starts at P_δ^0 , and it points towards the object at Q . This axis is controlled by the two parameters α and β while the palm is along the y-axis, which is normal to the plane described by the z-axis of the object and the hand. Basically, the hand-model is a kinematic tree representing the fingertips and internal points of the palm.

The simplified error function for a point under the given assumptions is then:

$$f_s = \min(f(P_\delta^i(0, \beta, Q), Q, \Sigma), f(P_\delta^i(\frac{\pi}{2}, \beta, Q), Q, \Sigma)) \quad (7.15)$$

Which leads to the best configuration for an object floating in free space:

$$[\beta, \alpha] = \arg \min_{\beta, \alpha} \sum_{i=0}^9 f_s(i, \delta, \beta, Q, \Sigma) \quad (7.16)$$

where β is, due to the symmetry of our point distribution, a value between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ and $\alpha = 0$ or $\frac{\pi}{2}$. The choice of δ which is a value between $\min(s_x, s_y, s_z)$ and $(\max(s_x, s_y, s_z) + \text{fingerlength})$ not only depends on this collision probability, but also on the enclosure probability, which is inversely correlated with δ . Then it requires an additional optimization for this parameter.

Incorporating the table leads to further reduction of the parameter space in α and δ for small objects:

$$\begin{aligned} \mu_z - z_{table} < \frac{\text{palmwidth}}{2} &\Rightarrow \alpha = \frac{\pi}{2} \\ &\Rightarrow \delta > z_{table} + \text{fingerlength} \end{aligned} \quad (7.17)$$

Further objects on the table at position $Q_j = [\mu_x^j, \mu_y^j, \mu_z^j]$ must be considered if they conflict with the arm trajectory. This only is the case, if the two following conditions hold:

$$\mu_z - \mu_z^j < s_z^j \quad (7.18)$$

$$\sqrt{(\mu_x - \mu_x^j)^2 + (\mu_y - \mu_y^j)^2} > \text{fingerlength} \quad (7.19)$$

All relevant obstacles are collected in $\mathcal{O} = [Q, Q_1, Q_2..Q_N]$ and $\mathcal{S} = [\Sigma, \Sigma_1, \Sigma_2.. \Sigma_N]$. Both variables contain as a first element the object to grasp, and subsequently all detected obstacles. Eq. 7.15 can be generalized for an obstacle:

$$f_s^j = \min(f(P_\delta^i(0, Q), Q_j \Sigma_j), f(P_\delta^i(\frac{\pi}{2}, Q), Q_j, \Sigma_j)) \quad (7.20)$$

7. Perception Guided Robotic Manipulation

This adaption can be incorporated into Eq. 7.16, which now can calculate the desired grasp configuration in our setup in a few iterations over δ and β . The optimization process should best start at a β corresponding to the smallest extension of the object and at the smallest possible distance δ , searching for the first local minimum.

This adaption of Eq. 7.16 follows:

$$[\beta, \alpha] = \underset{\beta, \alpha}{\operatorname{arg\,min}} \sum_{j=0}^N \sum_{i=0}^9 f_s^j(i, \delta, \beta, Q, Q_j, \Sigma_j) - a_\delta \quad (7.21)$$

where N denotes the number of obstacles which can be adapted during the grasp in case of unexpected collisions of the hand with the target object. a_δ is a factor charging for a further delta depending on the number of finger points and δ . The added Q_j would be the finger position at the time of collision and Σ_j will be the uncertainty of the hand obstacle relation at this time, which we assumed constant. If a collision is detected during the execution of the approach and grasp movement, the hand is moved back and a new calculation of the optimal grasping pose is done.

To conclude this section, we have shown a simple error measurement that allows searching on only three variables for a good grasping pose taking into account possible obstacles during the approach to the grasp position.

7.3.3. Detection of Collisions in the Fingers

As was seen at the end of Section 4.2.1, it is possible that the perception system underestimates the size of the object of interest due to occlusion, or measures the object's position inaccurately due to sensor uncertainty. We summarize here the system designed and developed for [69] for the sake of completeness, while this part was primarily done by Alexis Maldonado.

Since we suffer from self-occlusions, we will have to execute a partially 'blind' grasp. Executing a such a 'blind' or sensor-less grasp procedure will not be reliable due to the important effect caused by small errors in the pose estimation or calibration.

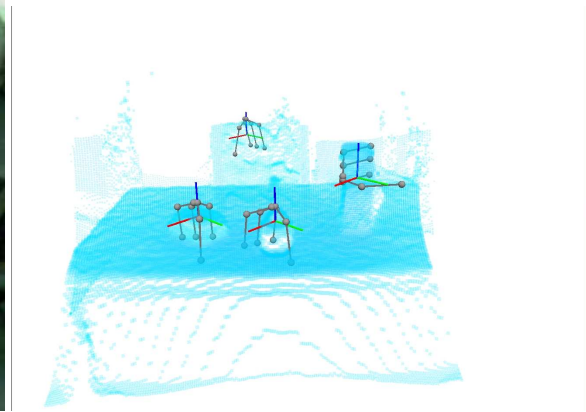
The best place to get information about the object being manipulated are the fingers themselves since they are the closest and most reliable source of contact information. With any kind of sensors integrated to the gripper or the fingers we can decrease the grasping risks by a lot.

E.g. the DLR-HIT hand of TUM-Rosieis equipped with three torque sensors on each finger, placed near each joint. Our hand controller reads torque data constantly from the fingers, and uses it to detect collisions with the object during the approach phase. A similar approach

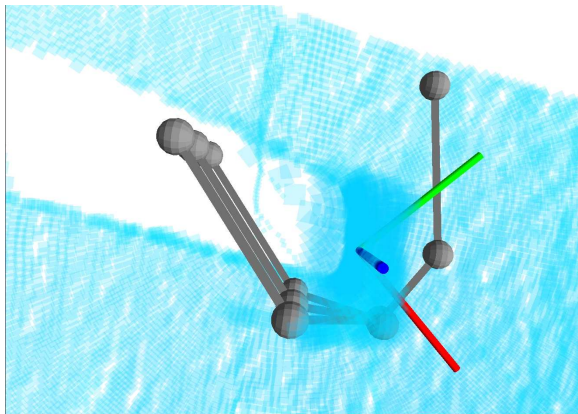
7.3. Grasp Planning



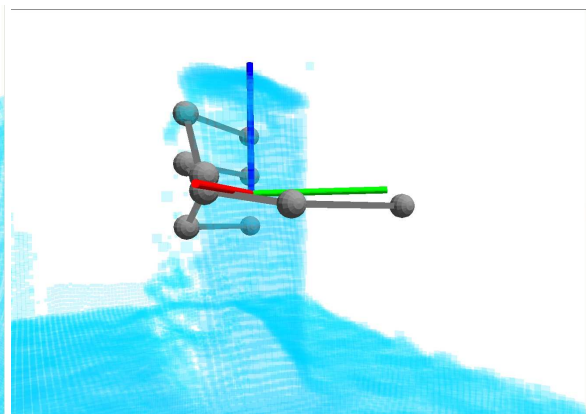
(a) A scene seen by the robot's RGB camera.



(b) Scene data from the ToF camera. Calculated grasp poses are shown.



(c) Resulting grasp position on the iced tea box (top view).



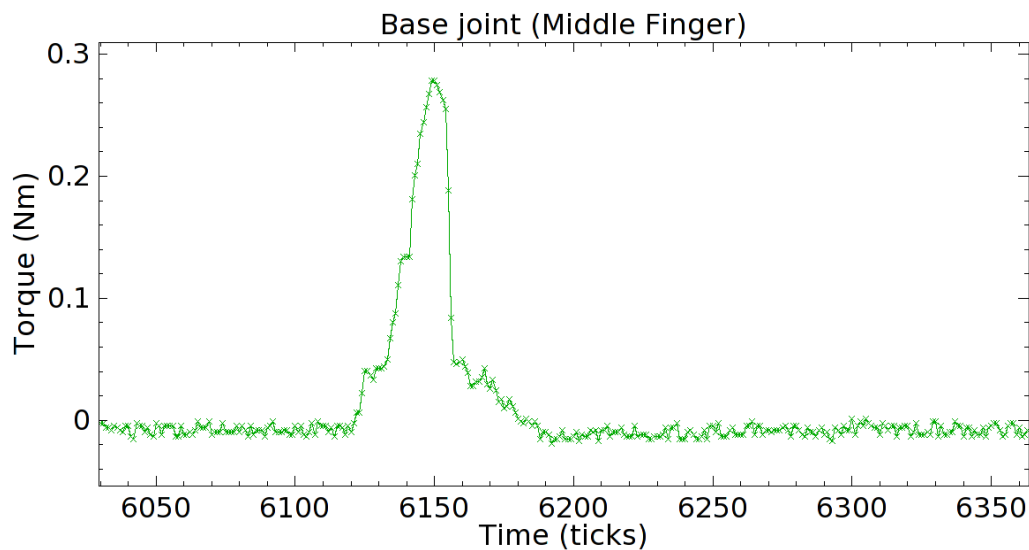
(d) Resulting grasp position on the iced tea box (side view).

Figure 7.7.: Visualization of selected grasps in a scene.

7. Perception Guided Robotic Manipulation



(a) Approach, collision, pull-back, grasping using the corrected pose.



(b) Base torque data for the middle finger.

Figure 7.8.: The collision with the object can clearly be detected based on the torque data. The finger was in contact with the object for 0.3 seconds, [69].

is used at TUM-James, which is equipped with a touch sensor array at each gripper side. In case of TUM-Rosie, a collision can simply be detected by applying a threshold on the filtered torque streams, as can be seen in Fig. 7.8(b).

Sensing and positioning of the arm will always have uncertainties, so we employ a haptic approach to make the grasping more robust: A detected collision event on any finger indicates that our estimation of the object's shape and/or position was wrong, but we can use the position of the collision to improve the estimation, so the system can locally correct the grasp. See Eq. 7.21 where the results are taken into account. Basically, the hand is moved back, the position where the crash was detected is put back into the grasp planner as a possible obstacle, and a new grasp pose can be calculated. The system also checks if the new desired position is reachable by the hand using a position-inverse-kinematics algorithm, and can look for other grasp positions until it finds an appropriate one.

Since this algorithm is designed to put the fingers around the object as closely as possible,

all that is needed is a simple enveloping grasp. We chose a 3-finger pinch on our DLR-HIT hands, but the same algorithm would work with a parallel-finger manipulator (with torque sensors installed to detect crashes on the fingers).

7.3.4. Motion Control

In order to guide the motion of the arm and the hand for manipulation, we employ a system based on vector fields [10], where it is possible to create point attractors, as well as planar and spherical repellers. We can easily encode the available information from the environment: the desired pose of the end effector, and all the detected obstacles.

The KUKA LWR-4 arms are being operated in Joint-Impedance mode, with a low stiffness set for each joint for safety. The arm could be pushed by a person or have a collision at any time, and the motion controller should deal with this gracefully and continue toward the desired pose while avoiding obstacles. For this reason, we do not pre-calculate any trajectories.

Our current implementation has an on-the-fly reconfigurable vector field that decides the direction in which the end effector of the arm should move, and it feeds this to a damped least squares inverse kinematics algorithm for finding the joint velocities. This loop runs in real time (currently 240Hz). The vector field also observes additional constraints, like the distance of each joint to its limits, or the distance of the whole arm posture to a preferred one, and influences the inverse kinematics algorithm by changing constantly the joint-weights (affect the relative movement of the joints), or task-weights (affects how much importance is given to position or orientation in Cartesian space).

For the experiments described in this paper, we used a relatively simple high-level controller that: (1) moves the hand and the arm out of the view of the sensors in the head. (2) asks the visual perception system for positions of objects and obstacles. (3) calculates approach and grasp poses using the grasp pose optimizer. (4) configures the vector field accordingly, moves the arm, deals with collisions. (5) executes a 3-finger pinch on the hand. (6) lifts the object, and evaluates the grasp. (7) drops the object at a pre-defined location.

Errors can be detected during each step of the procedure, and the controller takes corrective action, e.g. moving the hand out of the way, and detecting objects again if the grasp was not successful.

7. *Perception Guided Robotic Manipulation*

8. Results

For showing the performance of COP, we evaluate first the most important methods on their applications to object localization and recognition in the following Section 8.1. Since the grasping system is an important precondition for understanding the system evaluation, we will show results of grasping detected clusters in Section 8.2. Finally, experiments will be presented to evaluate the learning of models, and the integration of model learning into robotic actions in Section 8.3.

8.1. Object Localization

In order to show the capabilities of COP in object localization and recognition, we will have a closer look at four different methods for object localization. First, an analysis will evaluate the performance and robustness of the planar shape matching with stereo cameras. Second, we will analyze how the CAD matching combined with the search space reduction performs. Third, we will have a short look at the descriptor based matching for 3D objects. And last, we will show the transparent object detection and reconstruction. This last task is the most challenging regarding the data quality.

