UNIVERSITÄT BREMEN

Fachbereich 3 (Mathematik und Informatik)

Combining Perception and Knowledge for Service Robotics

Dejan Pangercic

Dissertation

zur Erlangung des Grades eines Doktors der Ingenieurswissenschaften

Dr. Ing.

Vorgelegt im Fachbereich 3 (Mathematik und Informatik)

der Universität Bremen

im Januar 2014

Prüfer der Dissertation:

1. Univ.-Prof. Michael Beetz, PhD
2. Assoc.-Prof. Kei Okada, PhD

Abstract

As the deployment of robots is shifting away from the industrial settings towards public and private sectors, the robots will have to get equipped with enough knowledge that will let them perceive, comprehend and act skillfully in their new working environments. Unlike having a large degree of controlled environment variables characteristic for e.g. assembly lines, the robots active in shopping stores, museums or households will have to perform open-ended tasks and thus react to unforeseen events, self-monitor their activities, detect failures, recover from them and also learn and continuously update their knowledge.

In this thesis we present a set of tools and algorithms for acquisition, interpretation and reasoning about the environment models which enable the robots to act flexibly and skillfully in the afore mentioned environments. In particular our contributions beyond the state-of-the-art cover following four topics: a) semantic object maps which are the symbolic representations of indoor environments that robot can query for information, b) two algorithms for interactive segmentation of objects of daily use which enable the robots to recognise and grasp objects more robustly, c) an image point feature-based system for large scale object recognition, and finally, d) a system that combines statistical and logical knowledge for household domains and is able to answer queries such as “Which objects are currently missing on a breakfast table?”.

Common to all contributions is that they are all knowledge-enabled in that they either use robot knowledge bases or ground knowledge structures into the robot’s internal structures such as perception streams. Further, in all four cases we exploit the tight interplay between the robot’s perceptual, reasoning and action skills which we believe is the key enabler for robots to act in unstructured environments.

Most of the theoretical contributions of this thesis have also been implemented on TUM-James and TUM-Rosie robots and demonstrated to the spectators by having them perform various household chores. With those demonstrations we thoroughly validated the properties of the developed systems and showed the impossibility of having such tasks implemented without a knowledge-enabled backbone.

Zusammenfassung

Während sich der Einsatz von Robotern von Industrieanlagen in den privaten und öffentlichen Sektor verschiebt, werden diese Roboter mit genug Wissen ausgestattet werden müssen, damit sie ihre Umgebung richtig wahrnehmen und interpretieren können und angemessen agieren. Statt in bekannten und kontrollierbaren Umgebungen wie Montageanlagen werden sich Roboter in Kaufhäusern, Museen und Haushalten zu recht finden müssen. Dort müssen sie in der Lage sein, mit offenen Aufgabenstellungen umzugehen und auf unvorhergesehene Ereignisse zu reagieren. Sie müssen sich selbst überwachen, um Fehlerzustände zu erkennen und zu beheben, und darüber hinaus die Fähigkeit zum kontinuierlichen Lernen besitzen.

In dieser Doktorarbeit präsentieren wir eine Reihe von Werkzeugen und Algorithmen für Akquisition, Interpretation und Reasoning über Umgebungsmodelle, die einem Roboter ermöglichen, in den genannten Umgebungen auf flexible und kompetente Weise tätig zu sein. Insbesondere gliedert sich unser Beitrag zu dem Stand der Technik in vier Teile: a) semantische Karten, die eine symbolische Repräsentation von Innenraumumgebungen sind und die der Roboter als Informationsquelle nutzen kann, b) zwei Algorithmen für die interaktive Segmentierung von Alltagsgegenständen, die dem Roboter erlauben, diese Objekte richtig zu erkennen und zu greifen, c) ein auf Punkt-Features basierendes System für die Erkennung einer großen Anzahl von Objekten und schließlich d) ein System, das logisches und statistisches Wissen kombiniert, um beispielsweise zu erkennen, welche Lebensmittel oder Besteckteile auf einem Frühstückstisch noch fehlen.

Alle vier Beiträge sind wissensbasiert in dem Sinn, dass sie entweder auf eine Roboterwissensdatenbank zugreifen oder Wissensrepräsentationen in den Datenstrukturen des Roboters, wie etwa Perzeptionsdaten, grundieren. Weiterhin sind in allen vier Ansätzen die drei essenziellen Kernkomponenten eines Roboters eng gekoppelt:

Wahrnehmung, Reasoning und Aktion. Wir betrachten dies als Grundvoraussetzung um Roboter in den Alltag zu bringen.

Fast alle theoretischen Ansätze wurden auf den beiden Robotern TUM-James und TUM-Rosie implementiert und vor Zuschauern demonstriert. Die Roboter bearbeiten in den Vorführungen Haushaltsaufgaben und validieren dadurch die Eigenschaften des in dieser Doktorarbeit entwickelten Systems. Es zeigt sich, dass die Lösung solcher Aufgaben für Roboter ohne umfangreiches Wissen nicht möglich ist.

Acknowledgments

The list of people that made this thesis possible is long and I am incredibly grateful that I was able to work with and to befriend so many great and smart people.

Without further ado, I would like to thank Michael and Radu for bringing me into the Intelligent Autonomous Systems (IAS) group which means that they believed in me even in the times when I had no proven record. Michael, working with you was truly amazing and unique in every aspect possible and I can only wish for such a mentor, boss and friend again. *Vielen, vielen Dank.*

Thomas at the beginning and then Zoli, Nico and Mihai were my best office mates. Thank you guys for so many fruitful discussions and all these laughters. Working in IAS group always felt like working in one big, symbiotic family. Luci, Ingo, Lars, Christoph, Freek, Daniel, Karinne, Alexandra, Lorenz, David W., David G., Georg, Gheorghe, Alexis, Dominik, Uli, Zahid, Fede, Sorin and Francisco, thank you very much for creating such a fantastic atmosphere. Special thanks goes to Moritz, Zoli and Jan. First two for being my closest collaborators and for helping me advance the scientist in me, and to Jan for taking me under his wing in my greenest research times. All three also became (and remained) my very good friends which I am especially proud of. Nothing would have been possible without a tremendous support of our admin staff – Doris, Sabine and Quirin, I will owe you one for ever. If I had to single out one thing during my PhD, then working with undergraduate students and building up talents would be the one. I had a real pleasure to work with the pool of unbelievable prospects. Fadri, Kai, Andy, Vlad, Ross, Karol, Niko, Ronny, Nacer, Rim, Julius, Martin, Monica, Shulei, Florian, Vlado, Hozefa, Andreas, Rok and Tom, you were the best and I am really proud of how you are doing today.

Working in IAS group made me really international and enabled me to meet and later also work with the brightest minds from all over the world. Kyle, Oscar, Asako, Jürgen, Kei, Wolfram, Daniel, Gajan, Andrzej, Matei, Vijay, Brian, both Stefans, Ryohei, Dubi, Kurt are just some that I can recall from the top of my head. Having such collaborators all around the world only reminds me how lucky I actually am.

Dr. Uwe Haass and CoTeSys provided us with funds and fantastic robots and infrastructure – something without which my research would have never been possible. Willow Garage revolutionized the field of service robotics with ROS and PR2 robots – thank you very much for making our lives so much easier.

After leaving IAS I joined Bosch RTC in Palo Alto and started working with just as smart and amazing people as I had before. Ben, Chris, Sarah, Phil, Jan, Matthias and Kaijen, it was an unbelievable year and it seems that nothing can really stop us.

Elena M., Elena D., Barbi, Luka, Ninchy, Outi, Emmi, Flo, Sandi, Nudzejma were always biggest fans of my work and friends that I could count on on any given day. EESTEC gave me some of the most fond memories of my undergraduate studies and EESTEC people (Borut, Marko, Petko, Ovidiu, Clee just to name a few) and EESTEC spirit have stuck with me during my PhD and apparently will forever.

My father Peter and mum Milena bear the ultimate ‘guilt‘ for me to be here. Thank you so much for instilling this relentless work ethic in me. It has worked out always. *Hvala lepa.* Andrej is my brother and someone I have an honor to mentor and guide through his life. *Mali*, I am really proud of how and what you do, keep going.

Lastly, thank you dear robot gods for bringing my dear angel from Portugal in my life. Diana is my true soul mate and someone that shows me that life is not just about the work. *Eu amo-te moja moja.*

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The robots are moving away from the industrial settings where they operate in cages, are pre-programmed and the code re-usability is in general not an issue anymore. According to the World Robotics report 2011 [Haegele, 2011] more than 13.700 service robots for professional use were sold in 2010 boosting this number up 4% from the year 2009. Furthermore, for the period 2011 to 2014, sales are forecast to about 87.500 professional service robots in total and a strong growing sector will be the mobile platforms in general use. Service robot suppliers estimate that about 12.000 mobile platforms in general use will be sold in the given period. Additionally, investements of large companies like Google, Amazon, and Bosch show real business case viability for service robots and accelerate the process of commercialization. The latter is very encouraging and supports in 1988 proclaimed breakthrough of the service robots by the father of service robotics, Joseph F. Engelberger.

According to Prats [2009], a service robot is a robot "which operates partially or fully autonomously to provide services useful to the humans". Because of their multitude of forms and structures as well as application areas, service robot subcategories are not easy to define [Haegele, 2011]. Within the scope of this thesis we will however consider mobile platforms equipped with two human-like arms and grippers (such as e.g. Personal Robot 2 [PR2]) and we will call them *personal robots*. Their main purpose is to help elderly and impaired people in their daily life.

1.1 Motivation

According to the VDE report [Eberhardt, 2012] the share of people age 80 and over will in Germany increase from 5% in 2008 to 14% in 2060. As life expectancy increases, so does the chance of people becoming physically and mentally limited, often increasing the demand for care services. While on the one hand the number of people requiring care is steadily increasing, on the other hand the number of caregivers is decreasing due to factors such as decline in population, decline in infrastructure, etc. [Eberhardt, 2012]. This societal changes force us to develop new concepts, including technologies such as *personal robots*, for promoting independent living. Prolonging the independence of elderly people with minor disabilities and increasing their participation in daily life is expected to improve the well-being and the health-state of these people and thereby also mitigate the care-giving problem. The example set of tasks that people want the personal robots to perform include preparing drinks and food, reaching for books and other objects from a shelf, plugging things, loading a video, watering plants and other gardening tasks, getting items from the refrigerator, turn knobs, opening/closing doors and drawers, turning appliances on and off, operating light switches, shopping for groceries, etc [Prats, 2009; Mitzner et al., 2011]. These open-ended tasks must be on the one hand executed in a variety of unstructured environments, according to the personal preferences of human companions, at the right times and in the right situations and on the other hand they must still be performed robustly, repeatedly and with the "style".

1.2 Contributions

To tackle above challenges we in this thesis develop two important contributions: On the one hand we investigate building of the knowledge-enabled perception systems that enable these personal robots to carry out open ended tasks for elderly and handicapped people in their domestic environments (e.g. households, elderly homes, etc.). Given such challenging tasks and environments, we on the other hand deliberately investigate tight integration of robot's perception, knowledge and action skills – a key enabler for robust and stereo-typical performance of such robots.

In Section 1.3 we present a use case where the personal robot is tasked with the preparation of pancakes for breakfast. We will use this example to motivate the work presented herein, to expose the relevant research questions, to show the collaborations and the placement of our work with respect to the rest of the related work, but also to refer to it as a reader's guide throughout the whole thesis. In order for the robots to competently tackle such tasks they have to be able to generate the knowledge about their working environment and they have to involve all their skills including perception, reasoning and manipulation. Only then will they be able to answer and execute the following types of queries:

- What is the semantic of my environment (e.g. How do I find the kitchen)?
- How do I navigate around my environment?
- How do I find, open and close the refrigerator?
- How do I find the table and set it up for breakfast?
- Which and how many objects are necessary to set up the table?
- Where do I find needed objects (e.g. a milk, pancake mix, plates, a cutlery, etc.)?
- How do I pick and place objects?
- etc.