8.1.1. Planar Matching

First, the planar shape matching is analyzed, which is explained in Section 5.1.3. This technique requires the fewest knowledge about the scene compared to the following. The relevant aspects to focus on, are accuracy and generality as well as the applicability to robotic, which is not intuitive, since the assumption of planarity seems not to hold often. This assumption introduces only small errors for compact objects, which makes it applicable for the average object which is considered here.

In the second experiment regarding robustness, we explore online learning and tracking of everyday objects on a kitchen service robot that is equipped with a pan-tilt unit. In the third experiment, manipulation using an industrial robot arm is shown. It is based on a monocular camera setup and shows 3D pick-and-place.

8. Results

Method	Norm 3D error	Rotation error
Match shape left, pose	0.0041	0.22°
Match shape right, pose	0.0042	0.20°
Left and right, best of two	0.0038	0.19°
Left and right, triangulated	0.0053	0.24°
Min. ground truth error	0.0013	0.11°

Table 8.1.: Measurement errors with ground truth for the stereo setup.

Accuracy

To test the accuracy, we used an additional stereo setup with a more controlled setup than we usually can provide on a robot. We are particularly interested in the stereo setup, because the errors from stereo calibration, model generation and detection accumulate. Table 8.1 shows the results of different test runs. Here stereo-based initialization is compared to the result of a marker-based approach. For this experiment we calibrated the stereo setup that had a baseline of 6.37 cm, the object was on average 26 cm far away. We used an image sequence of 150 frames.

We compare our pose estimation with a state of the art camera calibration system. We build up a model using the stereo setup and infer the position in space first from the left image (first line), and then estimating this pose again from the right image (second line). Then the better match from both images is taken to estimate a final pose (third line). The selection of the better pose is based on the matching score. Alternatively, two 2D-homographies can be taken to triangulate the 3D position (fourth line). This method turned out to be less accurate than the averaged monocular estimation.

For ground truth we use the estimated marker's positions as it is known that they can be extracted with a very high accuracy. However, they are still erroneous, so we cross check the markers position in the two stereo images for consistency. The inconsistency for the marker based approach can be found in the last line of Table 8.1 named minimal ground truth error. It describes the minimal error, which is very close to the true error value, but gives only a lower bound, since it is the observed inconsistency regarding the position estimates of the markers.

The first column describes the normalized translation error which is defined as translational error divided by the camera-object distance. The rotation error in the second column is based on the length of the axis-angle representation of the rotation, or stated otherwise the angle that is needed to bring the ground truth rotation to the measured rotation.

The result is that online usage of two cameras does increase the accuracy and the robustness,

while the usage of the stereo calibration directly to triangulate the position does not. This proposes, that a calibrated monocular setup can replace often a stereo setup, if there is at least a consistency check with a second camera. This conclusion is an argument in favor the independent treatment of different cameras in an architectures like COP. The error in the best method matching the object independently in both images was as expected not much higher than twice the ground truth error imposed by the marker-based approach.

Generality

To show interactive usage of the approach, a robot-mounted pan-tilt unit follows an object under 3D motion. The model is learned as soon as the robot sees a new object inserted into a predefined ROI. Next, the shape model is learned from this ROI and tracked directly. The pan-tilt unit was directed to the object after every detection to avoid loosing track.

Fig. 8.1 shows 8 objects to give an impression of the variety of objects that the approach can track. Half of them are planar the other have only planar substructures. For these and 16 further objects we measured an average time for searching of 1.277s (full rotation and scale range of 0.5 – 1.5) and for the tracking update step an average time of 0.076s. For this experiment we tracked each object for about 60 seconds. The images have a size of 695x519 and we are using one processor of a AMD Athlon(tm) 64 X2 Dual Core Processor 3800+. All objects were tracked robustly, except the spaghetti (Fig. 8.1(k)) and the napkin (Fig. 8.1(m)) caused problems due to their structural deformations, that are not modeled. Still, even those objects could be found and tracked for a while. This test was done with a stereo on a robot, which is similar to the stereo cameras mounted on TUM-Rosie, only using lower resolution cameras and less wide-angle lens setup.

Monocular Object Detection for 3D Pick-and-Place Applications

The experiment described in the following was executed in cooperation with Andreas Hofhauser. Commercial solutions for pick-and-place applications typically assume that the objects are located on a plane like a conveyor belt and the camera is fixed exactly perpendicular, such that object detection reduces to a 2D search problem. These systems fail if the object of interest is surrounded by clutter or is inside a complicated 3D object configuration that leads to an unpredictable 3D displacement of the object.

In our test scenario we attached a calibrated camera and a strainer to an industrial robot arm. We performed a hand-eye calibration and a calibration of the robot flange to the strainer. Then we acquired a contour model that contains the outer structure and parts of the label on the clamp. After model generation we measured the 3D size of the model by acquiring

8. Results

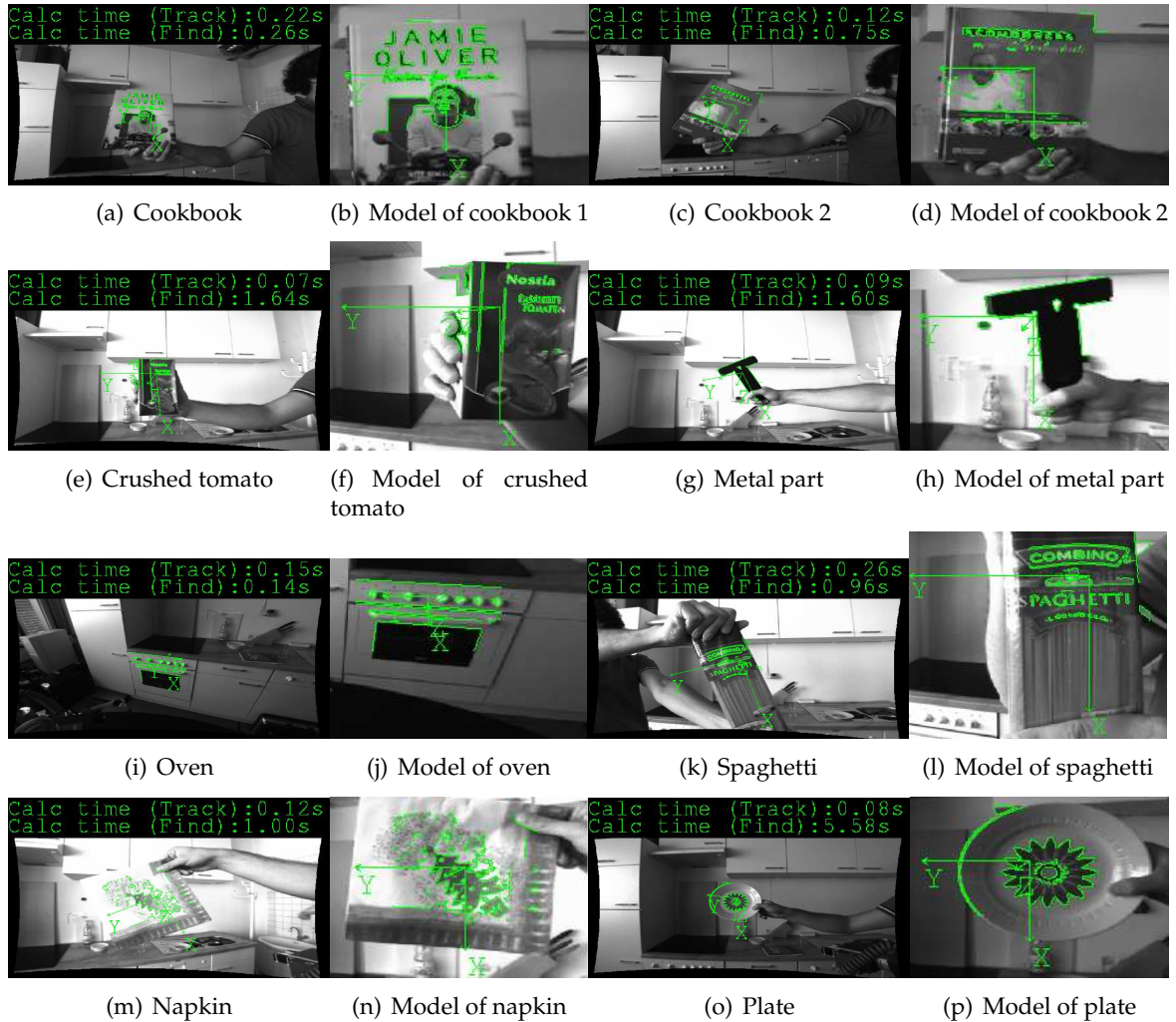


Figure 8.1.: A Selection of trackable objects. The objects 8.1(a),8.1(c),8.1(e) and 8.1(g) are planar, the other four (8.1(i), 8.1(k), 8.1(m), 8.1(o)) have only planar substructures.

two image: An image with a calibration plate on the fuse and another one without while the camera object relation was kept fixed. To have a sufficient field of view the robot drives to a surveying position from which a single image is grabbed and processed with the proposed algorithm. After a successful estimation of the 3D pose of the model, we convert the pose into driving commands for the robot gripper. After successful gripping the object can be further processed. For the grasping an accuracy of less than 5mm is typically needed, resulting in less than 0.01 normalized translational error (see Section 8.1.1 for a definition of the errors).

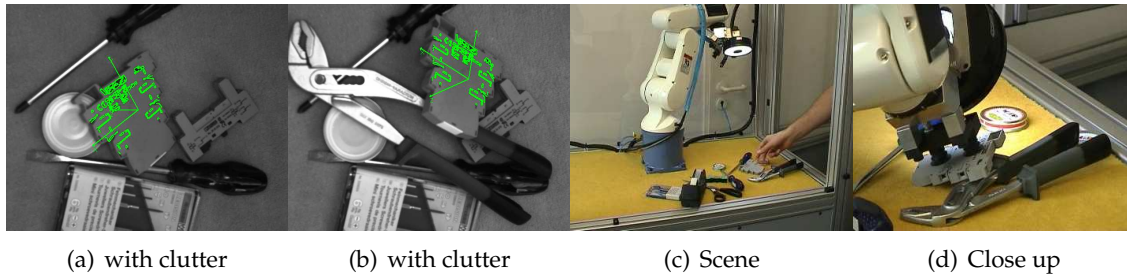


Figure 8.2.: Picture from Pick-and-Place Scenario. The complete setup can be seen in the accompanying video. The found model and the estimated 3D pose are depicted in typical search images.

8.1.2. CAD Based Matching

For the sake of completeness, we briefly review some of the results which were published in [57] in order to introduce the improvements which were introduced in [56]. The major problems that we faced using shape based monocular CAD matching, concern the missing robustness against clutter and the high calculation time of the model generation phase. The following sections demonstrate significant improvements in terms of computation time and detection robustness in cluttered scenes, by embedding the CAD matching into COP. The observed improvements are caused by the automatically refined search spaces and verifications using a stereo setup described in Section 5.1.2.

Computational Efficiency

To demonstrate the computational advantages obtained by incorporating the 3D clustering information into search spaces, we performed several tests for different objects located in various cluttered scenes. In each experiment, we recorded the calculation time that was necessary to generate the model and the to perform a search in the scene. Since the four tested objects have a large variety in the number of faces in their respective CAD models, the overall calculation times can vary significantly. The four objects were chosen to represent the usual distribution over the number of faces for CAD models for kitchen objects.

The experiments include a comparison between the results obtained using the entire search space of the full table versus the one automatically generated by the clustering. Figure 8.5(a) shows the resultant calculation times for the model generation phase. As presented, the highest relative gain in calculation time is obtained when a number of faces in the object is high as well. Additionally, the relative size of the cluster with respect to the initial search space influences the calculation time. The smaller the segmented clusters are, the higher is the gain. This can be seen by comparing the calculation times for large and small clusters.

8. Results

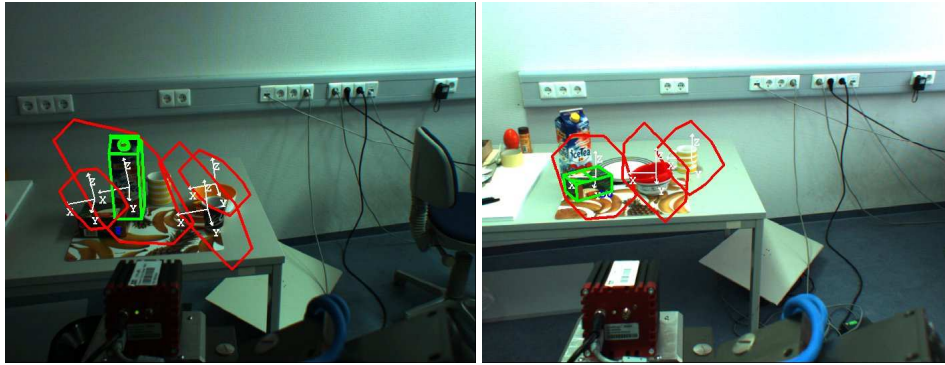


Figure 8.3.: Successful located objects in cluttered scenes. THE CAD models of an ice tea box and a tea box are overlaid in the camera image.

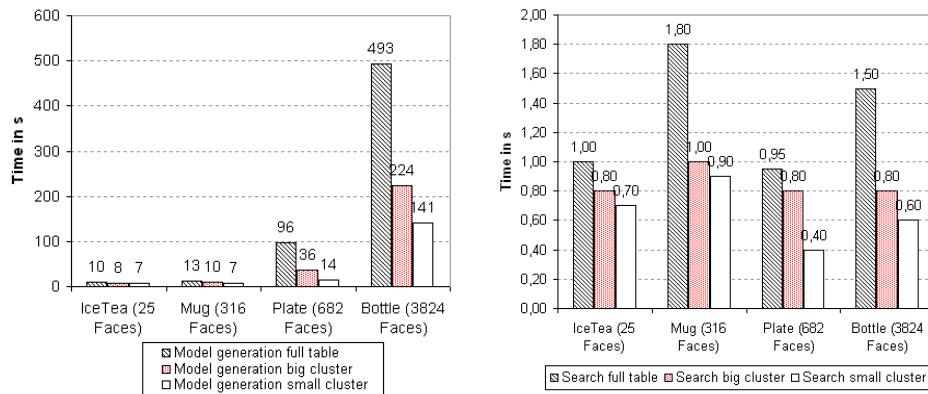


Figure 8.4.: Successful fitting object models in cluttered scenes. Several mugs are matched here.

An example of a large cluster is an ice tea box, while a small cluster can be caused by an appearance of the mug, as seen for example in Figure 8.6.

The time needed for the clustering added up to $\approx 50ms$ per frame and is omitted in Figures 8.5(a) and 8.5(b). The model generation can be skipped for new search spaces if there was another model generated before, that contains the current search space completely. Figure 8.5(b) shows the calculation times for the search phase. Again, the graph compares the calculation time with and without the segmentation. This experiment included the assumption of the object standing upright on the table having an arbitrary orientation. All measurements were done on a AMD Athlon(tm) 64 X2 Dual Core Processor 3800+, which constitutes one of the two stock computers of our mobile manipulation platform.

8.1. Object Localization



(a) The bars show the calculation time needed for **generating the model** for a given search space for several objects with various number of faces.

(b) The bars show the calculation time needed for **searching** in the search region for an object.

Figure 8.5.: The gained speed up by using clusters instead of the full table (about $30000cm^3$). For a better overview of the effects of our approach, we divided the clusters into big ($\geq 750cm^3$) and small ($< 750cm^3$) clusters.

Robustness through Search Space Segmentation

To show that this approach increases the robustness of the system we compared the search of several objects in more complex scenes.



Figure 8.6.: An example of a single model search in a cluttered scene, using the entire search space (left), and using several regions of interest from the 3D processing module (right). Notice that the multiple RoI approach is robust against false positives, while in the full search space, the model of the mug is matched incorrectly.

An example is shown in Figure 8.6, in which the robustness of the localization framework was greatly improved by using the resultant 3D clusters from the point cloud data. In contrast, the approach using a full search over the entire space can return erroneous results, due

8. Results

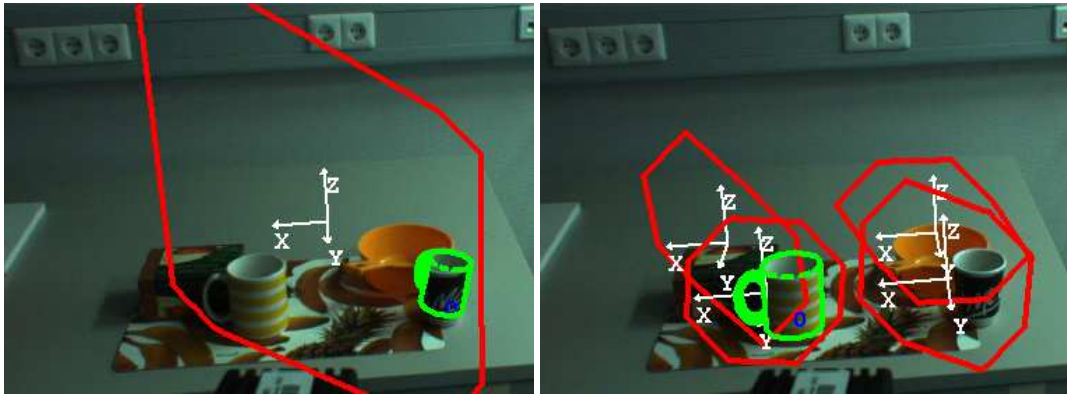


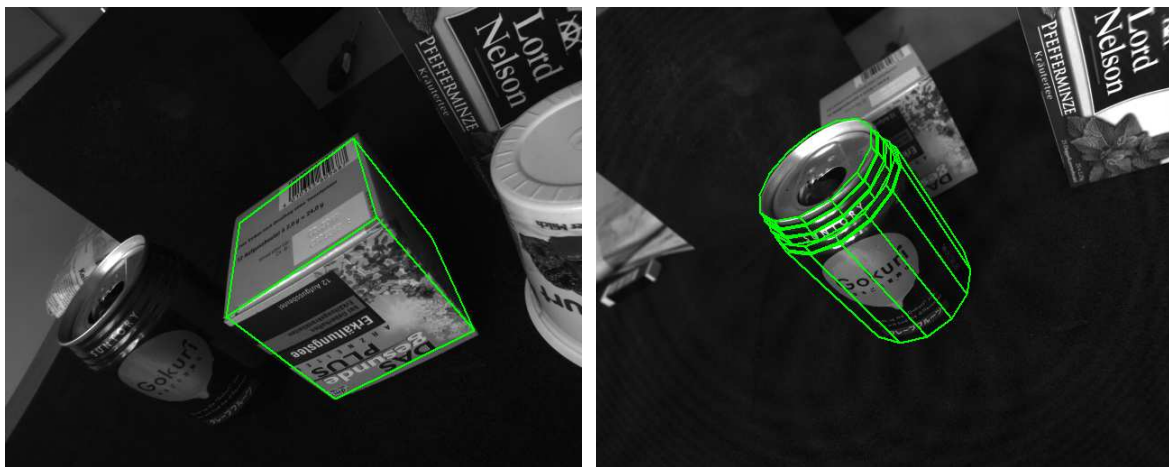
Figure 8.7.: A mismatch example with a wrong distance in the full search space (left), solved by using the multiple RoI approach(right). The model to be localized is represented by a mug.

	Tea	IceTea	Mug	Plate	All
Single Object	0.5	1.0	1.0	0.75	0.88
–with segmentation	0.75	1.0.	1.0	0.75	0.88
Partial Occlusions	1.0	0.75	1.0	1.0	0.94
–with segmentation	1.0	0.75	0.75	1.0	0.88
Cluttered Scene	0.2	0.75	0.6	0.2	0.44
–with segmentation	0.6	0.6	0.75	0.75	0.68
Overall	0.54	0.66	0.76	0.75	0.72
–with segmentation	0.76	0.84	0.84	0.83	0.82

Table 8.2.: The results for tests on 37 different scenes, including both approaches: with and without segmentation. The numbers are a ratio of good detected objects against present objects.

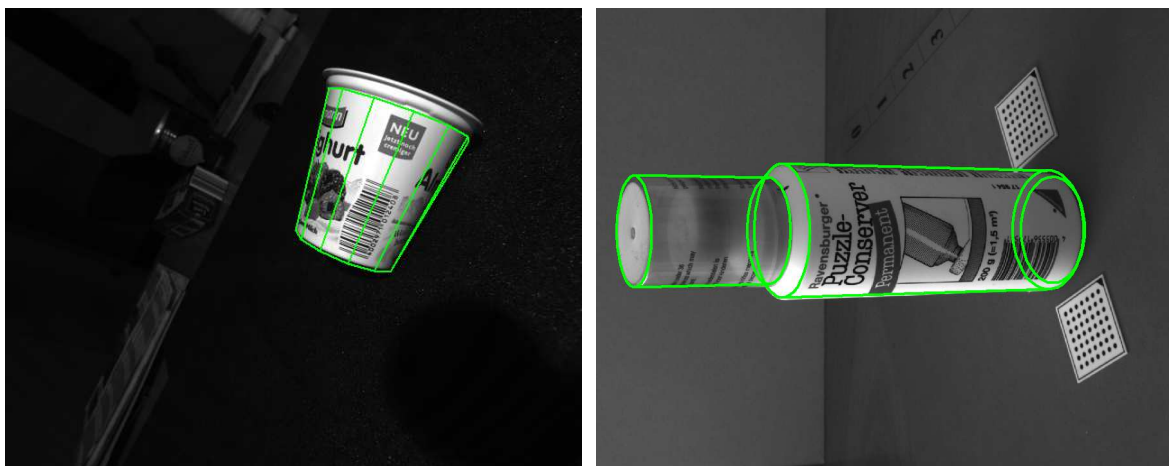
to problems with false positive matches. This can be observed particularly in scenes that contain similar or highly textured objects. Those views can easily produce hallucinations of degenerated views of an object. By a degenerated view we refer to situations that several projected 3D edges fall on the same 2D line. A similar problem can be seen in Figure 8.7, showing another object’s shape matches the projection of a mug model. These problems are overcome by segmenting the search space, which leads to a higher number of search spaces that have to be performed. This can result in case of many new search spaces in an increased computational time, but it allows a higher degree of parallelization.

We compared the earlier approach from [57] and the new pre-segmentation in three steps: i) first we let the algorithm run on simple scenes containing only one of the four object we were searching for, ii) then two objects that are partially occluding each other and iii) finally we built the scenes of ten and more objects and queried for all four objects.



(a) A box.

(b) A can.



(c) A cup.

(d) A bottle.

Figure 8.8.: The test objects for the descriptor based matching.

Table 8.2 shows the final results we obtained. For the simple scenes with one or two objects we got comparable robustness for both approaches, but when looking at the densely clustered scene the added value of the segmentation in 3D is significant and boosts up the robustness of the matching. The most interesting fact is the improvement we achieved on cluttered scenes. The ratio of well detected objects to all present objects was increased by more than 20%.

8.1.3. Descriptor Based Matching

We show here a short evaluation of the descriptor based matching. This approach is tested the following way: 10 - 16 images of the object are taken and are combined with a known

8. Results



Figure 8.9.: Virtual views of three of the test objects.



Figure 8.10.: Two objects matched based on descriptor points in cluttered scenes.

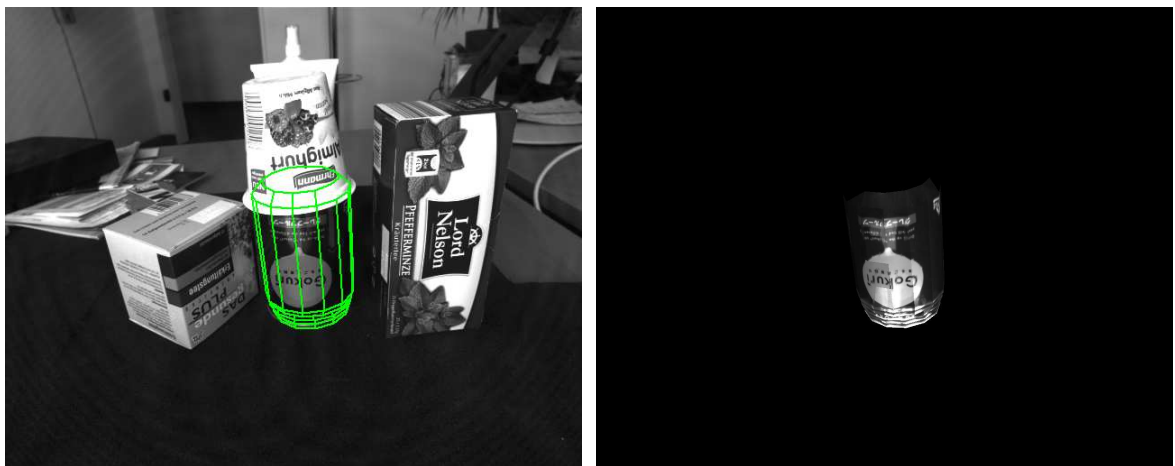


Figure 8.11.: An object matched based on descriptor points in a cluttered scene and the reconstruction of it.