The realization of a personal robot capable of answering above queries requires us not to only equip the robots with the necessary perceptual capabilities but also to abstract away these percepts into a sufficiently rich knowledge representation. Doing so enables us not to only be able detect, recognize, localize, and geometrically reconstruct the objects in their environments, but it also brings us means to interpret the perception results in the context of the actions and activities that the robots perform. Albeit a very challenging problem, the robots do not have to solve it everyday anew. Because they are to perform many activities on a regular basis, because they are to be deployed once and then operate in the same environment for the years to come and because the environment is to the larger extent kept stable, the robots can turn this into their advantage. For example, in order to get a pancake mix out

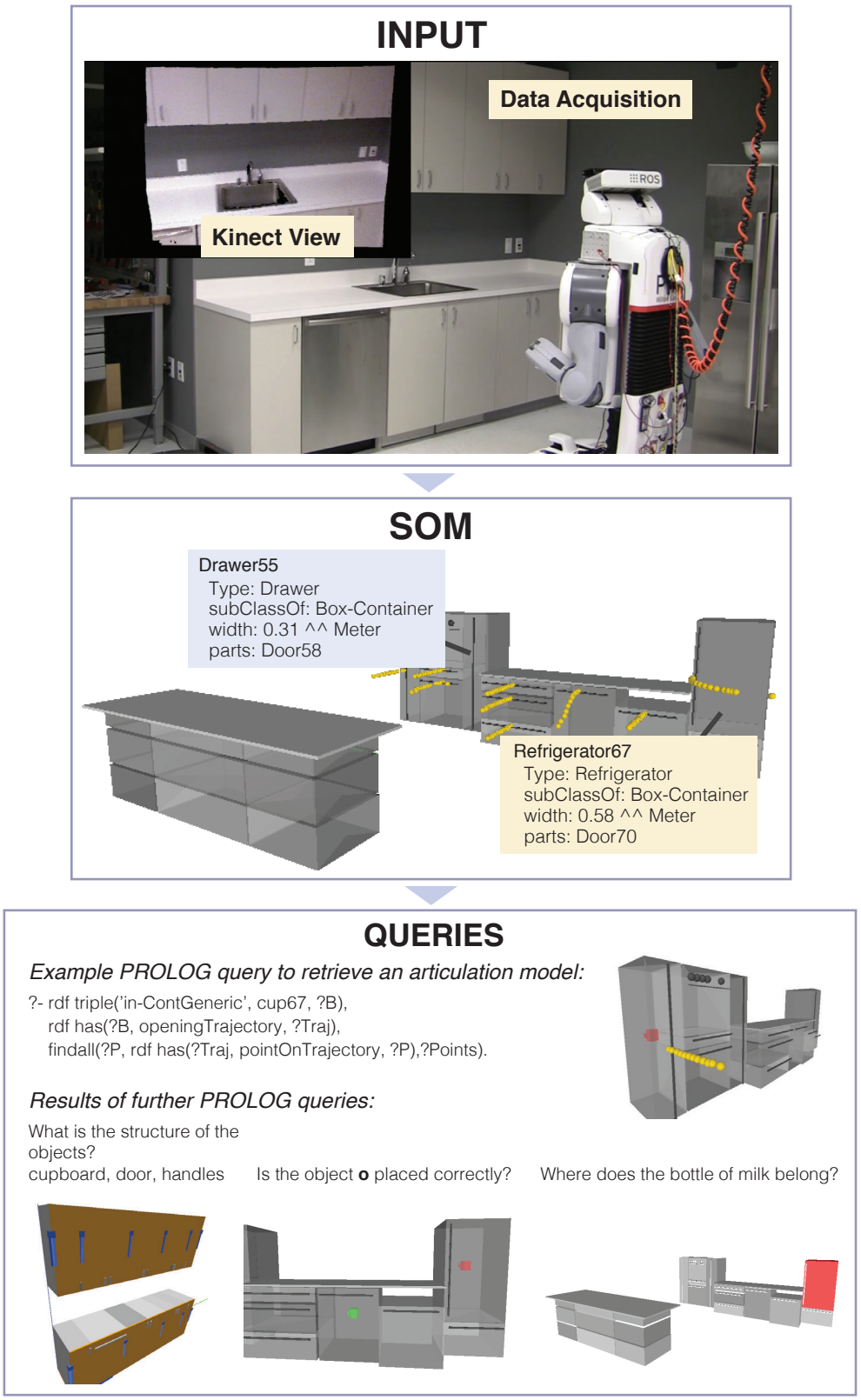


Figure 1.1: *Building of a SOM$^+$ map in a kitchen environment (top), SOM$^+$ map representation (middle) and a set of robot queries made possible due to such powerful representation (bottom).*

of the refrigerator, the robot has to learn about the location of the refrigerator only once, after-wards it inserts the refrigerator together with its semantic attributes such as e.g. type and function into its enviroment model and can re-use this knowledge in the later runs. Environment models thus serve as important resources for an autonomous robot by providing it with the necessary task-relevant information about its habitat. Robots can make use of environment models such that they perform their tasks more reliably, flexibly, and efficiently. To function efficiently, it is imperative that these environment models are acquired mostly autonomously and therefore support the deployment of mobile robots in new environments, without requiring too many software updates or (much of) manual user input.

We present a set of tools and algorithms for acquisition, interpretation and reasoning about the environment models which enable the robots to act flexibly and skillfully in the indoor kitchen environments. In particular our contributions beyond the state-of-the-art are the following:

- **Semantic Object Maps (SOM$^+$)**. These maps serve as information resources for autonomous service robots performing everyday manipulation tasks in kitchen environments. They provide the robot with information about its environment that enable it to perform fetch and place tasks more efficiently and reliably. To this end, the semantic object maps can answer queries such as the following ones: “What do parts of the kitchen look like?”, “How can a container be opened and closed?”, “Where do objects of daily use belong?”, “What is inside of cupboards/drawers?”, etc.

- **Algorithms for Interactive Segmentation of Textured and Textureless Objects**. We explore the robot’s ability to interact with the environment and design two novel object segmentation algorithms. The proposed system allows a robot to effectively segment textured and textureless objects in cluttered scenes by leveraging its manipulation capabilities. In this interactive perception approach, 2D or 3D features are tracked while the robot actively induces motions into a scene using its arm. The robot autonomously infers appropriate arm movements which can effectively separate objects. The resulting tracked feature trajectories are assigned to their corresponding object by using novel clustering algorithms, which sample rigid motion hypotheses for the *a priori* unknown number of scene objects.

- **Objects of Daily Use Finder (ODUfinder)**. We realize an *ODUfinder*, a robot perception system for autonomous service robots acting in human living environments. The perception system enables the robots to capture appearances and retrieve semantic types of textured objects of daily use and to then detect and recognize these sets of objects in the arbitrary scenes. Efficiency, robustness, and a high detection rate are achieved through the combination of modern text retrieval methods that are successfully used for indexing huge sets of web pages and state-of-the-art robot vision methods for object recognition. The barcodes are used to query a product information website, in order to retrieve the semantic types of the objects.

- **Knowledge-enabled Scene Understanding**. Recognition results from the above perception algorithms are used to generate symbolic representations of perceived objects and scenes and to infer answers to complex queries that require the combination of perception and knowledge processing. Using such system, called K-CoPMan (Knowledge-enabled Cognitive Perception for Manipulation), the robot can solve inference tasks such as identifying items that are likely to be missing on a breakfast table. In essence we use Bayesian Logic Networks (BLN), a formalism that combines statistical knowledge (in fragments representing conditional probability distributions) with logical knowledge (sentences in first-order logic). Key features of this system are that it can make a robot environment-aware and that it supports goal-directed as well as passive perceptual processing.

Our approach is holistic from the point of view of the autonomous robot and begins from the raw sensory data, through categorization and recognition of coarse objects and objects of daily use, over to the abstract and hierarchical representation of parts of the environment and finally, up to the statistical and logical knowledge that enable reasoning about the scenes and situations. Generally put, such system is an information resource for the robot, which informs the robot with respect to what to do, to which object, and in which way.

The set of problems that we do *not* consider in this thesis entails to: dynamics (e.g. moving human companions), large scale rooms, large pose uncertainties and perceptual anchoring. However we believe that the devised foundations and the developed

integrated system will provide an excellent starting point to elevate this research even further.

1.3 Example Use Case

Throughout the thesis we will use a realistic example of the robot being tasked with the "Prepare the breakfast with pancakes" task in order to show the research challenges investigated in this thesis, the solutions proposed, the applicability of our work using real robots[1]), the integration with respect to the related work and finally, we will use it as a reader's guide throughout this thesis. The given task was publicly demonstrated in front of a large crowd, including several world-renowned roboticists, in one of the CoTeSys workshops as reported in detail in Chapter 8.

Figure 1.2: *TUM-Rosie preparing pancakes.*

[1]http://ias.cs.tum.edu/robots

The whole task can, without the loss of generality, be broken up into the following set of subtasks. In the following we will present the challenges and our solutions and discuss assumptions and integration of the related work for the cases that fall beyond the scope of this thesis. The paragraphs with the underlined titles denote our contributions to the use case.

Receiving Task "Prepare a breakfast with pancakes" for Michael. The instruction for this task is communicated to the robot through a spoken dialog [Google], user interface dialog, etc. In our case we focused on the limited set of up to 2 tasks that the robot was able to decipher and process unambiguously. In order to generate the plan for the breakfast with pancakes task this means an automatic mapping to the generation of the execution plan from the WikiHow[2] instructions. For the remainder of the thesis we will assume that the task instructions are already transcribed in a form of a semi structured natural language as is the case for http://www.wikihow.com.

Generating Execution Plans from Internet Instructions. World Wide Web (WWW) is a large resource of everyday, commonsense knowledge. The websites such as EHow, WikiHow, WordNet, Cyc, etc. provide a bulk of structured knowledge about how to perform everyday tasks, semantic relations about the objects and the actions and so on. They were however written in natural language and thus for humans which means that they assume a great deal of implicit knowledge. For example, a certain instruction might state that in order to cook something the oven shall be turned on, but it is never explicitly stated that the oven shall be turned off at the end. Tenorth [2011]; Beetz et al. [2011, 2012] provide a partial solution to this problem in that they first expand their knowledge representation and processing framework KnowRob with the processed encyclopedic knowledge and secondly convert the latter into an execution plan. The result of this conversion are the needed objects and other kinds of entities and the action steps specified as the declarative goal statements in the plan executive language CRAM [Beetz et al., 2010]. In this thesis we primarily deal with two instructions: Make-Pancakes-Using-Mondamin-Pancake-Mix[3] and Set-

[2]http://www.wikihow.com
[3]http://www.wikihow.com/Make-Pancakes-Using-Mondamin-Pancake-Mix

a-Table[4]. The processing of both into the execution plans is discussed in [Beetz et al., 2011] and [Pangercic et al., 2009] respectively.

Finding the Kitchen. After the robot has resolved to the list of needed objects and actions to perform the given task, it proceeds on to find them. However, since in this thesis we assume the operation in the kitchen domain, the first task for the robot is to resolve the room categorization problem, the problem herein referred to as a global semantic map. For this we envision combining our work with the one proposed by Pronobis [2011] which links spatial concepts to sensory information originating from multiple modalities such as vision and laser range data. Their system is capable of incorporating semantic information extracted from such sources as the geometry and general appearance of places, presence of objects, topology of the environment and/or human input, etc. and in the end correctly categorize room types.

Finding the Furniture. Once in the kitchen, the robot shall make use of its local semantic map SOM^{+} , which we build autonomously and represent as symbolic knowledge bases in KnowRob. KnowRob contains facts about objects in the environment and that link objects to data structures such as poses, dimensions, appearance models, etc. In the acquisition step, as discussed in Chapter 4, we use a low-cost, low-accuracy Kinect sensor and advance the state-of-the-art registration, reconstruction and interpretation algorithms in order to generate maps with the sufficient quality to be later on used by the robot during its manipulation task. While the annotation of cupboards and drawers is currently done manually, the tables and chairs are categorized using matching of furniture CAD models from the web catalogs in a probabilistic Hough voting [Mozos et al., 2011] schema.

Finding the Objects of Daily Use. In order to find the needed objects (e.g. a pancake mix, a cutlery, dishware) we employ a library of specialized perception routines that solve different, well-defined perceptual sub-tasks and can be combined into composite perceptual activities including the construction of an object model database, multi-modal object classification, and object model reconstruction for grasping. In particular we developed algorithms that can can either categorize the objects by shape or recognize them by their visual appearance. For the former we together with Marton et al. [2011] and Kanezaki et al. [2011] developed two types of the

[4]http://www.wikihow.com/Set-a-Table

global shape descriptors and for the latter we developed SIFT-based and barcode-based algorithms that in particular scale very well for the large number of objects. All algorithms are tightly integrated into the KnowRob which extends the set of robot's task by allowing to perform reasoning and inferences upon the perceived objects. Lastly, the perception system gains its strengths by exploiting the knowledge from the above mentioned SOM$^+$s and by detecting and incrementally adding new object models in the course of robot's life cycle.

Localization and Navigation. Often items are not in stock in human kitchens. To illustrate that our robot can cope with this type of situations we assume that the robot was not able to find a pancake mix in the refrigerator. We have to retreat to the backup solutions which in this case will have our robot go shopping for the pancake mix. We integrated an approach by Saito et al. [2011] who use Open Mind Indoor Common Sense (OMICS) [Gupta and Kochenderfer, 2004] in order to describe the objects and their typical locations outside of our kitchen domain. In this particular case the second most likely location for a pancake mix was a grocery store. In order to navigate the robot from our assistive kitchen lab to the grocery store across the street we use a stock Adaptive Monte Carlo localization [Fox, 2001] and path planning software available in Robot Operating System[5]. Once in the grocery store, we propose an inclusion of the work by Joho et al. [2011] to rapidly approach the vicinity of the sought-after object, pancake mix in this case. Their search algorithm is based on the learning from object arrangements of example environments using a maximal entropy model.

Shopping. Once in the vicinity of the shopping rack we need to recognize the pancake mix and put it in the shopping basket. While the recognition of the pancakes mix is done using a combination of SIFT features and a Vocabulary Tree matcher, the problem is in that the objects are tightly cluttered together on the shopping rack. In order to generate the region of interest hypotheses we thus developed an interactive segmentation approach that lets the robot interact with the environment and segment the objects based on the analysis of the movement of extracted and tracked 2D and 3D features [Bersch et al., 2012]. Finally, we also demonstrate a bi-manual pick-and-place action in constraint spaces [Pangercic et al., 2011b].

[5] www.ros.org

Making of Pancakes. Making pancakes requires manipulation actions with effects that go far beyond the effects of normal pick-and-place tasks in terms of complexity. For instance, in the process the robot might encounter the following challenges: the robot must pour the right amount of pancake mix onto the center of the pancake maker, and monitor the action success to forestall undesired effects such as spilling the pancake mix. It must handle the spatula exactly enough to push it under the pancake for flipping it. This requires the robot to select the appropriate force in order to push the spatula just strong enough to get under the pancake, but not too strong to avoid pushing it off the pancake maker. The list of issues is in fact ongoing and we kindly refer an astute reader to the original publication [Beetz et al., 2011] for further details.

Setting a Table. To complete the task the robot is to set the table according to the processed and converted WikiHow instruction [6]. This task amounts to the robot finding the dining table, inferring which objects are already present and which are still missing. We use the above mentioned system for recognition of objects of daily us in order to correctly identify the objects on the table and then apply Bayesian Logic Networks (BLN) [Jain et al., 2009], a formalism that combines statistical knowledge (in fragments representing conditional probability distributions) with logical knowledge (sentences in first-order logic) to infer the missing objects. Once the robot knows which objects it needs and how they look like, it has to find them in the kitchen environment [Schuster et al., 2012] and grasp them [Witzig et al., 2013]. We use ontologies to semi-automatically link the objects to their possible locations in the kitchen and thus enable a much faster and reliable object search. To execute the right grasp on the right object we use probabilistic graphical methods and human in the loop approach.

[6] http://www.wikihow.com/Set-a-Table

1.4 Outline

The rest of the thesis is organized as follows.

Chapter 3: System Setup

After we motivated the general case, we will present the wide corpus of systems and tools supporting and enabling this thesis.

Chapter 4: Semantic Object Maps

Discusses and thoroughly presents the representation language, acquisition and interpretation algorithms to generate the SOM$^+$s for domestic environments.

Chapter 5: Interactive Segmentation of Textured and Textureless Objects

We explain how we solve a rather challenging problem of segmenting the objects of daily use in heavily cluttered tabletops and shopping shelves. This work falls in the area of research called interactive perception.

Chapter 6: Knowledge-linked Object Recognition

Describes a combined 2D-3D categorization and classification of objects of daily use and the generation of their abstract representations.

Chapter 7: Knowledge-enabled Scene Understanding

The outcomes of the **Chapter 4**, **Chapter 5** and **Chapter 6** are used in this chapter where we show how to effectively combine perception, knowledge representation and statistical learning in order to classify scenes and situations from everyday life.

Chapter 8: Demonstrations

To show the generality and scalability of herein proposed and developed algorithms we have run several public demonstrations about which we briefly report in this chapter.

Chapter 9: Conclusion

Finally we will conclude and give the directions for the future work.

Chapter 2

List of Prior Publications

The work presented in this document is partly based on prior publications. Sections of this work that drew upon content from prior publications cited the respective publications where appropriate. A complete list of publications that were (co-)authored during my research as a doctoral candidate is provided below.

Journal Articles

Zoltan Csaba Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. Combined 2D-3D Categorization and Classification for Multimodal Perception Systems. *The International Journal of Robotics Research*, 30(11):1378–1402, 2011.

Moritz Tenorth, Ulrich Klank, Dejan Pangercic, and Michael Beetz. Web-enabled Robots – Robots that Use the Web as an Information Resource. *Robotics & Automation Magazine*, 18 (2):58–68, 2011.

Michael Beetz, Dominik Jain, Lorenz Mösenlechner, Moritz Tenorth, Lars Kunze, Nico Blodow, and Dejan Pangercic. Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence. *Proceedings of the IEEE, Special Issue on Quality of Life Technology*, 100(8):2454–2471, 2012.

Conference Papers

Thomas Witzig, J. Marius Zöllner, Dejan Pangercic, Sarah Osentoski, Philip Roan, Rainer Jäkel, and Rüdiger Dillmann. Context Aware Shared Autonomy for Robotic Manipulation

Tasks. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo Big Sight, Japan*, 2013.

Karol Hausman, Ferenc Balint-Benczedi, Dejan Pangercic, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Tracking-based Interactive Segmentation of Textureless Objects. In *In IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, 2013.

Dejan Pangercic, Moritz Tenorth, Benjamin Pitzer, and Michael Beetz. Semantic Object Maps for Robotic Housework - Representation, Acquisition and Use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Thomas Rühr, Jürgen Sturm, Dejan Pangercic, Michael Beetz, and Daniel Cremers. A Generalized Framework for Opening Doors and Drawers in Kitchen Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

Michael Beetz, Moritz Tenorth, Dejan Pangercic, and Benjamin Pitzer. Semantic Object Maps for Household Tasks. In *5th International Conference on Cognitive Systems (CogSys 2012)*, 2012.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic Roommates Making Pancakes. In *11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

Nico Blodow, Lucian Cosmin Goron, Zoltan-Csaba Marton, Dejan Pangercic, Thomas Rühr, Moritz Tenorth, and Michael Beetz. Autonomous Semantic Mapping for Robots Performing Everyday Manipulation Tasks in Kitchen Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

Shulei Zhu, Dejan Pangercic, and Michael Beetz. Contracting Curve Density Algorithm for Applications in Personal Robotics. In *11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

Dejan Pangercic, Moritz Tenorth, Dominik Jain, and Michael Beetz. Combining Perception and Knowledge Processing for Everyday Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1065–1071, 2010.

Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3D Modelling of Novel Objects from a Single View. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010a.

Zoltan-Csaba Marton, Dejan Pangercic, Radu Bogdan Rusu, Andreas Holzbach, and Michael Beetz. Hierarchical Object Geometric Categorization and Appearance Classification for Mobile Manipulation. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2010b.

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. Real-time CAD Model Matching for Mobile Manipulation and Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 290–296, 2009.

Dejan Pangercic, Rok Tavcar, Moritz Tenorth, and Michael Beetz. Visual Scene Detection and Interpretation using Encyclopedic Knowledge and Formal Description Logic. In *Proceedings of the International Conference on Advanced Robotics (ICAR).*, 2009.

Workshop Papers

Christian Bersch, Dejan Pangercic, Sarah Osentoski, Karol Hausman, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Segmentation of Textured and Textureless Objects through Interactive Perception. In *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012.

Ross Kidson, Darko Stanimirovic, Dejan Pangercic, and Michael Beetz. Elaborative Evaluation of RGB-D based Point Cloud Registration for Personal Robots. In *ICRA 2012 Workshop on Semantic Perception and Mapping for Knowledge-enabled Service Robotics*, 2012.

Zoltan-Csaba Marton, Dejan Pangercic, and Michael Beetz. Efficient Surface and Feature Estimation in RGBD. In *RGB-D Workshop on 3D Perception in Robotics at the European Robotics (euRobotics) Forum*, 2011.

Asako Kanezaki, Zoltan-Csaba Marton, Dejan Pangercic, Tatsuya Harada, Yasuo Kuniyoshi, and Michael Beetz. Voxelized Shape and Color Histograms for RGB-D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, 2011.

Dejan Pangercic, Vladimir Haltakov, and Michael Beetz. Fast and Robust Object Detection in Household Environments Using Vocabulary Trees with SIFT Descriptors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, 2011.

Michael Beetz, Ulrich Klank, Alexis Maldonado, Dejan Pangercic, and Thomas Rühr. Robotic Roommates Making Pancakes - Look Into Perception-Manipulation Loop. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Mobile Manipulation: Integrating Perception and Manipulation*, pages 529–536, 2011.

Michael Beetz, Nico Blodow, Ulrich Klank, Zoltan Csaba Marton, Dejan Pangercic, and Radu Bogdan Rusu. CoP-Man – Perception for Mobile Pick-and-Place in Human Living Environments. In *Proceedings of the 22nd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on Semantic Perception for Mobile Manipulation*, 2009. Invited paper.

Chapter 3

System Setup

In this chapter we briefly present and motivate the reasoning behind the selected tools, testbed, robots and datasets that this thesis makes use of.

Personal robots are very complex systems requiring the knowledge that spans from knowing how the hardware components are built and function, how they interface to the software level, how to process and interpret the raw data, how to make use of the data in order to act sensibly in the given context and all the way to knowing how to tie all of this algorithms and actions together and possibly debug and analyze failure cases. To be able to focus on the objectives laid out in Chapter 1 we opt out to study problems commonly encountered in the assistive kitchen laboratory. We use a standard PR2 robot [PR2] and an in-house built TUM-Rosie robot, with the stock available drivers and basic algorithms (e.g. navigation), we write our algorithms in meanwhile de facto standard framework ROS (Robot Operating System) [Quigley et al., 2009] and finally use open source libraries such as openCV (Open Source Computer Vision) [Bradski, 2000], PCL (Point Cloud Library) [Rusu and Cousins, 2011] and KnowRob (Knowledge processing for autonomous Robots) [Tenorth and Beetz, 2009]. This allows us to on the one hand leverage the basic algorithms and on the other hand contribute our findings back to the research community. We take the importance of the latter utmost serious as it does not only help the robotics community and keeps re-inventing of the wheel syndrom low, but it also provides us with the valuable and critical feedback on the usability of herein proposed approaches.

3.1 The Assistive Kitchen Laboratory

The majority of the work presented in this thesis has been developed and tested in the *Assistive Kitchen* testbed [Beetz et al., 2008] located in the Central CoTeSys[1] Robotic Laboratory II. The assistive kitchen is a ubiquitous computing, sensing, and actuation environment with robotic assistants (see Figure 1.1).

This assistive kitchen includes two personal robots that are to learn and perform complex household chores. The robots shall either perform housework alone (as assumed within the scope of this thesis) or jointly with human companions. Having such testbeds is important for several reasons. They are realistic environments and much less structured than a normal industrial environment. Yet they are common to humans and to the degree well "understood". Assistive kitchens also raise challenging research problems. One of these problems is that performing household chores is a form of everyday activity that requires extensive commonsense knowledge and reasoning. Another challenge is the low frequency of daily activities, which requires embedded systems and robotic agents to learn from very scarce experience. Besides, household chores include a large variety of manipulation actions and composed activities that pose hard research questions for current mobile manipulation research.

On the flip side these kind of kitchens are becomming standard in research labs around the world which is an important prerequisite for sharing of research findings and knowledge about this domain. Further, a vast palette of information resources (e.g. websites and video portals with cooking instructions, websites with general knowledge bases and commonsense reasoning engines) allows for an integration of a large deal of useful assumptions and hypotheses. Finally, an instrumentation of assistive kitchens with e.g. high fidelity motion tracking systems allows for a repetitive task execution and comparison of results against the ground truth.

3.1.1 Hardware Infrastructure

The hardware infrastructure of the kitchen consists of mobile robots and networked sensing and actuation devices that are physically embedded into the environment.

[1] Cognition for Technical Systems project, www.cotesys.org

The latter incorporates global environment sensors, sensor-equipped furniture, web-enabled appliances, and “smart” objects.

Global Sensors of the Kitchen. We have mounted a set of static off-the-shelf cameras positioned to cover the kitchen area with high resolution in critical working areas. With these cameras, actions of the people and robots can be tracked from different locations to allow for more accurate positioning and pose estimation.

Sensor-equipped Furniture. The entities of furniture in the assistive kitchen are also equipped with various kinds of sensors. For example, we have cupboards with long-range RFID tag readers that enable the cupboards to “know” the identities of the RFID tagged objects that are currently in the cupboard. Additionally, the cupboards are equipped with magnetic contact sensors that sense whether the cupboard doors are open or closed. Another example is a table which contains several integrated capacitive sensors as well as short-range RFID readers. The capacitive sensors report the capacitance of different areas on the table, when an object is placed there, while the RFID readers provide exact information on what object was placed there.

Web-enabled Kitchen Appliances. such as the refrigerator, the oven, the microwave, and the faucet, allow for remote and wireless monitoring and control.

"**Smart**" **Objects.** In addition, kitchen utensils, tools and small appliances are equipped with integrated sensors such as RFID tags.

Note that in this thesis we only used robots’ on-board sensors, instrumented parts of the kitchen were used in different studies and demonstrations. Actual experiments were carried out in 6 different kitchens, 2 located at the TUM and 4 at the Bosch Research and Technology center in Palo Alto, CA.

3.1.2 Robots

We performed the experiments with two similar personal robots depicted in Figure 3.1. TUM-James is a PR2 robot [PR2] with the holonomic base and two back-drivable 7 DOF arms with jaw grippers. In order to perceive its environment TUM-James uses 2 Hokuyo UTM 30LX laser range finders (one used for navigation and

one for 3D model acquisition), a narrow stereo and a wide stereo camera set, high-resolution Prosilica camera, 2 forearm cameras and a head-mounted Kinect. The latter sensor became our de facto sensor since October 2010. In order to generate organized point clouds prior to Kinect's release, the PR2 used a visible infrared light projector. TUM-Rosie is a robot with the Kuka omnidirectional base and 2 Kuka