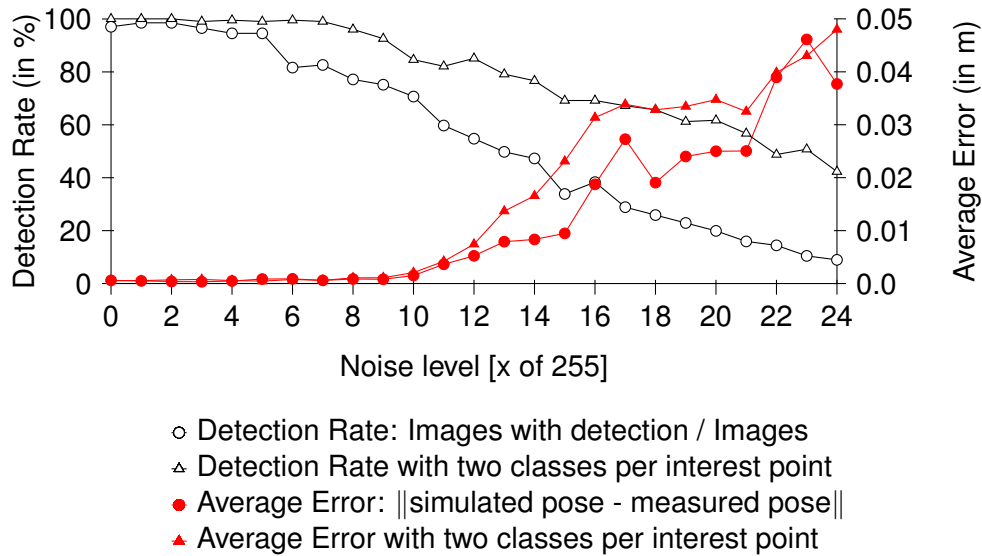


Figure 8.12.: Testing the localization error and rates of a yogurt cup in simulated images under white noise.

CAD model, in order to build a scaled 3D model. The positions of those images are approximately measured by a marker visible in all images, which could be replaced by the self localization of the robot. This was avoided in this test, in order to make the results easier comparable. We tested four objects, which can be seen in Figure 8.8. For those objects we could build textured 3D models which are visualized in Figure 6.13 and 8.9. For a challenging object regarding the texture is the object depicted in Figure 8.8(b). The drink label is the only feature with substantial size, while the rest are only letters which are not very discriminative in terms of the proposed method or any other feature point based method. Anyways, in Figure 8.10 and 8.11 we show situation in which we can localize the object well. In Figure 8.11 we additionally show the virtual view of the extracted position to visualize the accuracy of the method. A more formal test we want to present for the cone depicted in Figure 8.8(c): For simulated data we tested the localization method under the influence of white noise to the localization. The model is projected using the same technique as for training, so the task is much easier as for a real example. The run time for this graph per localization was about 300 ms. The graph in Figure 8.12 shows that the method works for data that is similar enough to the training images works perfectly, with a very low localization error and nearly no false or missed detection. Adding noise to the scene introduces new interest points and decreases the correct classification rate of those points, which results in a significantly higher localization error and missed detection.

This observation generalizes to the real scenes, the higher the lighting differences and the differences between the CAD model and the real shape are, the higher error rates can be

8. Results

observed.

To summarize, the descriptor based matching in 3D has high calculation and acquisition costs for the generation of the model, while it provides a fast method for recognition and localization of a certain object from all direction in a relatively large depth variance. The robustness and the degree of generalization from the training data is only mediocre compared to other evaluated methods, while it is very robust against false positives.

8.1.4. Transparent Objects

To demonstrate the performance of our system on transparent objects, we want to focus on two facts: First, does the system detect transparent objects without false positive detections. Especially, the system should not label any colors in the background or even opaque objects as transparent. Second, we want to show that the 3D reconstruction is good enough to try a grasp of this object without further modeling. Those results are also published in [54].

Testing Environments

To rate the performance of our approach, we tested our methods on a set of different objects shown in figure 8.13(a) and 8.13(b).



(a) Camera image of the objects illuminated by laboratory light with a high proportion of green.

(b) ToF point cloud of the objects.

Figure 8.13.: Objects that defined the testing environments.

The transparent objects are a plastic mug, a plastic wine glass, a drinking glass, a glass made of red plastic and a glass made of yellow plastic which appeared to be transparent for the SR4k. The opaque objects consisted of an ice tea container, a package of tea bags, a textured cup, a white cup, a white plastic cup, a light blue cup and a black thermos jug. We applied

our procedure to every object individually having it placed on a dark gray kitchen table. The positions on the table altered in a way that the respective object was positioned in the middle or close to one of the edges of the table. The dark gray table represents the more challenging scenario compared to a wooden or a white table, which is discussed in Section 4.4.3.

Accuracy and Grasping

How accurate our method performed overall on the test setup can be seen in table 8.3.

type	count	classified as	
		transparent	no transparent object
transparent	105	55.24%	44.76%
opaque	44	0.00%	100.0%
grasp success		24/58	-

Table 8.3.: Reconstructed transparent objects in the test with up to three objects in a scene.

We tested every mentioned object five times at different relative positions to the robot on a table. For the test we used a dark table, since it was a newly installed kitchen in our lab and it represents the most challenging scenario for our approach: It is striking that we had no false positive detection implying that our method did not try to reconstruct any object with a decent response in the ToF-camera. Accordingly, all the opaque candidates were omitted by the inconsistency checks. On the other hand, we achieved a higher amount of false negatives showing that our thresholds were immoderate at times. Yet we successfully reconstructed 58 out of 105 transparent objects and grasped 41% in a way that our platform confirmed its capability to further manipulate, relocate or retrieve the reconstructed object.

Our grasping method first executes a probabilistic encaging and applies corrections by re-actively repositioning the hand in case of collisions which are detected with force feedback sensors. The failed grasping attempts in our tests arouse from slight reconstruction errors that either overestimated the height of the object such that the grasping position was too high or introduced an inaccurate position estimation what lead to collisions between the grasper and the object. Those collision could eventually cause wrong corrections of the hand or they could be missed by the sensors in case of light objects like the plastic glasses, which would move the object out of the expected position.

This test includes objects solely on the table and three objects at the same time nearby not affecting the test performance which was for the single experiments all about approximately 53% - 58%. The following figure 8.14 denotes which stage of our checks principally contributed to candidates being omitted.

8. Results

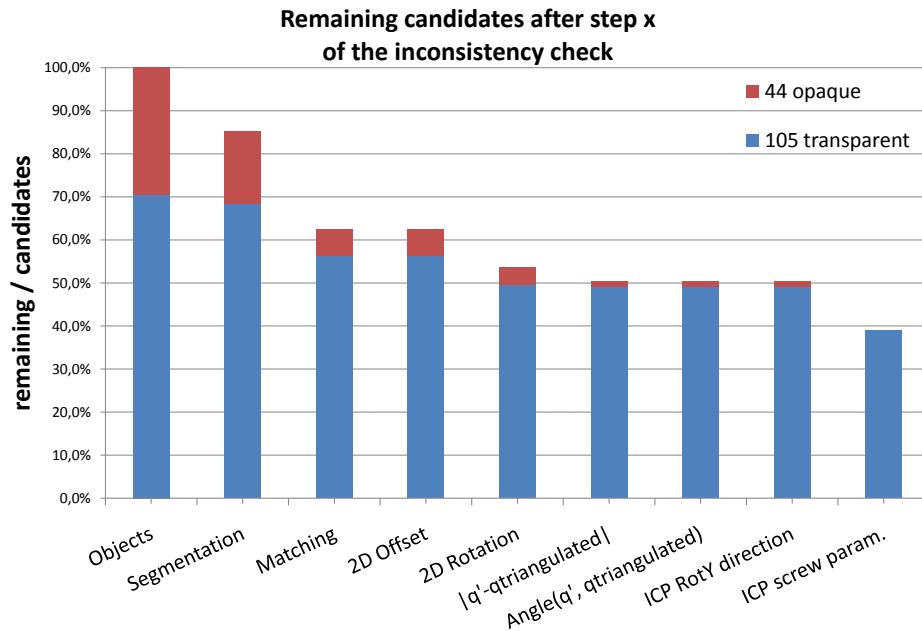


Figure 8.14.: Candidates remaining after a certain stage of the inconsistency checks.

As one can see, the candidates corresponding to opaque objects are mostly filtered by our segmentation method which originates from our initial idea of transparent objects yielding darker intensities through the doubled absorption. Our method filters further false positives in the matching phase in that the transparent object might not be found in the second view because it was omitted in the previous segmentation which of course has to be applied again to the second view data. The remaining opaque objects then get discarded because they do not bear the twist a transparent object creates. The amount of wrong negatives can be tracked in figure 8.14 as well. We only lost 3 positive candidates in our segmentation due to heavy occlusion but a fair amount due to unsuccessful matching expectedly. The loss occurring during the check of the 3D twist was identified to be an inconsistency in the calculations of our ICP method. Nevertheless, the method did not reject candidates wrongly for the rest of our checks. The segmentation results for bright background improve the entire system. Preliminary tests show that on a white table, 8 of 10 objects could successfully be detected and grasped being in presence of opaque clutter ¹.

¹see video <http://www.youtube.com/watch?v=QLzZtj9hqxQ>

8.2. Robotic Manipulation

In robotic manipulation we consider here two challenges: the grasping of unmodeled objects and a visual guided manipulation. The first method is a general approach to grasp objects, given a minimal amount of sensor data and interpretation of sensor data. The second part deals with the problem of tool calibration and tool usage for a certain task, which is not a general solution, but still an interesting problem to discuss here.

8.2.1. Unmodeled object Grasping

The following experiments show the robustness of the proposed grasping system for unmodeled objects, as well as the entire system accuracy. TUM-Rosie tries to pick up and set back objects that are only perceived as a cluster by selecting a grasp based on the minimal collision probability described in Section 7.3.2.

Handling Different Objects



Figure 8.15.: The household items used for testing. Results in Table 8.4.

The test set contains 15 objects which can be seen in Figure 8.15. We consider a grasp attempt successful if the object was lifted from the table, and held in the hand for at least 10 seconds. The statistics about successful grasps can be found in Table 8.4. The robot grasped correctly the majority of the random kitchen objects. The overall success was approx. 80 percent, decreasing mostly due to one object (“Nivea shower gel”), that was so slim that the large had of TUM-Rosie could not encage it. Since the system can recognize failed grasp attempts, it can repeat the action (including perception), and this would improve the reliability. Anyways, the tests recorded here did not allow a second attempt.

8. Results

Object	Size(WxHxD in cm)	Success/Trials
Tape roll	5x5x4	2 / 4
White Porcelain cup	6x6x4	4 / 4
Blue Porcelain cup	6x6x4	3 / 4
Melitta Coffee Filters Small	2x5x7	2 / 2
Paper Towel	28x7x7	4 / 4
Melitta Coffee Filters Big	4x8x20	4 / 4
Assam-Blend Tea box	7x10x17	4 / 4
Can of Peas	11x7x7	4 / 4
Soup Box (Heisse Tasse)	11x7x7	4 / 4
Green Teapot	22x17x12	1 / 1
Peppermint Tea box	16x7x6	4 / 4
Nivea Shower gel	17x8x3	1 / 8
Iced Tea	27x10x10	4 / 4
Leibniz cookies	22x7x4	4 / 4
Paper cup	6x6x4	3 / 4
Total		48 / 59

Table 8.4.: Results of the grasping experiments on household items.

Cleaning up a Cluttered Scene

Another experimental setup can be seen in Fig. 8.16, in which the task is to clear the table by picking up the objects on it. The robot successfully picked up all objects one after the other. Here we used the description of point distributions as obstacles for a reactive vector field controller. The simple distributions allowed safe navigation through the scene from and to the object. The grasps were selected with the same method as before, with the collision minimization metric including obstacles.

Handling Obstacles

This problem gets more complex if the trajectory of the object is not out of the scene but through the scene. Such a scenario can be seen in Figure 8.17. This scenario shows TUM-Rosie picking up a box and bringing it to the commanded position on the right. The target was set on the same height as the starting point. Still it can be seen, that the simple obstacle modeling allows the robot to safely navigate the arm with the object around.