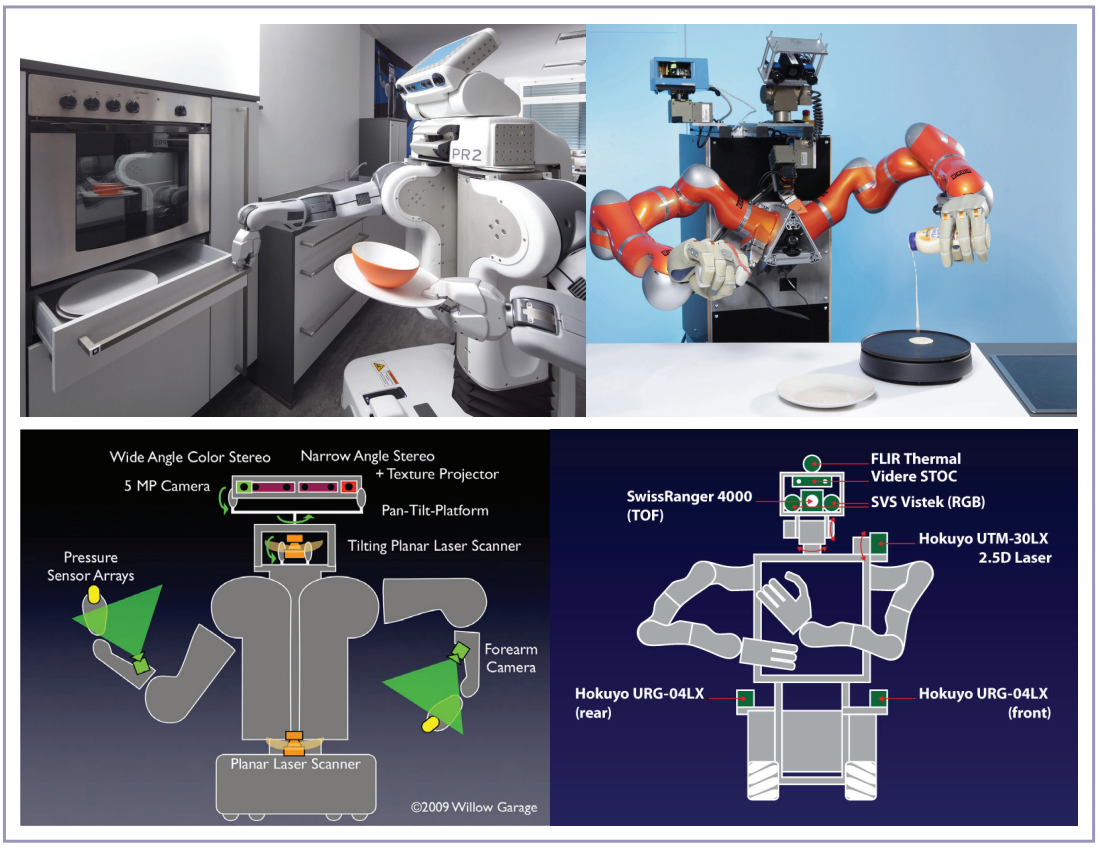


Figure 3.1: *Personal robots TUM-James (left) and TUM-Rosie (right).*

lightweight and compliant 7 DOF arms with the DLR-HIT hands. TUM-Rosie features 3 Hokuyo UTM 30LX laser range finders (two for navigation and one for 3D model acquisition), a stereo camera set, a thermal camera, an infrared depth sensor SwissRanger SR4000 and a Kinect. Both robots are equipped with powerful on-board computers and run ROS. While TUM-James has been mostly used for true mobile manipulation tasks (such as opening drawers in a coordinated arm-base fashion, etc.), TUM-Rosie has been on the other hand mostly used for very dexterous pick and place tasks (such as pouring pancake mix, making of sandwiches, etc.). The algorithms and

methods developed within the scope of this thesis have been applied to and are used by both robots (see Chapter 8).

3.2 Tools

Selecting the right tool for the right job is often of a crucial importance. Since the work proposed in this thesis finds itself mostly at the intersection between the perception and the knowledge processing for personal robots we needed the right tools for all three "departments". We thus selected openCV and PCL libraries for the perception problems, partly because they offer a great deal of openly available fundamental algorithms. In addition, PCL also started in our lab and provides us with the opportunity to conveniently push our algorithms back to the community. For the knowledge processing our natural choice was to select KnowRob by Moritz Tenorth, our labmate and one of the most frequent collaborators. KnowRob provides, to the best of our knowledge, the largest domain specification language for household robotics in the world. Finally, we use ROS as one the largest and most widespread frameworks for robots which we adopted very early in its emerging and which we also helped co-shape to the large extend. In the following we will briefly summarize each of the components and expose the main features used in this thesis work.

3.2.1 Robot Operating System

ROS is an open-source, meta-operating system for robots. It provides the services that normally make up an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. In general ROS ecosystem consists of three main parts: ROS file system, ROS computation graph and ROS community. While the first one defines how ROS stacks, packages and configuration files shall be laid out, the second one defines a peer-to-peer network of ROS processes that are processing data together. This network can either function synchronously or asynchronously. ROS community part enables us to create our own

federal repositories, develop the code rapidly, share it with the community in alpha version and then through the feedback slowly converge towards the final releases.

3.2.2 KnowRob

KnowRob is a knowledge processing system that combines knowledge representation and reasoning methods with techniques for acquiring knowledge and for grounding the knowledge in a physical system and can serve as a common semantic framework for integrating information from different sources. KnowRob combines static encyclopedic knowledge, common-sense knowledge, task descriptions, environment models, object information and information about observed actions that has been acquired from various sources (manually axiomatized, derived from observations, or imported from the web). It supports different deterministic and probabilistic reasoning mechanisms, clustering, classification and segmentation methods, and includes query interfaces as well as visualization tools. In this work we use KnowRob's household domain knowledge in OWL format, KnowRob's representation language for SOM$^+$ maps presented in Chapter 4 and finally KnowRob's interface to the perception algorithms for modeling of perceived and inferred objects.

3.2.3 Point Cloud Library

PCL is a comprehensive, free, BSD licensed, library for n-D Point Clouds and 3D geometry processing. From an algorithmic perspective, PCL incorporates a multitude of 3D processing algorithms that operate on point cloud data, including: filtering, feature estimation, surface reconstruction, model fitting, segmentation, registration, tracking, etc. PCL is also fully integrated with ROS. In this thesis we mostly leverage PCL's filtering algorithms for the down-sampling of the data, segmentation algorithms to break the whole-room point cloud into a set of planes and fixtures, features for the categorization of objects of daily use and finally tracking library for the interactive segmentation of textureless objects.

3.2.4 OpenCV

OpenCV is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms that operate on 2D image data, including: segmentation, calibration, feature extraction, feature matching, tracking, etc. We deployed feature extraction and matching algorithms for the recognition of textured objects, feature extraction and optical flow-based feature tracking for the interactive segmentation of textured objects, etc. OpenCV is also fully integrated with ROS.

3.3 Datasets

As a side product of this thesis we generated two datasets of commonly encountered objects of daily use with the groundtruth pose information which the community has been using as benchmarks to evaluate their object categorization and recognition algorithms. In the following we briefly explain the configuration of both datasets.

3.3.1 Semantic3D

The platform used for the acquisition of models in this dataset is briefly described in Figure 3.2, and consists of a B21 mobile base with the calibrated compound of Amtec Powercube 6-DOF arms and sensors such as a SICK LMS400 laser device and Basler Scout stereo cameras[2]. Given that both the laser and the arm are very fast and accurate, dense scans of tables can be made in under a second. To facilitate the assembly of a large database of object models, we have created a rotating table using a DP PTU47 pan-tilt unit that is controlled by the robot over the network. Objects placed on this rotating table are scanned and saved as point cloud and image data. The resultant database of object models was then used to categorize and classify objects found in natural table setting scenes as reported in Chapter 6.

The database of 3D objects is available at http://semantic-3d.cs.tum.edu, The images in the dataset have been acquired using Basler Scout scA1390 stereo cameras at a resolution of 1390x1038 pixels. The 3D depth data was recorded using a SICK

[2]Please note that the TOF camera located on the robot head was not used.

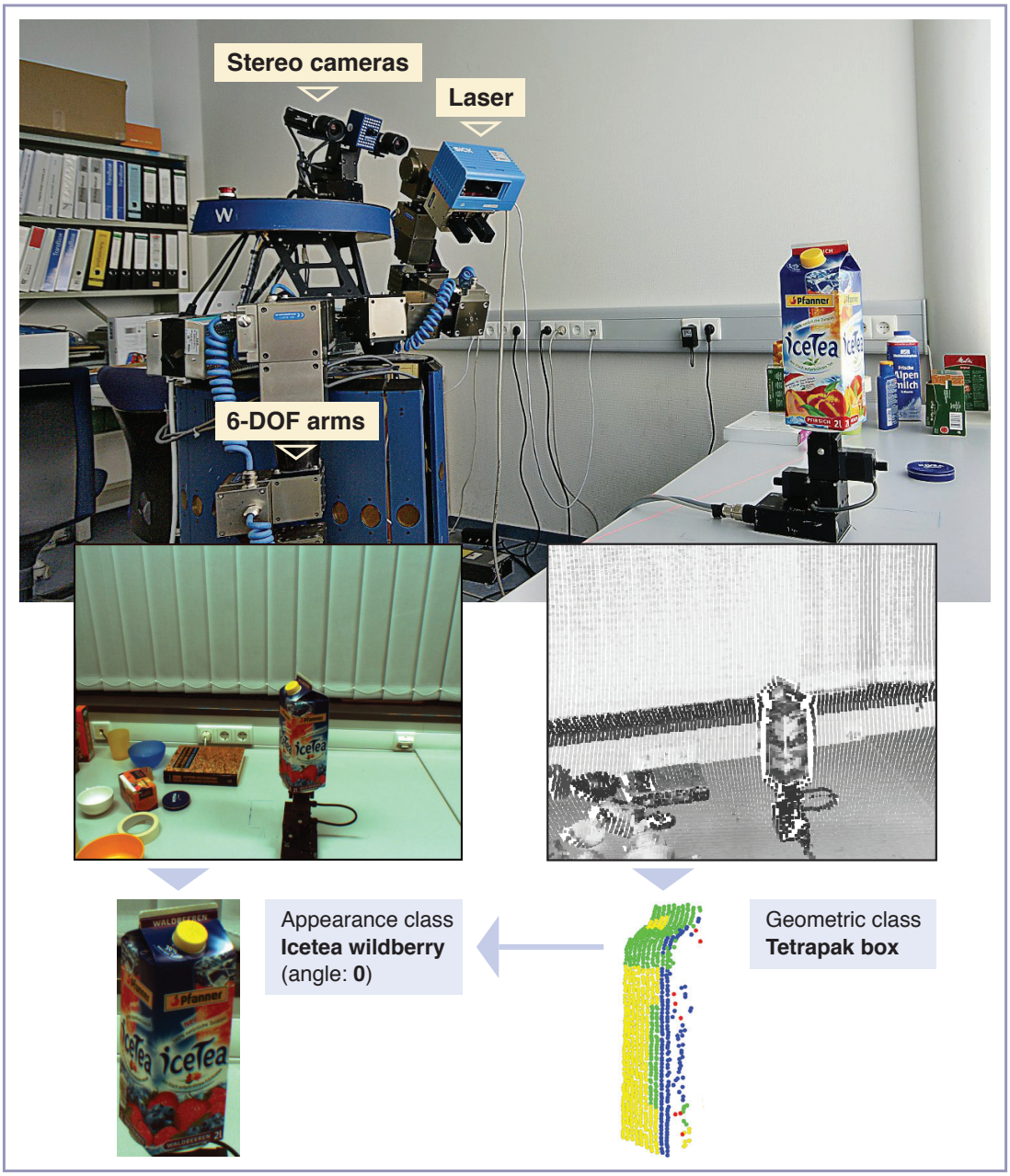


Figure 3.2: *The mobile manipulation platform used for obtaining the database and performing hierarchical object categorization and classification. The hardware setup consists of a B21 mobile base with two 6-DOF arms, stereo cameras, a laser sensor mounted on the end effector and a rotary table. The bottom area of the image shows the input as observed by the robot and the surface and geometric categorization and classification of an iced tea box.*

LMS400 range scanner with 0.5° angular resolution, resulting in point clouds with roughly 200-1000 points per object after the gross statistical removal procedure. The range scanner was tilted with $30rad/s$ during one scanning cycle.

The set of objects encompasses the ones commonly used in a typical household environment (mugs, utensils, books, etc) and is envisioned for a larger expansion in the future. In a pursue to account for a wide variety of view angles, we rotated the objects on the rotating table with a given angle-step (30° in the preliminary version) and acquired partial snapshots from a human-eye perspective, i.e. the ones that the best approximate the robot's view point during its working cycle. We consider this to be an important point as opposed to similar initiatives (e.g., [KIT, 2010]) where the datasets are acquired using high-precision but non-affordable, fixed sensors, and thus not usable for applications such as ours.

3.3.2 VOSCH

This database of 3D objects was obtained using the Kinect sensor mounted on the head of a PR2 robot. The set of objects (see Figure 3.3) encompasses those commonly used in a typical household environment as well (mugs, utensils, groceries, etc.) and is available at http://www.ros.org/wiki/vosch. In this case we rotated the objects on the rotating table with an angular step of 15° around the up-axis, and acquired partial snapshots from a perspective that best approximates the robot's viewpoint during its working cycle. For every view of the object, a VOSCH or ConVOSCH [Kanezaki et al., 2011] descriptor was estimated and a database of 63 objects was generated. An object detection pipeline with Support Vector Machines or Linear Subspace Method classifiers was then used to detect objects in natural scenes.

Figure 3.3: *TUM-James robot equipped with a Kinect sensor acquiring training data of the objects shown in the bottom-right of the figure.*

Chapter 4

Semantic Object Maps

4.1 Introduction

Robots that do not know where objects are have to search for them. Robots that do not know how objects look have to guess whether they have fetched the right one. Robots that do not know the articulation models of drawers and cupboards have to open them very carefully in order to not damage them. Thus, robots should store and maintain knowledge about their environment that enables them to perform their tasks more reliably and efficiently. We call the collection of this knowledge the robot's *maps* and consider maps to be models of the robot's operation environment that serve as information resources for better task performance.

Robots build environment maps for many purposes. Most robot maps so far have been proposed for navigation. Robot maps for navigation enable robots to estimate their position in the environment, to check the reachability of the destination and to compute navigation plans [Thrun, 2002]. Depending on their purpose maps have to store different kinds of information in different forms. Maps might represent the occupancy of environment of 2D [Thrun et al., 1998] or 3D grid cells [Wurm et al., 2010], they might contain landmarks [Montiel et al., 2006] or represent the topological structure of the environment [Kortenkamp and Weymouth, 1994]. The maps might model objects of daily use [Nakayama et al., 2009a], indoor [Rusu et al., 2009a; Henry et al., 2010], outdoor [Rusu et al., 2009c], underwater [Smith et al., 1997], extraterrestrial surfaces, and aerial environments [Shen et al., 2011].