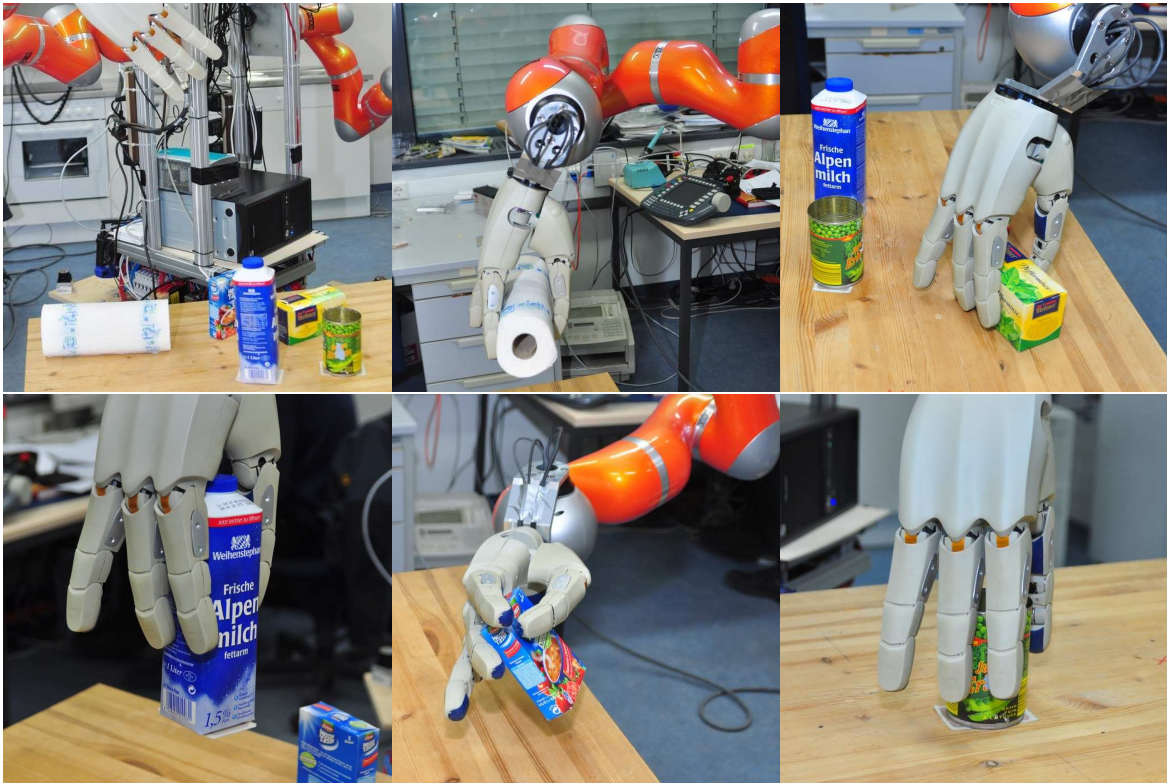
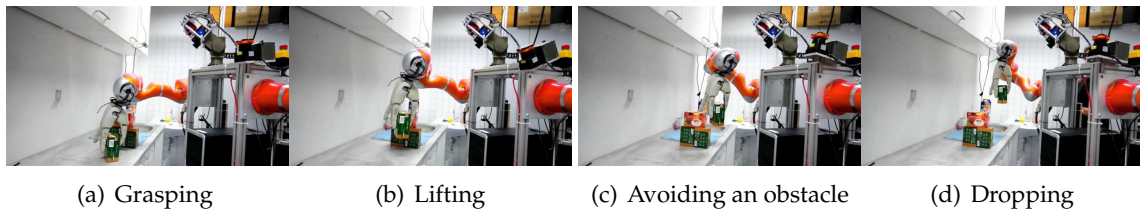


Figure 8.16.: Grasping all the objects from a cluttered scene.



(a) Grasping

(b) Lifting

(c) Avoiding an obstacle

(d) Dropping

Figure 8.17.: This sequence illustrates a transport task using collision avoidance.

8.2.2. Online Tool Calibration for Complex Tasks

Given a robot has to address a problem like the flipping of a pancake, we will need a special tool for it. Also assumed we have here our general purpose platform and not a specialized robot, we first have to pick up the tool. Principally, this is not a harder challenge than grasping an unmodeled object, but afterwards we need the exact location of our tool parts. Even if we model the tool exactly and plan the grasp accordingly to our purpose, the final position to the hand will essentially vary from time to time. This is especially the case, when the tool is placed randomly in the scenario. In order to compensate this lack of accuracy, an online inspection of the new tool has to be performed. These results are also published in [62].

8. Results



(a) The tool frame that has to be calibrated.

(b) The planar shape template that was used as model of the spatula.

Figure 8.18.: The tool that has to be calibrated in the pancake preparation scenario with TUM-Rosie.

One solution is the scenario of pancake preparation with TUM-Rosie's tool we had to use a spatula, which can be seen in Figure 8.18(a), which was modeled manually. In a more general test setup, the tools are in the hand of the robot, such that their position is approximately known (search space 15cmx15cmx15cm). We classify the concavity and extract the lines and holes. Table 8.5 show the results we got for several trials for each of the tools. If there was variation in the results the minimal and maximal numbers of results are printed. The results show, that we can in the setup on the robot robustly detect the features we are interested in. Visualizations for some selected results can be found in Figure 8.19. Only one of the spoons, namely the wooden spoon, could not be classified correctly. Its concavity was too small to be detected with the Kinect sensor. The measured maximal difference between the two circles was beyond the sensor noise of the Kinect. Big holes are detected completely accurate, while the number of small holes is not estimated accurately, but the existence of holes is detected correctly. Depending on the positioning of the tool in the search space, additional edges may be detected from the end of the tool-head towards the handle. Those edges created the variation in the number of detected edges, this is also not considered as an error, if the relevant edges were detected correctly, too.

With our simple ontology we could come from the features to a robust classification, which only fails for features that are smaller than the sensor resolution. The combination of a semantic model and the features we propose here turns out to be valid for the class of tools we investigate here.









Tool	Concavity/Plane	Num Edges	Num Holes	Resulting Tooltype
	Concavity	6	81	Skimmer
	Plane	3	3	Spatula
	Concavity	2-3	0	Spoon
	Concavity	0-1	0	Spoon
	Concavity	4-6	22-35	Skimmer
	Plane	2	0	Spatula
	Plane	0	0	HandTool
	Plane	3-4	15	Spatula

Table 8.5.: The visual exploration results for the set of tools, errors are marked in red.

8.3. System Evaluation

In order to show the performance of our systems we present several experiments which are executed on recorded scenarios. Additionally we will present an online task which requires all capabilities of this system. In this context, the observed behavior of the system will be discussed.

8. Results

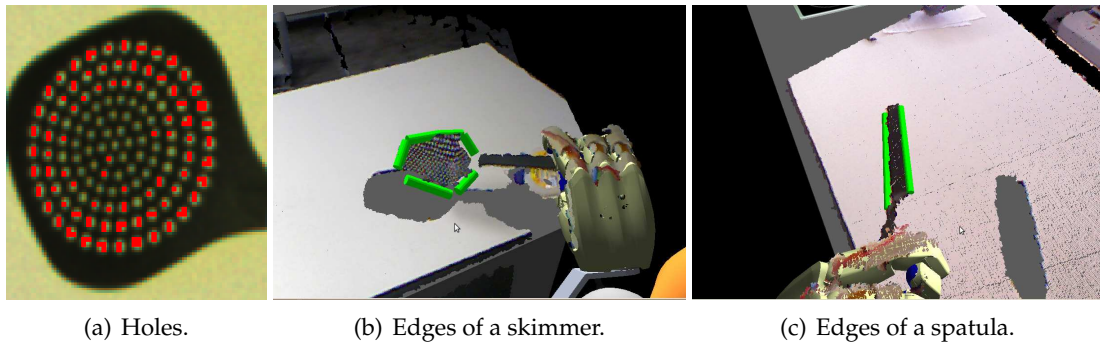


Figure 8.19.: Visualization of several results: First the small holes which could only be partially detected, then two examples for extracted edges.

8.3.1. Offline Evaluation

First we want to show the capabilities of COP in a simple setup: the learning over time of unmodeled objects in three steps, under the precondition that good feedback is provided. First, we show the capabilities of recognize previously seen objects, then we show the capabilities to understand a full scene and finally show the converged values of method evaluation for an object.

Recognize Unmodeled Objects

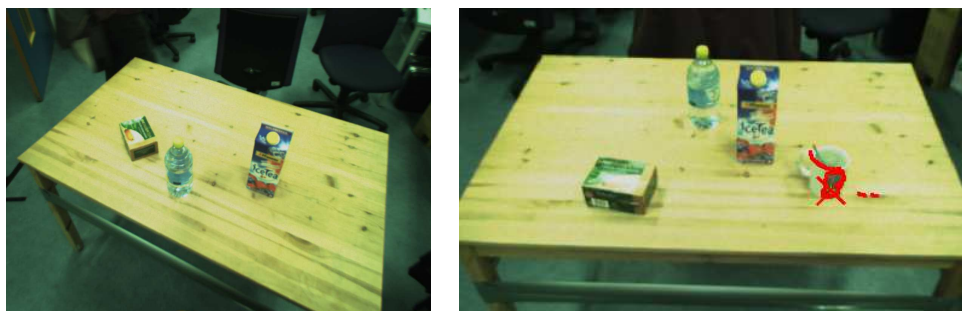
The first experiment evaluates localization of objects in different scenes, without prior modeling of the objects or their types. Here we use the object candidate detection followed by learning the shape and a color model. The setup has no assumption about passed time or movement between the scenes and positions. In Table 8.6 the present objects can be seen, with the information when they appeared and from what positions they can be seen. The table contains the current belief state of objects position which is coded as follows: Belief state which correspond to the actual scene are "1", "out" and "new", while "out" means a previously seen model is currently not in the scene, "new" means there was a previously unseen model newly appearing in the scene and "1" says a previously seen model is still in the scene. Not or wrongly detected world states are labeled with "0", "-" or "err", while "-" means an existing object was not seen previously and is not in the scene (not an error), "0" depicts objects in the scene, modeled before but not detected, and "err" labels an object that was interpreted as a new object while it was seen and modeled before. The scenes are depicted in Fig. 8.20.

Concluding the results, most of the objects that appear in the object candidate detection are detected well. Objects like a plate, is often not detected in the ToF data and was not detected at all in this test. The bottle is semitransparent and lacks also of a reliable response in all

Object	Scene 1			Scene 2			Scene 3			Scene 4		
	P1	P2	P3	P1	P2	P3	P1	P2	P3	P1	P2	P3
at												
Teabox	new	1	1	0	1	1	out	out	out	out	out	out
Bottle	new	1	1	1	0	0	1	1	1	1	1	0
Mug	-	-	-	new	1	1	out	out	out	out	out	out
Iced tea	new	1	1	1	1	0	out	out	out	err	1	1
Coffee	-	-	-	-	-	-	new	1	1	1	1	1
Jug	-	-	-	-	-	-	new	1	1	out	out	out
Plate	-	-	-	-	-	-	0	0	0	0	0	0

Table 8.6.: The list of recognized object over four scenes from different view points. It shows if the system's belief state differs from the actual scene. The errors are encoded with (0, err) and success is labeled with (1, out, new), please find a precise definition in the text.

sensors, and also has major problems. The last problem can be seen in for the iced tea which appears once in a frontal view and once in a back side view. This two appearances could not be merged. Besides those problems, the system can keep track of the other objects, given it has at least two views of a scene.



(a) First scene from first pose (P1 in Table 8.6).

(b) Second scene at the second pose (P2) with the relocated mug

Figure 8.20.: Two of the scenes used for the evaluation.

Recognize Modeled Objects

Now, we allow the system to prepare itself for this task by training it on one additional scene containing all objects and annotating the object with a semantic concept that allows also the usage of available CAD models for the objects. This leads to far better results which can be found in Table 8.7. In this experiment we can see, that only the bottle was not detected twice and the system hallucinated one *Plate*, while all other states (if an object is in the scene or not) are detected correctly. In this experiment the training phase was supervised and not autonomous, but it shows that the system is able to use additional information to improve

8. Results

Object	Scene 1			Scene 2			Scene 3			Scene 4		
	P1	P2	P3	P1	P2	P3	P1	P2	P3	P1	P2	P3
at	new	1	1	1	1	1	out	out	out	out	out	out
Teabox	new	1	1	1	1	1	out	out	out	out	out	out
Bottle	new	1	1	1	0	0	1	1	1	1	1	1
Mug	out	out	out	new	1	1	out	out	out	out	out	out
Iced tea	new	1	1	1	1	1	out	out	out	1	1	1
Coffee	out	out	out	out	out	out	new	1	1	1	1	1
Jug	out	out	out	out	out	out	new	1	1	out	out	out
Plate	out	out	out	err	out	out	1	1	1	1	1	1

Table 8.7.: The system’s belief state against the Errors are again encoded by (0, err) and success by (1, out, new), definitions see text.

the robustness. Fig. 8.21(a) and 8.21(b) show CAD matching of the two critical objects in the last experiment.



(a) Matched iced tea model.

(b) Matched Plate model.

Figure 8.21.: Scene three from third pose (P3).

8.3.2. Online Evaluation

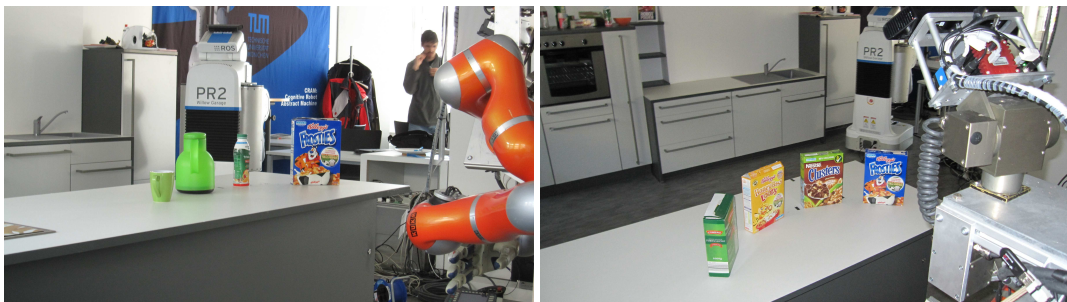
In order to show the performance of the system in a direct world interaction, the outcome of two experiments as described in the following sections. First, we will show the success rate of one-shot learning in order to recognize a previously unseen object again. Second, we want to evaluate how the system can autonomously test if a segmentation was successful or not, by using the one-shot learning mechanism and the grasping of unmodeled objects. Those results were also summarized in [55].

	Shape Model	Surface Model	Learning
Boxes	69.6%	55.0%	76.3%
Texture-less	51.9%	62.5%	83.5%
Diverse	62.1%	60.0%	76.0%
All	67.9%	59.2 %	78.2%
Measurements	125	240	239

Table 8.8.: Comparison of the two methods and the learning process over the methods for different object scenarios.

Evaluation of Model Selection

To get an intuition how often one-shot learning will succeed and return a valid model we made the following experiment: We let the robot learn several objects in a scene from one view, and let the robot to a set of points inside a region of about 2 square meters. From all points the robot gets a glance to the scene and tries to localize the newly learned objects in the scene. For this localization task, the search space is restricted to “on the table” and the assumption of the previously known location is dropped internally to prevent favoring of a similar location. This experiment should show first, how different methods handle this task and how the learning internal mechanism of COPwork.



(a) One of the ‘diverse’ scenes.

(b) One of the ‘boxes only’ scenes.

Figure 8.22.: Examples of scenes that were used to evaluate COP’s model selection.

We extracted from a repetition of this experiment several key parameters of our system which are summarized in Table 8.8: For the average object in our kitchen the planar shape model performs slightly better than the surface based model. But this result can be heavily influenced by the selection of special subsets. When a subset with texture-less objects is selected, the surface model is clearly favored. On the other hand, if a subset is selected including different object of similar shape like boxes, the shape model will be performing better. Nevertheless, if the perception system is allowed to decide which model to use, the performance is far better, namely 78.2% correct detection within the localization error of

8. Results

the robot compared to 59% and 68% for constantly favoring one method. This requires, of course, that after every trial, feedback about the success is provided to the system. The learning switched in average 2 - 3, maximally 5 times between the two models until the learning process converged for a static scene. E.g. in the `Boxes` scenario with several equal boxes and one distinct box, all but the distinct box converged to the shape model. This shows the high context awareness of the method, while never executing two methods at once in order to avoid additional execution time.

Evaluation of the Pick and Place Plan

In short, we execute the following plan on the robot to execute the following standard pick and place task in order to evaluate our system in regards to capabilities to check the segmentation, which is crucial for several of the model building procedures:

- Segment the scene
- Learn models for all objects
- Try to grasp one of them
- Place the object into another context
- Validate the predicted position with models

More interesting than this short description are the possibilities to detect and recover from errors in this plan: If the models are learned and one of the objects is selected, for this object, the model can be preliminary validated by moving the robot in order to grasp the object and using the new model to localize the object again before grasping it. If this fails, the robot will retry which might already cause the use of another model in the hierarchy. While grasping, the hand's torque parameters are observed to detect any unpredicted collisions, which can give information about unseen objects or an inaccurate position estimation of the target. This information might already indicate that the robot has moved the object and in case nothing is grasped the object has to be localized again. If the robot grasped something, is decided based on the joint angles of the hand after the grasp. When they correspond more to a fully closed hand than to a hand holding an object, the grasp is considered a failure.

If the object is put down to an empty spot on the table, the near environment of the put-down location is scanned for the object again. If it appears to be close enough to the predicted position, the entire process is considered as success. The model should even be able to localize the object if it was not put down carefully enough to stand upright.

The high-level control program executes this simple pick-and-place plan and observes the different errors that can occur. For instance, if the object is not reachable, the vision system

	Segmentable	Not Segmentable
Error in pickup phase	30%	50%
Error in put down phase	0%	10%
Model failed post move eval.	0%	20%
Plan Succeeded	70%	20%
Trials	30	10

Table 8.9.: Statistics of result states of the plan for test scenes.

is most probable not responsible for it but the executive's choice to reach the object was bad. On the other hand, if the manipulation system does not detect the object in the gripper, the perceived object location was most probably bad and the perception system is sent a negative evaluation value. Other errors that occur include that the object could not be re-detected which indicates that the learned object model does not fit under the current environment configuration and location of the robot. Such errors are handled locally by moving the robot to a different location and trying to perceive and pick up the object from there. If the robot could successfully pick up the object, the system puts it down at a different location and then searches for it again at the put down location. Since the robots places the object at a defined location, is have a rather good knowledge about the location of the object. The highlevel system therefore can measure the distance between a new detection of the object and the put-down pose. If the object is close to the desired pose, the perception system receives a reward. A re-detection of the object that is not close to the put-down pose indicates a false-positive and the currently learned model for that object receives a negative

The segmentation is a crucial precondition for a new object candidate in nearly all current methods as well in the tow methods we used in this experiment. In order to validate the segmentation and with the segmentation also the quality of the model we want to perform the following test: The robot will only be capable to move the object away from its position when the segmentation was correct, otherwise, it will only move parts of the object or it will fail initially. We will apply the previously mentioned action².) to a scene which was setup with one of two modes: first, easily segmentable meaning objects have a distance over 5 centimeter to each other and second, not segmentable in 3D space, meaning no distance between at least two objects.

The results we collected can be seen in Table 8.9 and can be interpreted as follows: In case of a successful segmentation, the objects could be always localized correctly using the learned models, which were used at least twice before. In both cases, in the segmentable scenes as well as in the not segmentable scenes our grasping system had some problems, in various cases caused by a bad model, which was not giving an accurate enough localization. Ex-

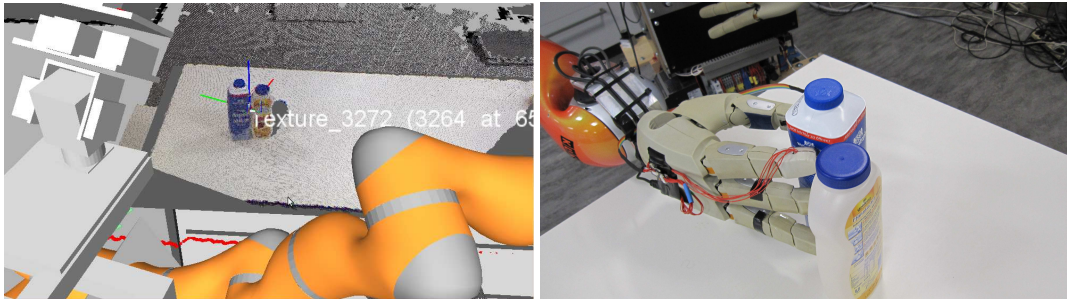
²High quality video of the task, in cooperation with A. Maldonado, L. Mösenlechner: <http://www.youtube.com/watch?v=9nwyUmdDIQ4>

8. Results



(a) A setup for the pick and place which is well segmentable. task. (b) A view of the verification step in Rviz, an object at a new position. task.

Figure 8.23.: Snapshots from the experiment, used to evaluate the segmentation and model building.



(a) A view in Rviz from the model learning step with a filed segmentation. (b) The subsequent grasp, that pick up one of the objects.

Figure 8.24.: The border case, when the grasping works even with wrongly segmented objects, has to be recognized.

amples for segmentable scenes can be found in Figure 8.23. In case of the not segmentable scenes we observed four times the trickier case which is when the robot moves part of the objects in the selected segment. Figure 8.24(a) shows such a segment and Figure 8.24(b) shows a grasp that was considered as successful while only lifting the left object, this object was then rejected due to low correlation of the placed object and the learned model. The bold numbers in Table 8.9 are the cases which successfully generated information that could be passed to the perception system. Only in two tests in which the plan was considered successful in the not segmentable scenes, the information led to a wrong believe state of the system about the quality of the model.

9. Conclusion

This work gives an insight in a perception system that is running on a mobile manipulation platform. We formalized context awareness in a perceptual system based on model-method-sensor selection and showed examples for the system's performance. COP addresses a large set of perceptual tasks and is well accessible for a highlevel system, while providing results with a sufficient accuracy for manipulation.

Combining the internal evaluation of triples, consisting of model, method and sensor, to improve the method selection process, enabled us to easily adapt a complex perception system to specific situations like specific backgrounds, kinds of textures or model quality. Especially in a well known environment, we observed a good performance, while also observing automatic adaptation to new scenarios.

Additionally, this work describes a large set of methods which were made applicable to the specified scenario on a mobile platform. Those methods were tested and evaluated to serve the intended purpose and have shown good performances in all tests as well as in live demonstrations. Especially the detection of transparent objects and the fast 3D localization of textured 3D objects had to be newly developed and specialized to the scenario of a mobile manipulation platform in a domestic environment.

Observing the evolution of the robot's behavior using the presented system showed a great advance of the capabilities of the robot. The abstraction of the basic perception system, from sensors and the platforms capabilities turned out to be of great use and allowed the fast integration with new hardware.

9.1. Summary of Results

To be more specific, we came to a conclusion regarding state of the art object localization and recognition methods: Not all objects can be recognized by a single method with optimal performance, but we could find a method which will recognize any specific object. Unfortunately, we cannot use all available methods, which are applicable to an object. This it would exceed our anyways relatively soft real time requirements, which are imposed by the problems we wanted to solve, e.g. finding the pancake before it burns. We showed on the one

9. Conclusion

hand how simple 2D methods with very low preconditions on the modeling. On the other hand, we showed that also nearly perfect modeling of the object may have strengths and weaknesses in certain setups. We discussed the using of 3D information and the advantages as well as the risks regarding transparent objects which do not appear in a usual 3D sensor. In this context we also showed interesting results on how we could anyways reconstruct transparent objects and interact safely with them.

This variety of methods and applicability is why we proposed the solution of online learning of the best applicable method, which turned out to give better results even in the learning phase, without increasing the calculation time used to recognize an object.

We opted for such a system, since we needed a solution which could be executed embedded on the robotic platform. In a realistic setup we still would need more computation power to reduce the maximal reaction time of the current system which is up to several seconds for the analysis of a complex scene. As reaction time of our system we measured currently an average under half a second using an automatically learned model. This measurements were done for a specific task, which did not require only the analysis of a single object.

9.2. Hardware Challenges

Not only the restriction of computation power but also the limitations of the rest of the hardware caused many of the challenges faced in this work. A mobile platform is never as mobile as a human, if it runs on wheels. Especially in the case of TUM-Rosier or the TUM-James, the base size limited the application to large rooms and kept the platforms and their sensors away from the objects of interest. The same applies to most available grippers, they are for many tasks either too large or too small and underpowered. Developments in the mentioned and other areas in robotic hardware will nearly automatically open new application scenarios and allow robots to solve more tasks.

Like we discussed earlier in this work, also the sensors available on the market are not perfect for the application which were tackled in this work. Higher resolution in cameras and in 3D sensors will boost most of the proposed algorithms to higher accuracy and robustness.

9.3. Future Work

An open issue in current research is the clean integration and modeling of motion and moving object. Both things are only partially tackled here: If the necessary method to detect

an object is not fast enough, the used static prediction model will not be sufficient satisfactory. Several problems like motion blur, motion prediction, delays in camera images are not modeled or considered yet. The integration of specialized systems or a deep integration of a position extrapolation could help here.

Another open issue is the observation of scenes in which humans also interact with. The detection of a human and the its possible influences on the environment should be considered in the system. Until now, we did only few work on detecting human faces with standard methods inside this framework and had a look in obstacle awareness. The perception mechanisms which are necessary to interact with a human are definitely different compared with the requirements of a human inhabited environment without a real interaction. This topic might profit from the capabilities of the Kinect sensor, which might give easy access to observation of humans.