A research area that has received surprisingly little attention is the automatic acquisition of environment maps that enable robots to perform human-scale manipulation tasks, such as setting a table, preparing a meal, and cleaning up.

In this thesis as one of the contributions we investigate *semantic object maps* (SOM$^+$s), which are a subcategory of maps that store information about the task-relevant objects in the environment, possibly including geometric 3D models of the objects, their position and orientation, their appearance, and object category. We focus here on semantic object maps that **represent all the furniture entities of kitchen environments including cupboards, electrical devices, tables, counters, positions, appearances, and articulation models**.

Overview of our system for the generation of SOM$^+$ maps is depicted in Figure 1.1 where a PR2 robot acquires the data using an RGBD sensor in a kitchen environment (top), the resulting representation of an environment as a SOM$^+$ map is in the middle and a set of example queries that SOM$^+$ map provides to the robot is shown in the bottom. We can see that the SOM$^+$ map is an abstract representation of the environment that represents the environment as a hierarchically structured object where the parts themselves are objects that have a geometric 3D model, an appearance, and a 3D position and orientation. In addition, objects might have associated articulation models that tell the robot how they can be opened and closed, which is visualized by the yellow trajectories in the bottom part of the figure.

In this chapter we investigate and describe how SOM$^+$ maps can be represented and how the representations can be automatically acquired autonomously.

In this context the key contributions of this chapter are:

- a functional end-to-end system that covers all steps required to automatically reconstruct textured SOM$^+$ models of kitchens, annotates them with the functional and semantic information and articulation models for opening and closing drawers and doors;
- methods for acquiring accurate environment maps with low-cost RGBD sensors by using vision and active manipulation actions such opening drawers and doors;

- a logic-based formal language and background knowledge for the representation of SOM$^+$ maps;
- an application level with a rich set of task queries that the system can answer and thus enable the personal robot to carry out every day manipulation tasks.

We validate the concept of SOM$^+$ maps and the robot system for their acquisition in extensive experimental studies, in which the robots operate autonomously to acquire SOM$^+$ maps in 5 kitchens.

The remainder of the chapter is organized as follows. In Section 4.2 we introduce our representation language for SOM$^+$ maps and explain how the maps are organized and how different types of information are stored and handled. Next two sections will present the system integration by i) giving an overview of the SOM$^+$ map acquisition step (Section 4.3) and ii) discussing the data interpretation step (Section 4.4). Section 4.5 presents the empirical evaluation and explains example queries and put them to use. In the final section we will conclude and discuss the on-going extensions of this work.

4.2 Representation Language and Integration with KnowRob

We represent SOM$^+$ maps as symbolic knowledge bases that contain facts about objects in the environment and that link objects to data structures such as appearance models or SIFT features which can be directly used by the robot's perception system to recognize the respective objects [1]. Encoding maps into symbolic knowledge bases rather than lower-level data structures has two main advantages: First, it allows to have a uniform interface for querying for information, combining low-level information like the dimensions and poses of objects with semantic information about their properties. Second, this approach facilitates the integration of background knowledge from other sources like the WWW [Tenorth et al., 2011] or common-sense knowledge bases [Kunze et al., 2010]. This enables the robot to apply this knowledge to reason about objects in the map, for example to infer problems that can occur when operating the dishwasher.

[1] Moritz Tenorth derived the representation language and implemented it in KnowRob as part of the joint publication [Pangercic et al., 2012].

More formally, we consider a SOM$^+$ map to be a pair SOM$^+ = \langle \mathcal{SOM}^+\mathcal{KB}, \mathcal{C} \rangle$, where $\mathcal{SOM}^+\mathcal{KB}$ is the knowledge base representing the environment and $\mathcal{C}$ is a set of inference methods that can infer knowledge that is implied by the knowledge base but not directly stored. For example, $\mathcal{C}$ includes a method to infer whether two positions p_1 and p_2 satisfy the qualitative spatial relationship "on top of".

The knowledge base $\mathcal{SOM}^+\mathcal{KB}$ itself is formalized as a triple $\langle \mathcal{T}, \mathcal{A}, \mathcal{S} \rangle$ where $\mathcal{T}$ is a terminological knowledge base that specifies the categories of objects that are used to represent the environment. $\mathcal{A}$ denotes assertional knowledge, for example that *Refrigerator67* is a physical object in the environment and an instance of concept *Refrigerator*. Finally, $\mathcal{S}$ is spatial knowledge that asserts the pose of *Refrigerator67* in the environment. The different components of a SOM$^+$ knowledge base are depicted in Figure 4.1.

The encyclopedic knowledge stored in $\mathcal{T}$ provides definitions of classes of objects and their properties, similar to those that can be found in a dictionary. It is very useful as a common vocabulary to describe the robot's knowledge. The different classes are arranged in a hierarchy, and are inter-connected by roles, a structure called an "ontology". A small part of this ontology, describing entities of furniture and household appliances, is shown in the upper part of Figure 4.1. The major part of $\mathcal{T}$ is prior knowledge that is already available before the map has been built.

The objects in the semantic map are represented as instances of the semantic classes in $\mathcal{T}$ and form the assertional knowledge base $\mathcal{A}$. It contains information about their composition from parts, e.g. that *Refrigerator* consists of a box-shaped frame, a door, a hinge, and a handle, that the door is rotationally connected to the frame by the hinge, and that the handle is attached to the door. Each of these components is described as an instance of the respective semantic class with all of its properties, e.g. the information that a refrigerator is used as storage place for perishable items, or that an oven can be used for heating food. The elements of $\mathcal{A}$ are generated from the perception system and can be passed as arguments to the robot executive. They are thus grounded in both the perception and in the action execution system.

The spatial knowledge $\mathcal{S}$ includes both metric and qualitative spatial information about the poses of objects in the environment. Metric object poses are determined by the mapping procedure (Section 4.4) and are stored in the knowledge base. Addi-

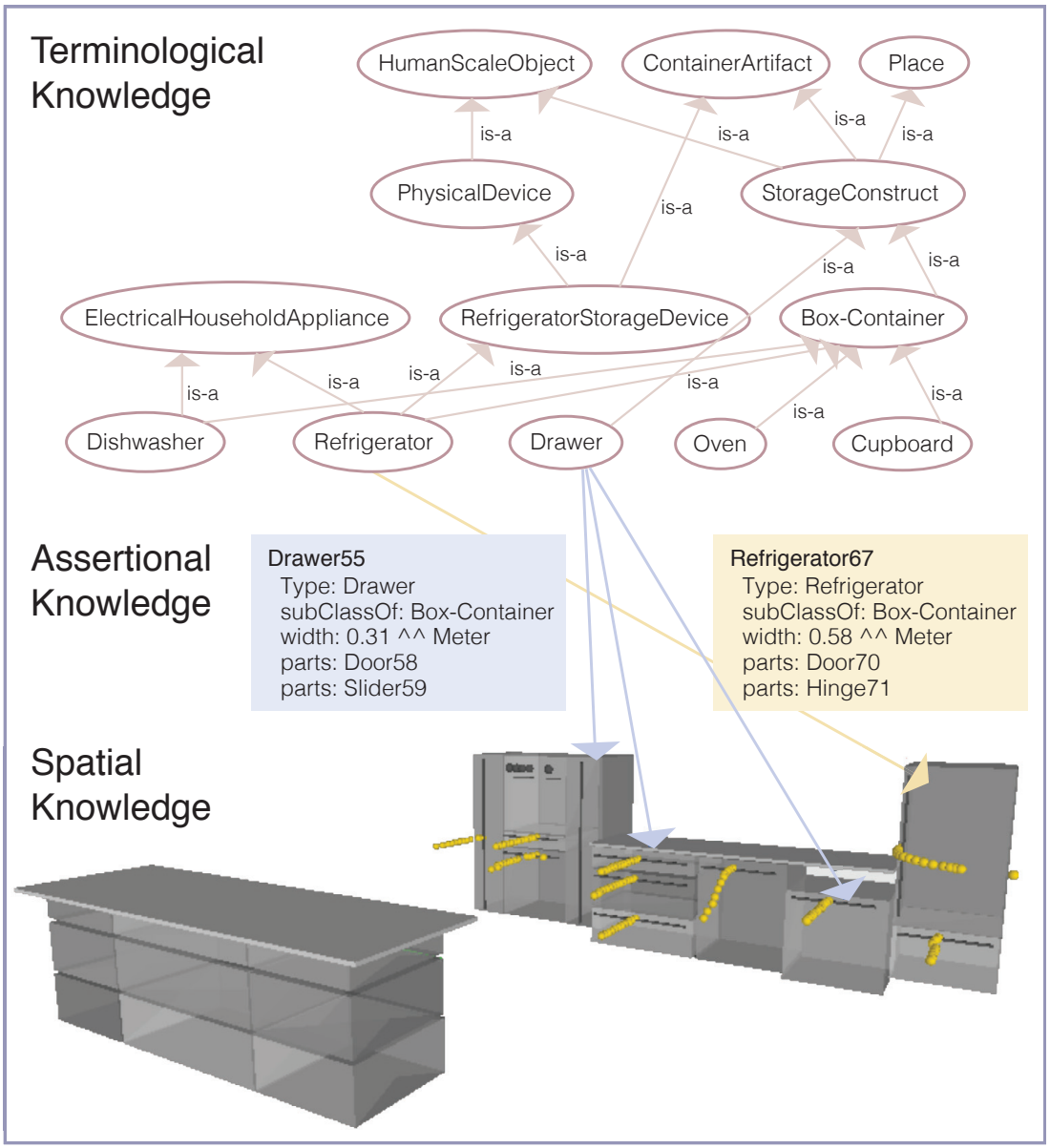


Figure 4.1: *Part of the ontology of household appliances and entities of furniture. Superclasses of e.g.* HumanScaleObject *have been omitted for better readability. Courtesys@Tenorth.*

tional qualitative descriptions, like “on the table” can be computed as a different view on the data. These more abstract descriptions are not directly stored in the knowledge base, but computed at a query time. This approach helps to avoid inconsistencies due to duplicate data storage [Tenorth et al., 2010]. The computational methods are part of the set of inference procedures $\mathscr{C}$, which further includes methods to e.g. transparently convert units of measure (Section 4.2.2).

The SOM$^+$ map provides a *tell-ask*-interface to other components in the system. The *tell*-interface allows to add knowledge to the knowledge base and is mainly used by the mapping component: Whenever new objects are detected in the environment, they are added to the knowledge base. The *ask*-interface provides reasoning services to the robot’s executive and to other components that require map information.

4.2.1 Object Representation in SOM$^+$ Maps

Most of the objects found in semantic maps of household environments are furniture entities and household appliances – which are complex, composed objects consisting of several parts (Figure 4.2). Complementary to this part-of hierarchy, the connections between parts in terms of links and joints describe a kinematic chain. In the example, hinges are described as parts of the door, which is linked to the refrigerator’s body with the *hingedTo* relation:

```
1   Individual: semanticmap14
2     Types:
3         SemanticEnvironmentMap
4
5   Individual: Refrigerator67
6     Types:
7         Refrigerator
8     Facts:
9         describedInMap semanticmap14
10        width "0.51"^^Meter
11        depth "0.59"^^Meter
12        height "0.78"^^Meter
13        properPhysicalParts Door70
14        properPhysicalParts Hinge70
15
16  Individual: Door70
17    Types:
18        Door
19    Facts:
20        width "0.51"^^Meter
21        depth "0.01"^^Meter
22        height "0.78"^^Meter
23        hingedTo Refrigerator67
24        properPhysicalParts Handle160
```

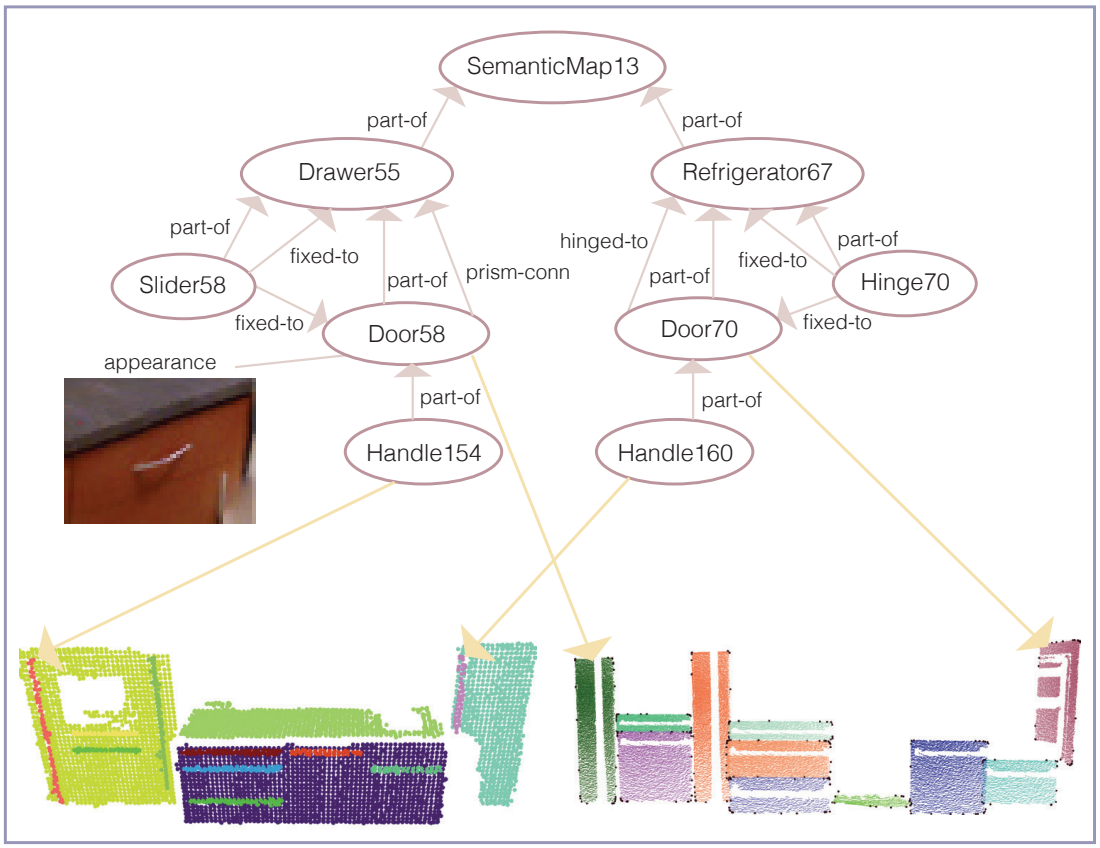


Figure 4.2: *Hierarchy of* part-of *relations between the different object components in the semantic map and a grounding example for doors and handles.*

An explicit description of the units of measure is important for the representation of spatial information in order to correctly interpret coordinate values. In the proposed representation, all numeric values can be annotated with the unit of measure that is being used. The units are described in the extensive QUDT ontology[2] including conversion factors. Compatible units, such as lengths, can be transparently converted into each other. For example, if the map contains dimensions and positions in meters, the user can query for information in feet and will automatically receive the converted values.

4.2.2 Spatio-temporal Object Pose Representation

The hierarchical representation introduced in the previous section qualitatively describes the composition of the environment out of objects and their parts, but does not specify their poses. We represent the pose information separately to account for object poses that change over time. Such a spatio-temporal representation is especially important for objects that are regularly moved, but it can also describe static objects as well as movable parts of static objects, such as the furniture doors.

We realized the spatio-temporal aspect by reifying the event that created some belief about an object pose, e.g. the detection of an object at some position. Instead of storing the information that an object "is at location A", the system thus describes that it "has been detected at location A at time T". This allows to store multiple detections of the object at different poses. In a naive implementation, attaching multiple poses to one object would lead to inconsistencies. The following code describes the detection of an object that is modeled in the knowledge base: An instance of a *SemanticMapPerception* is created for each detection (*perception24*), and is annotated with the time at which the perception has been made (*timepoint24*), the pose at which the object was estimated to be (*pose24*), and the object instance that has been perceived (*Refrigerator67*).

[2] http://qudt.org/

```
Individual: perception24
  Types:
      SemanticMapPerception
  Facts:
      eventOccursAt pose24
      objectActedOn Refrigerator67
      startTime     timepoint24

Individual: pose24
  Types:
      Pose3D
  Facts:
      m00 "1.00"^^float
      m01 "0.00"^^float
      ...
```

By default, system determines the pose of an object based on the most recent detection, but if needed, it can also go back in time and reconstruct previous world states.

4.2.3 SOM$^+$ Inference Methods

Using the inference methods $\mathscr{C}$, the system can infer novel statements from the information in the map. Let us consider the computation of the "inside" relation as an example. If this relation holds can be calculated based on the poses and dimensions of two objects. Based on the spatio-temporal representation of object poses described in the previous section, such qualitative relations can be evaluated both for the current and for previous world states.

We use the *holds(rel(?A, ?B), ?T)* predicate to express that a relation *rel* between *?A* and *?B* is true at time *?T*. The following Prolog code computes the "inside" relation in a simplified way (not taking the rotation of the objects into account) by comparing the axis-aligned bounding boxes of the inner and outer object to check whether one contains the other. First, the latest perception of each object before time *?T* is determined using the *object_detection* predicate. The poses where objects have been perceived are read using the *eventOccursAt* relation. Then, the system reads the objects' positions and dimensions, and compares the bounding boxes.

```
1
2   holds(in_ContGeneric(?InnerObj, ?OuterObj), ?T) :-
3
4       object_detection(?InnerObj, ?T, ?VPI),
5       object_detection(?OuterObj, ?T, ?VPO),
6
7       rdf_triple(eventOccursAt, ?VPI, ?InnerObjPose),
8       rdf_triple(eventOccursAt, ?VPO, ?OuterObjPose),
9
10      % read the center coordinates of the inner entity
11      rdf_triple(poseX, ?InnerObjPose, ?IX), [...]
12
13      % read the center coordinates of the outer entity
14      rdf_triple(poseX, ?OuterObjPose, ?OX), [...]
15
16      % read the dimensions of the outer entity
17      rdf_has(?OuterObj, widthOfObject, ?OW), [...]
18
19      % read the dimensions of the inner entity
20      rdf_has(?InnerObj, widthOfObject, ?IW), [...]
21
22      % compare bounding boxes
23      >=((?IX - 0.5*?ID), (?OX - 0.5*?OD)),
24      =<((?IX + 0.5*?ID), (?OX + 0.5*?OD)),
25      [...]
26      ?InnerObj \= ?OuterObj.
```

4.3 Data Acquisition

We investigate domain specific map acquisition. This means that we make assumptions/assertions about the environments such as the existence of horizontal planar surfaces at table height, or the existence of front faces of furniture pieces that contain doors and drawers and let the interpretation algorithms use this prior knowledge to infer much richer environment models that contain all the furniture objects and structures introduced in Section 4.4.

Our mapping system thus makes a set of assumptions reasonable for typical kitchens, which include the following ones. (1) Kitchens have vertical planar walls (outmost boundaries) and kitchens have planar floors (the ground plane) and ceilings. (2) Front faces of furniture are vertical planes often in front of walls. Front faces of furniture's are typically rectangular and contain doors and drawers. Front faces of drawers and doors are parts of containers (typically used for placing objects inside). (3) Doors typically have handles and hinges, drawers have handles. Both can be opened (by

pulling the handles). Some cupboards have tabletops that are planar surfaces in table height. Tables are tabletops standing on legs. (3) Some cupboards have special purposes: refrigerator, oven, microwave, dishwasher, etc. They are specializations of boxed containers. (4) Task-relevant objects are liftable and stand on table tops and shelves. (5) All others "object" structures are obstacles.

We will come back to the issue that these assumptions are of heuristic nature in Section 4.6 and outline how the next generation of the system is supposed to generalize from this overspecialization.

The SOM$^+$ mapping algorithms exploit these assumptions to better and faster process the raw sensor data through registration, plane fitting, etc and to generate and validate object hypotheses and infer better models of them.

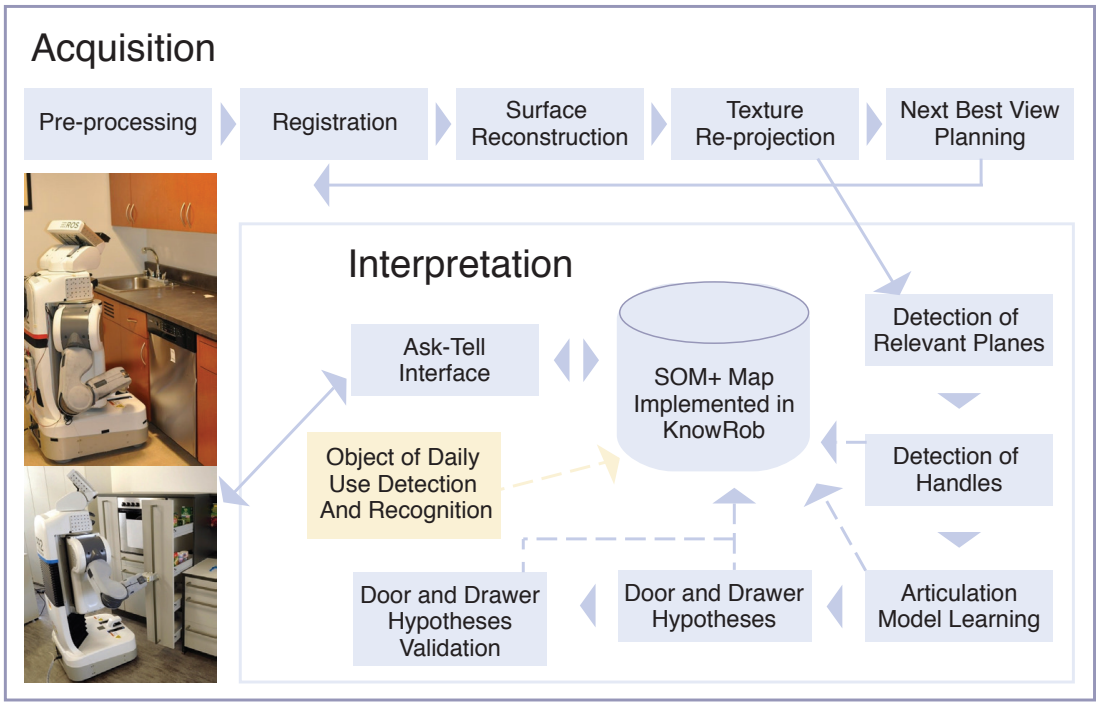


Figure 4.3: *System integration for autonomous SOM$^+$acquisition. Module for objects of daily use detection and recognition [Pangercic et al., 2011a] is part of the system but discussed in Chapter 6.*

In order to acquire a SOM$^+$ map the robot has to explore and solve a number of perceptual tasks in order to obtain the necessary information pieces. The overall struc-

ture of the map acquisition process is illustrated in Figure 4.3. The first phase in doing so is to obtain an accurate, smoothed and textured triangular 3D mesh of the environment where holes in the mesh are eliminated as much as possible (upper block in Figure 4.3). The result of this phase is a mesh representation that forms the basis for the detection, categorization and recognition of furniture objects (lower block in Figure 4.3). These two blocks will be further detailed in the following two subsections.

The SOM$^+$ mapping system is designed for autonomous manipulation platforms that are equipped with a low-cost RGBD sensor on a pan-tilt basis (we use a PR2 robot with a head-mounted Kinect sensor).

4.3.1 Acquisition of the Basic Mesh Representation

The robot acquires an accumulated RGBD point cloud by exploring the environment and panning and tilting its head in order to cover the desired view frustum. The raw data are processed using a statistical noise removal kernel and then run through a Moving Least Squares module as proposed by Rusu et al. [2008]. These pre-processing steps enable a robust alignment of the point clouds and facilitate mesh reconstruction and texture re-projection.

4.3.2 Registration

To create a consistent and accurate 3D mesh model, the individual point cloud views are transformed into one common coordinate system and merged with each other. The merging step is performed through the geometric alignment of three-dimensional views using the estimated robot position as an initial guess using a variant of the *Iterative Closest Point (ICP)* algorithm [Besl and McKay, 1992]. Here we employ the more robust *point-to-plane* variant of ICP that uses a Levenberg-Marquardt algorithm to minimize distances between points in one point cloud to respective corresponding tangent planes in the other point cloud. To avoid the accumulation of registration errors over many scans, which could cause inconsistencies in the map, we globally op-

Figure 4.4: ***Left-column:*** *Testbed kitchens at TUM and Bosch RTC.* ***Middle-column:*** *Poisson-based surface reconstruction.* ***Right-column:*** *Blending-based texture re-projection on the left surface mesh.*

timize the registration in a second step using a graph optimization technique [Grisetti et al., 2007].

4.3.3 Surface Reconstruction

To obtain a compact and fast-loading 3D model of the environment we use triangle meshes as our geometric and visual representation for SOM$^+$ maps. We apply a volumetric approach for reconstructing triangle meshes from the point clouds generated by the registration module. The first step of this approach calculates a 3D indicator function with positive values for points inside the model, and negative values for points outside. Kazhdan et al. [2006] proposed an efficient way of calculating this indicator function on a regular grid constructed of smoothly overlapping volumetric field functions using a system of Poisson equations. The second step extracts the iso-surface of this indicator function by creating mesh vertices at zero-crossings along edges of grid cells [Lorensen and Cline, 1987]. The middle column in Figure 4.4 shows the reconstructed triangle meshes of five kitchens. Each mesh consists of roughly 50K triangles while the raw point cloud is made up of more than 18M points.

4.3.4 Texture Reconstruction

In general, the environments are made out of a variety of different materials which influence their appearance. Realistic reconstruction and reproduction of the surface appearance greatly enhances the visual impression by adding more realism and can thus be used for segmentation of surfaces, environment change detection, scene analysis [Xiong et al., 2011] or for object of daily use recognition [Pangercic et al., 2011a]. To achieve the texture reconstruction we capture color images together with point clouds. We use those images to reconstruct texture maps that are mapped onto the 3D mesh. The first step of texture reconstruction computes a mapping for each mesh 3D vertex position into the 2D texture domain. In our system we use a least-squares method [Lévy et al., 2002] for finding the *conformal mapping* that minimizes distortions introduced by the 3D-2D mapping. When stitching multiple images into a texture, discontinuities on boundaries between images may become visible. For a

consistent texturing we want to minimize the visibility of those undesired artifacts. Here we employ the blending technique proposed by Pitzer et al. [2010] to globally adjust the color of all pixels simultaneously. The result is a texture composite without visual boundary artifacts. The right column in Figure 4.4 presents the final texture mapped meshes.

4.3.5 Next Best View Planning

In this chapter we focus on the SOM$^+$ maps of the kitchenette parts of indoor environments. Whole room data acquisition that requires next best view planning for a mobile base was presented in our earlier work [Blodow et al., 2011] and is based on the information gain approach in which we use costmaps to find those poses that guarantee enough coverage of the unknown space as well as sufficient overlap with the already containing data for successful registration.