The potential of robots to act safely and smartly in a domestic environment is very high. The range of possible applications was extended by this work. So we improved the handling of unknown or previously undetectable objects, as well as we enabled the autonomous training of robotic systems in recognizing new objects. These interesting capabilities are easily accessible for an automatic highlevel control instance. But those capabilities are definitely not yet enough to enable already today a robot to act sufficient safely in any domestic environment. This safe acting would be an important requirement for a ready-to-use service robot and it requires further research.

9. Conclusion

Part V.

Appendix

A. Table of Sensor Properties

A. Table of Sensor Properties





Sensor	Properties	Example	Picture
RGB Camera	<ul style="list-style-type: none"> • High resolution • Perceives Color • Fast • No 3D 	SVS Vistek eco 274	
Stereo Cameras	<ul style="list-style-type: none"> • Low resolution • Perceives Color • Fast • Little 3D 	Videre STOC	
LASER range sensor	<ul style="list-style-type: none"> • Medium resolution • Perceives Intensities • Slow • Accurate 3D 	Hokuyo UTM-30LX Laser	
Time of Flight Sensor	<ul style="list-style-type: none"> • Low resolution • Perceives Intensities • Fast • Rough 3D 	Swissranger 4000	

Table A.1.: Discussed sensors with their properties and one example of a sensor with an example of its data output. Part I.



<p>Structured Lighth Stereo</p>	<ul style="list-style-type: none"> • Medium resolution • Perceives Intensities • Fast • Rough 3D 	<p>Willowgarage PR2 Head</p>	
<p>Optical Lattice Monocular Depth</p>	<ul style="list-style-type: none"> • Medium resolution • Perceives Intensities • Fast • Good 3D 	<p>Kinect</p>	

Table A.2.: Discussed sensors with their properties and one example of a sensor with an example of its data output. Part II.

A. Table of Sensor Properties

B. List of Implemented ROS Packages and their References in this Work

B.1. `cognitive_perception`

The central package is '`cognitive_perception`'. Most methods explained in this work are implemented as plugins for this package, which can be loaded during running the cognitive perception server '`cop_srv`'. The architecture is explained in Chapter 2 and reflects also the way it should be integrated into any system. The documentation can be found in the wiki ros.org under in the section for `cop`.

B.2. `cop_halcon_plugins`

The package '`cop_halcon_plugins`' depends on the Machine Vision library [HALCON](http://halcon.mvtec.com) of [MVTec](http://mvt.com). The methods contained in this packages are partially explained in this work. The package `cop_halcon_plugins` contains:

- `TwoInOneAlg`
- `IntersectTwoRuns`
- `CheckColorClass`
- `DeformShapeBasedAlg`, Section 5.1.3
- `FindCalTab`
- `HClusterDetector` , Section 4.2.1
- `RFAColorByShape`
- `ShapeModelDownloader`, Section 6.2
- `RFAClassByDPs`
- `PanCakeDetector`

B. List of Implemented ROS Packages and their References in this Work

- DetectPlate
- RFADeformByCluster, Section 6.1.3

B.3. cop_cad_plugins

The package 'cop_cad_plugins' depends on the Machine Vision library [HALCON](#) of [MVTec](#). The package cop_cad_plugins contains:

- SurfaceDetection
- ShapeBased3DAlg, Section 5.1.2

B.4. cop_ros_plugins

The package 'cop_ros_plugins' contains:

- FaceDetection

B.5. cop_sr4_plugins

The package 'cop_sr4_plugins' contains:

- ClusterDetector, Section 4.2.1

B.6. cop_transparent_objects_plugins

The package 'cop_transparent_objects_plugins' depends on the Machine Vision library [HALCON](#) of [MVTec](#). The package cop_transparent_objects_plugins contains:

- DetectTransparentObjectCandidate, Section 4.4
- DetectTransparentObject, Section 4.4

B.7. cop_barcode_plugin

The package 'cop_barcode_plugins' depends on the Machine Vision library [HALCON](#) of [MVTec](#). The package cop_barcode_plugin contains:

- FindBarCode

B.8. cop_odu_plugin

The package 'cop_odu_plugin' contains:

- CopOduRefine

B.9. cop_tool_plugin

The package 'cop_tool_plugins' depends on the Machine Vision library [HALCON](#) of [MVTec](#). The package cop_tool_plugin contains:

- FindPlanes, Section 4.3
- FindLines, Section 4.3
- FindConcavity, Section 4.3

B. List of Implemented ROS Packages and their References in this Work

Bibliography

- [1] S. Agarwal, S. Mallick, D. Kriegman, and S. Belongie. On refractive optical flow. *Computer Vision-ECCV 2004*, pages 483–494, 2004.
- [2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 9, Issue 5*, pages 698–700. IEEE, 1987.
- [3] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. Armar-iii: An integrated humanoid platform for sensory-motor control. In *Proceedings of the IEEE/RAS/RSJ International Conference on Humanoid Robots (Humanoids06)*, pages 169–175, 2006.
- [4] P. Azad. *Visual Perception for Manipulation and Imitation in Humanoid Robots*. PhD thesis, Universität Karlsruhe (TH), 2008.
- [5] P. Azad, T. Asfour, and R. Dillmann. Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009*, pages 4275–4280, 2009.
- [6] B. Baeuml, T. Wimboeck, and G. Hirzinger. Kinematically optimal catching a flying ball with a hand-arm-system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2009.
- [7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.
- [8] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *9th European Conference on Computer Vision*, May 2006.
- [9] M. Beetz, L. Mösenlechner, and M. Tenorth. CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *IEEE/RSJ International Conference on Intelligent RObots and Systems.*, 2010.
- [10] M. Beetz, F. Stulp, P. Esden-Tempski, A. Fedrizzi, U. Klank, I. Kresse, A. Maldonado, and F. Ruiz. Generality and legibility in mobile manipulation. *Autonomous Robots Journal (Special Issue on Mobile Manipulation)*, 28(1):21–44, 2010.

BIBLIOGRAPHY

- [11] M. Ben-Ezra and S. Nayar. What Does Motion Reveal About Transparency? In *Proceedings of the Ninth IEEE International Conference on Computer Vision-Volume 2*, page 1025. IEEE Computer Society, 2003.
- [12] P. Besl and H. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.
- [13] P. Bevington and D. Robinson. *Data reduction and error analysis for the physical sciences*. McGraw-Hill New York, 1969.
- [14] G. Bradski, R. B. Rusu, and K. Konolige. Opencv: Solutions in perception challenge. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop. Defining and Solving Realistic Perception Problems in Personal Robotics*, October 2010.
- [15] N. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. *Image and Vision Computing*, 2008.
- [16] C. Choi and H. I. Christensen. Cognitive vision for efficient scene processing and object categorization in highly cluttered environments. In *IEEE/RSJ Intl. Conf. on Intell. Robots and Systems*, page (in press), St. Louis, MO, Oct. 2009. IEEE.
- [17] K. Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research*, 18(3):286, 1999.
- [18] R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.
- [19] R. Diankov, S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation planning with caging grasps. In *2008 IEEE International Conference on Humanoid Robots*, December 2008.
- [20] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.
- [21] R. Eidenberger and J. Scharinger. Active Perception and Scene Modeling by Planning with Probabilistic 6D Object Poses. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2010.
- [22] G. Eren, O. Aubreton, F. Meriaudeau, L. Secades, D. Fofi, A. Naskali, F. Truchetet, and A. Ercil. Scanning from heating: 3D shape estimation of transparent objects from local surface heating. *Opt. Express*, 17:11457–11468, 2009.

- [23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. <http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html>.
- [25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. <http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html>.
- [26] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.
- [27] M. D. F. Trujillo-Romero. Appearance-based and active 3D object recognition using vision . In *International Conference on Computer Vision Theory and Applications, (VISAPP 2009)*. Springer, 2009.
- [28] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning Object Categories from Google’s Image Search. *Tenth IEEE International Conference on Computer Vision (ICCV), 2005.*, 2, 2005.
- [29] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–780, 1965.
- [30] P. Forssen, D. Meger, K. Lai, S. Helmer, J. Little, and D. Lowe. Informed visual search: Combining attention and object recognition. *IEEE International Conference on Robotics and Automation (ICRA), 2008*, pages 935–942, 2008.
- [31] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 2009.
- [32] S. Geidenstam, K. Huebner, D. Banksell, and D. Kragic. Learning of 2D grasping strategies from box-based 3D object approximations. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.
- [33] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The Columbia Grasp Database. In *International Conference on Robotics and Automation (ICRA), 2009*.
- [34] R. Gonzalez and P. Wintz 2nd. Digital Image Processing 2nd Edition Addison Wesley. Reading, Mass, 1987.

BIBLIOGRAPHY

- [35] D. Gonzalez-Aguirre, T. Asfour, E. Bayro-Corrochano, and R. Dillmann. Improving model-based visual selflocalization using gaussian spheres. In *To Appear, International Conference AGACSE 2008*, 2008.
- [36] D. Gonzalez-Aguirre, T. Asfour, E. Bayro-Corrochano, and R. Dillmann. Model-based visual self-localization using geometry and graphs. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–5. IEEE, 2009.
- [37] D. Gonzalez-Aguirre, S. Wieland, T. Asfour, and R. Dillmann. On Environmental Model-Based Visual Perception for Humanoids. In *Proceedings of the 14th Iberoamerican Conference on Pattern Recognition: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 901–909. Springer, 2009.
- [38] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34–46, 2007.
- [39] T. Grundmann, R. Eidenberger, M. Schneider, M. Fiegert, and G. Wichert. Robust high precision 6d pose determination in complex environments for robotic manipulation. In *Workshop of Best Practice in 3D Perception and Modeling for Mobile Manipulation at the International Conference on Robotics and Automation*, 2010.
- [40] S. Gudmundsson. *Robot vision applications using the CSEM swissranger camera*. Master’s thesis. Informatics and Mathematical Modelling. Technical University of Denmark. Copenhagen. Denmark, 2006.
- [41] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.
- [42] R. Hartley and A. Zisserman. *Multiple view geometry*, volume 642. Cambridge university press Cambridge, UK, 2000.
- [43] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2000.
- [44] S. Hinterstoisser, S. Benhimane, and N. Navab. N3m: Natural 3d markers for real-time object detection and pose estimation. In *IEEE International Conference on Computer Vision*, 2007.
- [45] A. Hofhauser, C. Steger, and N. Navab. Edge-based template matching with a harmonic deformation model. In A. K. Ranchordas, H. J. Araújo, J. M. Pereira, and J. Braz, editors, *Computer Vision and Computer Graphics: Theory and Applications - International Conference VISIGRAPP 2008, Revised Selected Papers*, volume 24 of *Communications in Computer and Information Science*, pages 176–187, Berlin, 2009. Springer-Verlag.

- [46] A. Hofhauser, C. Steger, and N. Navab. Perspective planar shape matching. In K. S. Niel and D. Fofi, editors, *Image Processing: Machine Vision Applications II*, Proc. SPIE 7251, 2009.
- [47] R. Hoover, A. Maciejewski, and R. Roberts. Pose detection of 3-d objects using s2-correlated images and discrete spherical harmonic transforms. *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pages 993–998, May 2008.
- [48] I. Horswill. Integrated systems and naturalistic tasks. In: Strategic Directions in Computing Research AI Working Group, 1996.
- [49] E. Hsiao, A. C. Romea, and M. Hebert. Making specific features less discriminative to improve point-based 3d object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010.
- [50] I. Ihrke, K. Kutulakos, H. Lensch, M. Magnor, and W. Heidrich. State of the art in transparent and specular object reconstruction. *STAR Proc. of Eurographics*, pages 87–108, 2008.
- [51] T. Inamura, K. Okada, S. Tokutsu, N. Hatao, M. Inaba, and H. Inoue. HRP-2W: A humanoid platform for research on support behavior in daily life environments. *Robotics and Autonomous Systems*, 57(2):145–154, 2009.
- [52] C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):20–29, 2007.
- [53] O. Khatib, L. Sentis, and J. Park. A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *European Robotics Symposium 2008*, pages 303–312. Springer, 2008.
- [54] U. Klank, D. Carton, and M. Beetz. Transparent object detection and reconstruction on a mobile platform. In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May, 9–13 2011.
- [55] U. Klank, L. Mösenlechner, A. Maldonado, and M. Beetz. Robots that validate learned perceptual models. In *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012. Accepted for publication.
- [56] U. Klank, D. Pangercic, R. B. Rusu, and M. Beetz. Real-time CAD Model Matching for Mobile Manipulation and Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 290–296, Paris, France, December 7-10 2009.