4.4 Data Interpretation

4.4.1 Detection of Relevant Planes

Given the mesh generated by the texture re-projection module, our system first extracts relevant planes from it, categorizes them as walls, floor, ceiling, tables or other horizontal structures and doors or drawers. The latter is achieved by first locating the relevant planar structures, testing for the existence of handles and segmenting the doors and drawers first passively, and then actively through an interaction of the robot with the environment. As an exhaustive search for all planar structures is computationally intractable, we only search for those that are aligned with the walls of the room. The alignment of the latter is determined using a RANSAC-based approach on the normal sphere, as in [Marton et al., 2010]. Since in many indoor environments, most of the surface normals estimated at every point coincide with one of the three main axes of the room, these directions can be used to limit the plane extraction.

4.4.2 Detection of Handles

We identified two types of handle appearances[3] that have different characteristics with respect to sensor data: handles that have specular reflection and the ones that do not. To tackle these two distinct cases we propose a two-fold approach that first tries to recognize and localize a handle in a 3D model of the given environment. Shall the latter fail we resolve to finding the handle in the parts of the 3D model that lacks range measurements due to the reflection of the sensor's projected infrared light pattern on specular surfaces [Rühr et al., 2012]. We assert the handle's pose and dimension as SOM$^+$'s assertional knowledge according to Figure 4.1. The example result of such a handle detection is depicted in the bottom of Figure 4.2.

4.4.2.1 Detection of Handles without Specularity

We apply a version of the approach proposed by Rusu et al. [2009b] which uses 3D mesh data as an input and assumes that the handles are to be found at a certain distance h_d from a segmented plane of a door along the normal direction of the plane. The parameter h_d is given by the ADA (Americans with Disabilities Act) requirements as the maximum distance from a door plane where a handle could be located. Their pipeline gets all points whose distance from the plane model is smaller than h_d, and checks whether their projection on the plane falls inside the bounding polygon of the plane. The actual handle is obtained by first projecting the handle on the door plane, and then fitting the best line within a plane parallel to the door in it using a RANSAC-based line fitting to infer the correct orientation. The geometric centroid of the handle cluster is then used along with the orientation of the line and the plane normal to grasp the handle.

4.4.2.2 Detection of Handles with Specularity

The handle detector described in the previous subsection works well when the handle is thick enough to be visible in the 3D mesh acquired by the robot. In practice, we

[3]Please note that we only considered handles that correspond to the Americans with Disabilities Act: http://www.ada.gov/pubs/ada.htm.

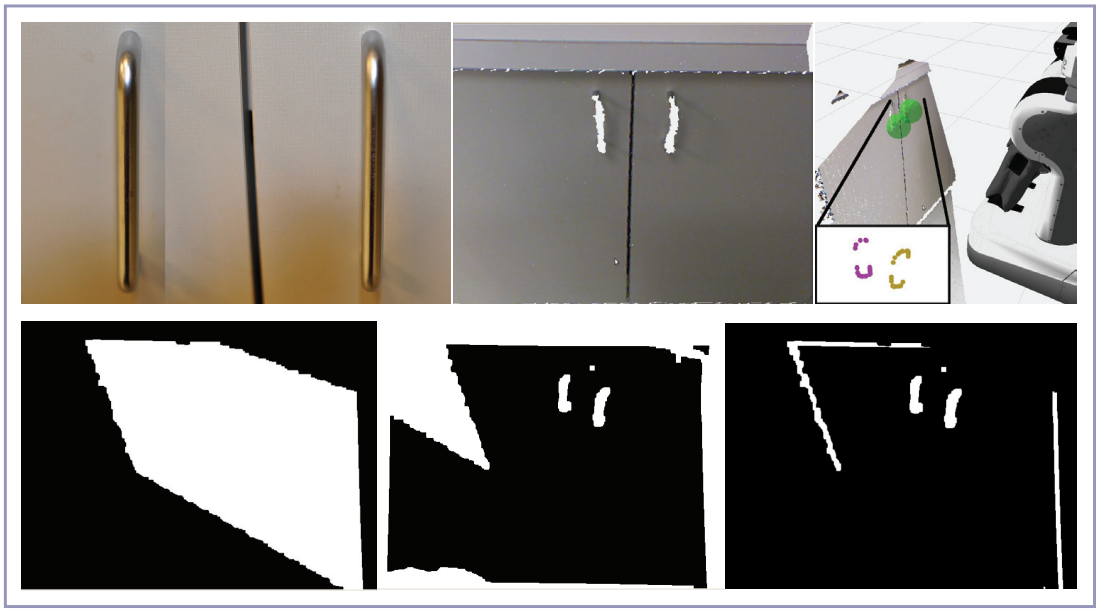


Figure 4.5: *Visualization of the processing steps for the handle detector based on invalid measurements.* Top-left: *example of two specular handles,* top-middle: *invalid measurements in place of specular handles as seen in a point cloud by Kinect,* top-right: *final handle poses (green spheres) computed from the generated convex hulls visualized in the corner of the subfigure,* bottom-left: *cabinet front face as binary mask (white),* bottom-middle: *invalid measurements as binary mask (white),* bottom-right: *result of a bit-wise conjunction operation.*

found that this strategy fails when the handles are too thin or specular, so that they are not seen by the sensor. Technically, the Kinect sensor requires that a large enough block of the projected pattern to be visible to compute the disparity [WillowGarage, 2010b]. In case of thin or specular handles the infrared pattern gets reflected, which results in low correlations during block matching so that the sensor returns missing values in these regions (see Figure 4.5, top-left).

Our key idea is thus to actively exploit patches with invalid measurements ("holes") in the depth images of the Kinect sensor, as they potentially indicate the presence of a specular handle. To robustly detect these holes and restore their poses we proceed as follows. Firstly we create two binary masks (2D images) of the invalid measurements and projected points of the detected cabinet face (ROI). Next we perform a series of dilation and erosion operations on such generated images to fill the holes in the binary images. Following we perform a bit-wise conjunction of the two binary masks and obtain an image containing handle candidates within the ROI (Figure 4.5, bottom row). Following we apply an Euclidean clustering using a region growing approach and keep only clusters C that correspond to the expected size of the handle in the image. To restore the position of the centroid of the handle we first compute the convex hull H around every cluster c_i from a set of clusters C, find the corresponding 3D point in the ROI point cloud for every point h_i on the hull H and compute the centroid (Figure 4.5, top-right). The orientation of the handle is calculated by first converting the image with the clusters into an edge image using a Canny operator and then using a RANSAC-based line fitting. The pose of the handle is finally transformed into the coordinate frame of the base of the robot (with Z pointing upwards and X pointing forwards) and the handle is grasped. To find the distance between the handle and the supporting plane we make use of the PR2's tactile sensors in its fingertips: We steer the robot's arm towards the transformed pose and the X component of the handle position is determined as the contact point between the handle and fingertip.

4.4.3 Articulation Model Learning

To open the cabinets we use a controller developed by Sturm et al. [2011]. The controller assumes that the robot has already successfully grasped the handle of an

articulated object and that a suitable initial pulling direction is known. The robot then pulls in this direction using an equilibrium point control (EPC) [Jain and Kemp, 2010] and observes the resulting motion of its end effector. From this partial trajectory, it continuously (re-)estimates the kinematic model of the articulated object. The robot uses the kinematic model to predict the continuation of the trajectory. To deal with the workspace limits of the manipulator we make use of a secondary controller that moves the omni-directional base of the robot so that the reachable volume of the manipulator is maximized. After the motion of the end effector has come to a rest, the range of valid configurations of the articulated object is estimated. In sum, this gives us the full kinematic model of the articulated object. Finally, we sample the so-generated trajectory and store the poses of the sampled points on the trajectory as SOM$^+$map spatial knowledge according to Figure 4.1. An example of the model learning step is visualized in Figure 4.6.

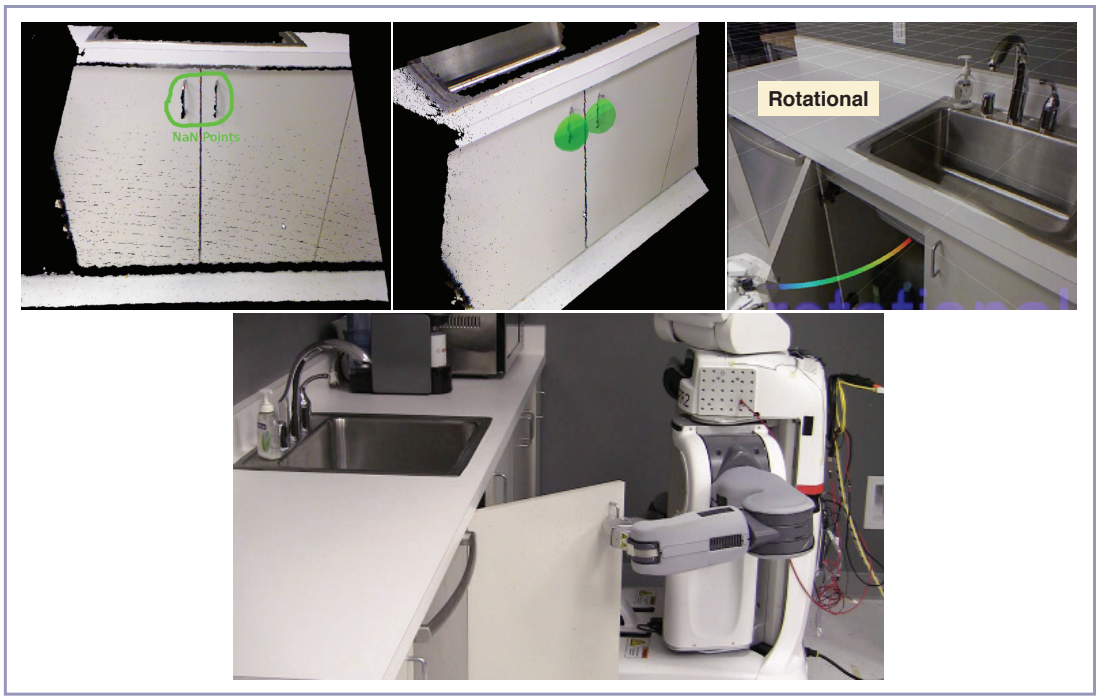


Figure 4.6: *The PR2 robot operates the cabinet in the Bosch RTC kitchen and learns the kinematic model. Top-left and middle figures depict a pair of doors with the handle with specularity and a successful handle detection. Top-right and bottom row figures show two snapshots from the opening sequence.*

4.4.4 Generation of Door and Drawer Hypotheses

This module, initially proposed by Blodow et al. [2011], uses mesh vertices as seed points around footprint of handles to estimate an initial model of the color distribution of the door. The model consists of the intensity values' median $\tilde{i}$ and median average distance (*MAD*). The seed regions are expanded by adding neighboring vertices whose colors match the estimated color model, using a basic region growing algorithm based on the assumption that vertices on the door border are surrounded by vertices with different color. The color model for a region is updated after all possible vertices are added, and the process is repeated until the values of $\tilde{i}$ and *MAD* stabilize. After this step, fixtures that produce overlapping segments are marked for further examination, while the rest are added to the SOM$^+$ map, along with the rectangular approximations to the found planar segments.

4.4.5 Active Door and Drawer Hypotheses Validation through Interaction

Concurrently with the learning of the articulation models we also make use of the movement of the respective front of a cabinet and accept and reject the hypothesis generated in the previous subsection. To achieve this we use a temporal difference registration of two *point clouds* (of a closed and an open cabinet) as put forth by Rusu et al. [2008], using a search radius parameter of 0.5 cm, which corresponds to the noise level of the sensor data after the pre-processing step as discussed above. We project the points that only appear in the second point cloud (corresponding to the door or the drawer $plane_{SEG}$) by applying the inverse of the transform between the first and the last pose of the stored opening trajectory. We then obtain the convex hull around such projected $plane_{PROJ}$, and assuming an environment based on rectangular furniture, we extract the width and the height of the cabinet front. For prismatic joints such as in case of drawers, we compute the distance between the two planes, which gives us the maximum opening distance and the depth of a drawer. For rotational joints, we assume that the depth of the cabinet is the same as the depth of the horizontal surface above it. We store poses and dimensions of cabinets

as SOM$^+$map's assertional knowledge according to Figure 4.1. Result of the final segmentation is shown in the bottom of Figure 4.2.

4.5 Results

We evaluated the proposed integrated approach in five kitchens (see Figure 4.4) by measuring the quality of the generated SOM$^+$ map in terms of the handle re-detection and the re-opening of the doors using learned and stored articulation models, and by measuring the average run times needed to generate one instance of SOM$^+$ map. In the accompanying video[4] we also present a range of possible queries that our system can answer but are currently still difficult to evaluate quantitatively.

kitchen	#cabinets	#trials	#handle detection success	#opening success (w/o model)	#opening success (w model)
1	3	9	9 (100%)	8 (89%)	9 (100%)
2	5	15	15 (100%)	15 (100%)	15 (100%)
3	7	21	**18 (86%)**	19 (90%)	18 (100%)
4	1	3	3 (100%)	0 (0%)	3 (100%)
5	6	18	18 (100%)	14 (78%)	18 (100%)
Total:	22	66	63 (95%)	**56 (85%)**	**66(100%)**

Table 4.1: *Results of detecting the handles and opening the cabinets based on the information derived from the SOM$^+$ map.*

4.5.1 Door Opening

In this experiment, we had the PR2 robot detect handles and three times open each of the 22 cabinets within five different kitchens (see Figure 4.4). Due to the PR2's limited arm reach, we omitted the cabinets with handles located above $1.2m$ and the cabinets positioned in constrained spaces such as the ones adjacent to walls. The objective of the experiment was to asses the detection rate of handles given their a priori poses stored in the SOM$^+$ map, and to evaluate the robustness of a cabinet

[4]http://youtu.be/B7kMviETh50

opening given their a priori learned and stored articulation models. The results of the experiment are presented in Table 4.1. In column four we notice that the detection of the handle only failed three times. All failures occurred on a cabinet located next to the metal dishwasher that generated the invalid measurements which our handle detection algorithm took as a handle hypothesis. Column five presents the success rate of opening the cabinets without a priori learned model and column six with the a priori learned model. Playing back the stored trajectories turned out to be be 100% successful.