BIBLIOGRAPHY

- [57] U. Klank, M. Z. Zia, and M. Beetz. 3D Model Selection from an Internet Database for Robotic Vision. In *International Conference on Robotics and Automation (ICRA), Kobe*, pages 2406–2411, 2009.
- [58] K. Konolige. Sparse bundle adjustment. In *British Machine Vision Conference, Aberystwyth, Wales, 08/2010* 2010.
- [59] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *International Journal of Robotics Research (IJRR)*, 29(10), 2010.
- [60] G. Kootstra, J. Ypma, and B. de Boer. Active exploration and keypoint clustering for object recognition. *IEEE International Conference on Robotics and Automation (ICRA), 2008*, pages 1005–1010, May 2008.
- [61] D. Kragic, M. Björkman, H. Christensen, and J. Eklundh. Vision for robotic object manipulation in domestic settings. *Robotics and Autonomous Systems*, 52(1):85–100, 2005.
- [62] I. Kresse, U. Klank, and M. Beetz. Multimodal autonomous tool analyses and appropriate application. In *11th IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, October, 26–28* 2011.
- [63] K. Kutulakos and E. Steger. A theory of refractive and specular 3D shape by light-path triangulation. *International Journal of Computer Vision*, 76(1):13–29, 2008.
- [64] P. Lagger, M. Salzmann, V. Lepetit, and P. Fua. 3d pose refinement from reflections. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.
- [65] F. Lazarus and A. Verroust. Three-dimensional metamorphosis: a survey. *The Visual Computer*, 14(8):373–389, 1998.
- [66] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1465–1479, 2006.
- [67] J. Li, E. Li, Y. Chen, L. Xu, and Y. Zhang. Bundled depth-map merging for multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2769–2776. IEEE, 2010.
- [68] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [69] A. Maldonado, U. Klank, and M. Beetz. Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2586–2591, Taipei, Taiwan, October 18-22 2010.

- [70] P. Mihailov. *Model Selection*. Diploma thesis. Department of Computer Science. Technische Universität München, 2010.
- [71] A. Miller, S. Knoop, H. Christensen, and P. Allen. Automatic grasp planning using shape primitives. *IEEE International Conference on Robotics and Automation (ICRA)*, 2003, 2:1824–1829 vol.2, Sept. 2003.
- [72] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–312, 1990.
- [73] G. Miller and W. Charles. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28, 1991.
- [74] D. Miyazaki, M. Saito, Y. Sato, and K. Ikeuchi. Determining surface orientations of transparent objects based on polarization degrees in visible and infrared wavelengths. *JOSA A*, 19(4):687–694, 2002.
- [75] 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.
- [76] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *International Conference on Intelligent Robots and Systems (IROS)*, 2006, Beijing, China, 2006.
- [77] A. Morales, T. Asfour, P. Azad, S. Knoop, and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *IROS*, 2006.
- [78] J. More. The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, pages 105–116, 1978.
- [79] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate Non-Iterative $O(n)$ Solution to the PnP Problem. *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007., pages 1–8, 2007.
- [80] L. Natale and E. Torres-Jara. A sensitive approach to grasping. 2008.
- [81] A. Ng, S. Gould, M. Quigley, A. Saxena, and E. Berger. STAIR: The STanford Artificial Intelligence Robot project. 2008.
- [82] W. Niblack. *An introduction to digital image processing*. Strandberg Publishing Company Birkeroed, Denmark, Denmark, 1985.

BIBLIOGRAPHY

- [83] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168. IEEE, 2006.
- [84] T. Oggier, M. Lehmann, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, and N. Blanc. An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRanger). *5249(65):534–545*, 2004.
- [85] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba. Vision based behavior verification system of humanoid robot for daily environment tasks. In *Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 7–12, 2006.
- [86] K. Okada, M. Kojima, S. Tokutsu, T. Maki, Y. Mori, and M. Inaba. Multi-cue 3D object recognition in knowledge-based vision-guided humanoid robot system. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*., pages 3217–3222, 2007.
- [87] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.
- [88] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE transactions on pattern analysis and machine intelligence*, 2009.
- [89] P. Paalanen, V. Kyrki, and J. Kamarainen. Towards monocular on-line 3d reconstruction. 2008.
- [90] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: probabilistic feature-based on-line rapid model acquisition. In *Proc. 20th British Machine Vision Conference (BMVC). London*, 2009.
- [91] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004 on XX*, pages 38–41. Association for Computational Linguistics, 2004.
- [92] J. Pelletier and X. Maldague. Shape from heating: a two-dimensional approach for shape extraction in infrared images. *Optical engineering*, 36:370, 1997.
- [93] J. Peters. *Object recognition using a web based product database*. Bachelor thesis. Department of Computer Science. Technische Universität München, 2010.
- [94] A. Petrovskaya, O. Khatib, S. Thrun, and A. Ng. Touch Based Perception for Object Manipulation. In *Robotics Science and Systems, Robot Manipulation Workshop*, 2007.

- [95] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2007*, 2007.
- [96] B. Pitzer, M. Styer, C. Bersch, C. DuHadway, and J. Becker. Towards perceptual shared autonomy for robotic mobile manipulation. *IEEE International Conference on Robotics and Automation (ICRA), 2011, Shanghai*, 2011.
- [97] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.
- [98] V. Pradeep, G. Medioni, and J. Weiland. Visual loop closing using multi-resolution sift grids in metric-topological slam. 2009.
- [99] J. Prankl, M. Zillich, B. Leibe, and M. Vincze. Incremental model selection for detection and tracking of planar surfaces. *Int. J. Comput. Vision*, 79(2):159–177, 2008.
- [100] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello. A visual odometry framework robust to motion blur. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2009.
- [101] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Ng. High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening. In *IEEE International Conference on Robotics and Automation*. Citeseer, 2009.
- [102] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *In IEEE International Conference on Robotics and Automation (ICRA 2009)*, Kobe, Japan, May 12–17 2009.
- [103] O. Chum and J. Matas. Optimal Randomized RANSAC. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 30(8):1, 2008.
- [104] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. *Arxiv preprint cmp-lg/9511007*, 1995.
- [105] A. C. Romea, D. Berenson, S. Srinivasa, , and D. 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.
- [106] A. C. Romea, D. Berenson, S. Srinivasa, and D. 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.

BIBLIOGRAPHY

- [107] D. Ruck, S. Rogers, M. Kabrisky, M. Oxley, and B. Suter. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *Neural Networks, IEEE Transactions on*, 1(4):296–298, 2002.
- [108] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard. Unsupervised Learning of 3D Object Models from Partial Views. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan, 2009.
- [109] R. Rusu, A. Holzbach, R. Diankov, G. Bradski, and M. Beetz. Perception for mobile manipulation and grasping using active stereo. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 632–638, 2009.
- [110] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, May 12-17, 2009*.
- [111] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15 2009.
- [112] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D Point Cloud Based Object Maps for Household Environments. *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge in Robotics)*, 56(11):927–941, 30 November 2008.
- [113] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz. Laser-based Perception for Door and Handle Identification. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, Munich, June 22-26 2009. Best Paper Award.
- [114] R. B. Rusu, A. Sundaresan, B. Morisset, M. Agrawal, and M. Beetz. Leaving Flatland: Realtime 3D Stereo Semantic Reconstruction. In *Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA) 2008, October 15-17, Wuhan, China, 2008*.
- [115] E. Sali and A. Avraham. THREE-DIMENSIONAL MAPPING AND IMAGING, Apr. 12 2010. US Patent Application. 2010/0265316 A1.
- [116] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157, 2008.
- [117] A. Saxena, L. L. S. Wong, and A. Y. Ng. Learning grasp strategies with partial shape information. In *AAAI*, pages 1491–1494, 2008.

- [118] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.
- [119] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. 2007.
- [120] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2024, 2006.
- [121] E. Sisbot, A. Clodic, R. Alami, and M. Ransan. Supervision and motion planning for a mobile manipulator interacting with humans. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 327–334. ACM, 2008.
- [122] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. 2003.
- [123] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. C. Romea, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and J. M. Vandeweghe. Herb: a home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, January 2010.
- [124] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual slam for flying vehicles. *IEEE Transactions on Robotics*, 24(5):1088–1093, 2008.
- [125] C. Sun and J. Sherrah. 3d symmetry detection using the extended gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):164–169, February 1997.
- [126] L. Sun, U. Klank, and M. Beetz. Eyewatchme - 3d hand and object tracking for inside out activity analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 9–16, June 2009.
- [127] M. Tenorth and M. Beetz. KnowRob – Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, 2009.
- [128] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled Robots – Robots that Use the Web as an Information Resource. *Robotics & Automation Magazine*, 18(2):58–68, 2011.
- [129] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.
- [130] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

BIBLIOGRAPHY

- [131] R. Tsai, R. Lenz, I. Center, and Y. Heights. A new technique for fully autonomous and efficient 3D roboticshand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, 1989.
- [132] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008.
- [133] A. Ude, D. Omrcen, and G. Cheng. Making object recognition an active process. *International Journal of Humanoid Robotics (IJHR)*, 5(2), 2008.
- [134] M. Ulrich, C. Wiedemann, and C. Steger. CAD-Based Recognition of 3D Objects in Monocular Images. In *IEEE International Conference on Robotics and Automation*, pages 1191–1198, 2009.
- [135] M. Vahedi and A. F. van der Stappen. On the complexity of the set of three-finger caging grasps of convex polygons. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.
- [136] K. F. von Gleissenthal. *Internet based classification of daily life objects using a bag of visual features*. Bachelor thesis. Department of Computer Science. Technische Universität München, 2010.
- [137] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real Time Detection and Tracking for Augmented Reality on Mobile Phones. *IEEE Transactions on Visualization and Computer Graphics*, 2009.
- [138] E. Wahl, G. Hillenbrand, and G. Hirzinger. Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification. *Proceedings. Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM), 2003.*, pages 474–481, 2003.
- [139] A. Wallace, P. Csakany, G. Buller, A. Walker, and S. Edinburgh. 3D imaging of transparent objects. In *Proc. British Machine Vision Conf*, pages 466–475. Citeseer, 2000.
- [140] A. Webb. *Statistical pattern recognition*. A Hodder Arnold Publication, 1999.
- [141] K. Welke, T. Asfour, and R. Dillmann. Active multi-view object search on a humanoid head. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2009)*, pages 417–423, 2009.
- [142] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann. Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

- [143] C. Wiedemann, M. Ulrich, and C. Steger. Recognition and tracking of 3d objects. In G. Rigoll, editor, *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 132–141, Berlin, 2008. Springer-Verlag.
- [144] T. Wimbock, D. Nenchev, A. Albu-Schaffer, and G. Hirzinger. Experimental study on dynamic reactionless motions with DLR’s humanoid robot Justin. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5481–5486. IEEE, 2009.
- [145] K. Yamazaki, R. Ueda, S. Nozawa, Y. Mori, T. Maki, N. Hatao, K. Okada, and M. Inaba. A Demonstrative Research for Daily Assistive Robots on Tasks of Cleaning and Tidying up Rooms. In *First International Symposium on Quality of Life Technology*, 2009.
- [146] S. Yang and C. Wang. Dealing with laser scanner failure: Mirrors and windows. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, 2008.
- [147] M. Z. Zia, U. Klank, and M. Beetz. Acquisition of a Dense 3D Model Database for Robotic Vision. In *International Conference on Advanced Robotics (ICAR)*, 2009.

Index

- TUM-Rosie, 8, 11
- algorithm selection, 22
- AMCL, 46
- ARMAR-III, 39
- Armar-III, 26
- CAD, 26, 74, 101, 102, 111, 112
- calibration, 124
- cluster, 157
- clutter, 158
- contribution, 10
- CoP, 17
- descriptor 3D, 151
- Desire, 38
- environment mapping, 46
- feedback, 24, 25
- GMapping, 46
- grasp, 157
- HERB, 37
- HRP2, 39
- Justin, 38
- learning, 10, 97, 99–102, 114
- Located Objects, 129
- map, 23
- mapping, 45
- method, 13
- model, 13
- object classification, 82
- object recognition, 45
- obstacles, 158
- online calibration, 159
- pancake, 9, 159
- planar matching, 46, 99, 159
- plane detection, 47
- point cloud, 48
- PR2, 12, 41, 124
- scene localization, 46
- search space, 76
- segmentation, 48
- sensor, 13
- STAIR, 37
- surface matching, 80, 100
- surface model, planar matching, 165, 166
- system architecture, 11
- textured object, 88
- ToF, 33
- tool calibration, 159
- transparent, 10, 57, 154
- TUM-Rosie, 157, 159
- unmodeled objects, 157
- WordNet, 19
- World Wide Web, 10, 82, 101, 102