4.5.2 Performance Profiling

In Table 4.2 we broke down our processing pipeline into a set of independent components and profiled their performance on Intel Xeon W3520 desktop computer with 2.67GHz processor and 24GB of memory. Total time amounting to building of one SOM$^+$ map of one kitchenette from the scratch is $1.2h$. Querying times for the information stored in SOM$^+$ map are around $1s/query$.

$Component$	$\overline{Runtime}$
Data acquisition and pre-processing	$0.1h$
Registration	$0.4h$
Surface reconstruction	$0.3h$
Texture re-projection	$0.3h$
Door opening and segmentation	$0.1h$
Generation of SOM$^+$map	$1s$
Total	$1.2h$

Table 4.2: *Execution times for building of SOM$^+$maps (Figure 4.3).*

4.5.3 SOM$^+$ Example Queries

The bottom part of Figure 1.1 and an accompanying video[5] show different queries that can be answered by the SOM$^+$ map representation. Let us consider the following query as an example:

[5]http://youtu.be/B7kMviETh50

```
1  ?- rdf_triple(knowrob:'in-ContGeneric',knowrob:'Cup67',B),
2      rdf_has(B,knowrob:openingTrajectory,Traj),
3      findall(P,rdf_has(Traj,knowrob:pointOnTrajectory,P),
4              Points).
```

It reads the trajectory for opening the container where cups are stored in by first computing the 'in-ContGeneric' relation based on the poses and dimensions of the objects. For the resulting containers, it is checked whether there is an opening trajectory attached, and if that is the case, all points on this trajectory are returned. This query shows how the semantic map representation can translate qualitative abstract queries into information that can be used to parametrize the robot's actions such as the trajectory. In Chapter 7, we show how different kinds of knowledge can be integrated with semantic maps, such as statistical relational information [Tenorth et al., 2010].

4.6 Discussion

In this chapter we presented an integrated system for semantic mapping which enables the robot to build Semantic Object Maps that support rich and powerful queries. While having such queries is appealing and useful at first, we also evaluated and validated the system and proved that robot with SOM$^+$maps can execute its tasks much faster and more reliable. We are aware that some of our perceptual heuristics do not fit (to e.g. old fashioned doors or doors without handles) and will in the future continue working towards the ensemble of experts-based methods (first version of such system we implemented and describe in Chapter 6.5) to alleviate that. Furthermore, we will also integrate our algorithm for the recognition of beds, chairs, etc. [Mozos et al., 2011] to scale towards mapping of whole apartments. Another avenue worth exploring to overcome the heuristic nature of the perceptual routines is to learn probabilistic models for the appearance of furniture entities. To this end we will use the textured component of the photo-realistic textured mesh. This however requires huge training data bases [Mozos et al., 2011] and the scaling of probabilistic learning and reasoning. Our system requires approximately $3h$ to build a SOM$^+$map of a kitchen room. Even though this contributes to the long robot setup time, we believe that it is still permissive given it is required to be done only once at the beginning of robot's inception as a robot companion.

Chapter 5

Interactive Segmentation of Textured and Textureless Objects

5.1 Introduction

A robot operating in human environments may be required to perform complex dexterous manipulation tasks in a variety of conditions. Personal robots are likely to perform tasks that require them to interact with objects that populate human environments. For example, when emptying a shopping bag [Klingbeil et al., 2011b] the robot is likely to be confronted with a cluttered unstructured scene like the examples shown in Figure 5.1. In order to successfully perform this task, the robot must be able to detect the individual objects. Without the ability to interact with the environment, it can be difficult to distinguish between the object boundaries and the background.

To further exemplify this case we point to Figure 5.2 which consists of objects of similar colors, shapes and sizes. To demonstrate how difficult is it to segment them we tested three state-of-the-art segmentation algorithms operating in depth, RGB and RGBD space respectively on the given scene. We notice that they are far from being optimal in the cases of i) same color objects (a coffee mug and a saucer), ii) similar shape objects and occlusions (a white and a blue box), iii) stacked objects (an egg and a plate) and also in the case of iv) a sensor default (cutlery in this case appears transparent to the Kinect sensor). Following structure from motion approaches, one could observe the scene from various views and apply merging of hypotheses. This approach would however fail in the case of non-navigable spaces for the robot. While one can certainly fine tune the algorithms' parameters for a certain setup and environment, it is easier and arguably more natural to exploit the robot's embodiment

Figure 5.1: *Top: PR2 robot successfully picking-up the object after segmenting it in clutter using segmentation algorithm for textured objects. Bottom: the result of clustering of two highly cluttered scenes.*

and interaction capabilities in order to obtain a better understanding of its environment. Reaching out to get a sense of what is around is the way how infants get to know their "near space" according to Piaget's theory of spatial cognition in the sensorimotor stage (until the age of 2), and getting a hold of connectivity (i.e. object unity) is an important factor in the infant's understanding of objects at that stage [Cohen and Cashon, 2003].

To solve above and similar challenges we, similar to Katz and Brock [2011] and Bergström et al. [2011], propose a system that uses a robot arm to induce motions in a scene to enable effective object segmentation. Our system employs a combination of the following techniques: i) estimation of a contact point and a push direction of the robot's end effector by detecting the concave corners in the cluttered scene, ii) feature extraction using features for textured and textureless objects, iii) tracking using algorithms optimized for the before selected features and finally, iv) two novel clustering algorithms to segment both types of the objects.

Research in passive perception has traditionally focused on static images and segmented images based on a set of features such as color [Bruce et al., 2000] or higher order features like in graph cut approaches [Boykov and Kolmogorov, 2004]. Other approaches, such as that of Vidal et al. [2004], employ camera motion for image segmentation.

Segmentation of rigid objects from a video stream of objects being moved by the robot has been addressed by Fitzpatrick [2003] and Kenney et al. [2009]. These works are based on the segmentation of objects from a video stream of a pre-planned arm motion, use a simple Gaussian model of the color values to infer the possible motion and a graph cut algorithm for the final object segmentation. These approaches can deal with textured as well as textureless objects. In contrast, our arm motion is not pre-planned but adapts to the scene and we make use of the 3D data to segment the object candidates from the background.

Katz and Brock [2011] address the problem of segmenting the articulated objects (e.g. drawer). A Lucas-Kanade tracker and a set of predictors (relative motion, short distance, long distance, color, triangulation and fundamental matrix) are applied to obtain rigid body hypotheses (in a form of a graph) and a subsequent fixation point on the object. The latter is used to segment an object based on color, intensity and

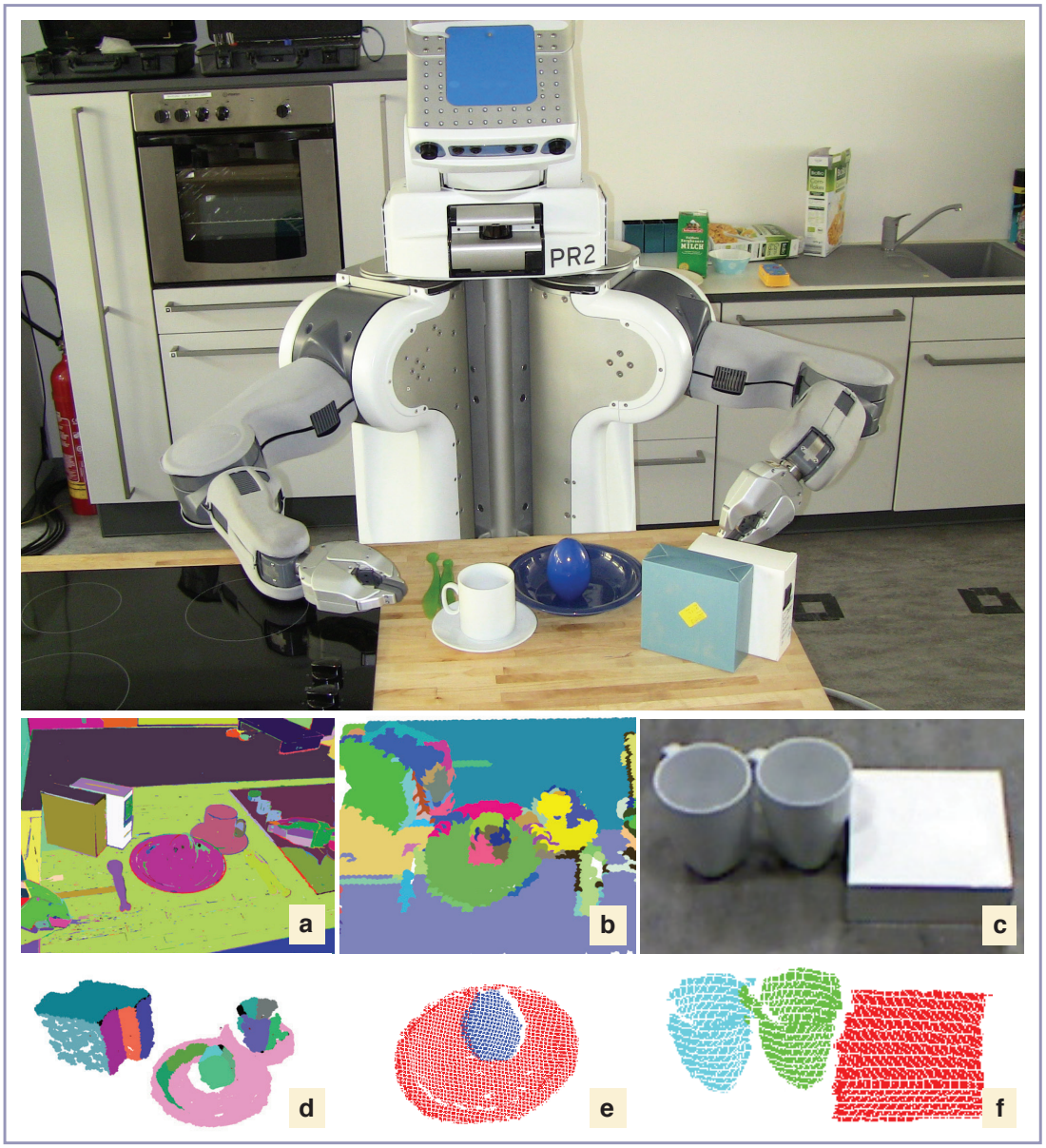


Figure 5.2: *Top: The service robot PR2 aiming to segment the scene consisting of textureless object. Results of the scene segmentation using Region Growing method [Zhan et al., 2009] (b), Part-Graph-based Hashing [Marton et al., 2012] method (d) and Graph-based segmentation method [Felzenszwalb and Huttenlocher, 2004] (a). These methods work in depth, RGB and RGBD space respectively and all underachieve due to the complexity of this challenging task. On the other hand blue egg on the blue plate (e) was correctly segmented using the interactive approach presented in this chapter. Subfigures c and f: 3 white objects segmented correctly showing the generality of the approach for multiple objects.*

texture cues. The major limitation of this approach is the pre-planned arm motion and the time needed to break the graph of object hypotheses into the subgraphs using a min-cut algorithm.

Bergström et al. [2011] propose an approach to interactive segmentation that requires initial labeling using a 3D segmentation through fixation which results in a rough initial segmentation. The robot interacts with the scene to disambiguate the hypotheses. Points in the motion space are clustered using a two component Gaussian mixture model. A limitation of the system lies in the number of objects, which was never greater than two in the experimental results.

Some approaches examine how the perturbations can be planned to accumulate a sequence of motion cues. Gupta and Sukhatme [2012] use a set of motion primitives consisting of pick and place, spread, and tumble actions to sort cluttered piles of single-color objects. Euclidean clustering is used in the distance and the color space to classify the scenes as uncluttered, cluttered, or piled. Distance-based clustering is limited as its success is subject to correctly selected threshold. Color-based clustering may fail in the presence of sudden lighting changes. Additionally, this approach can not deal with heavily textured objects but could work well in combination with ours. Chang et al. [2012] present a framework for interactive segmentation of individual objects with an interaction strategy which allows for an iterative object selection, manipulation primitive selection and evaluation, and scene state update. The manipulation primitive selection step uses a set of heuristics to maximize the push action, however, it is unclear in how much this component contributes to the successful segmentation of the objects. The manipulation primitive evaluation step uses sparse correspondences from the Lucas-Kanade optical flow tracker and computes a set of transforms which are color matched against a dense point cloud. A likelihood ratio of a target being a single item or multiple items is determined based on the magnitude of the transform motion and the percentage of dense point matches. The major limitation compared to our work is that they do not estimate corner contact points and do not accumulate the transforms across the history of push actions.

Finally, there is a corpus of works dealing with the estimation of the articulation models for drawers, boxes, etc. [Yang et al., 2011; Sturm et al., 2010]. The common problem for both approaches is in that they assume the presence of a large, moving plane which they can reliably detect by running e.g. a RANSAC algorithm on the

input point cloud and which unanimously represents the part of the object they are looking for.

Overall, we present the following main contributions for the segmentation of scenes consisting of textured and textureless tabletop objects.
Textured Objects:

- A heuristic for finding a contact point and a push direction for the robot's manipulator to induce distinct motions to effectively separate the objects (Section 5.4).
- A novel clustering algorithm for 2D-feature trajectories that is based on sampling rigid motion hypotheses for the *a priori* unknown number of scene objects (Section 5.5.1);

Textureless Objects:

- A set of RGBD features suitable for the tracking of flat and round textureless object (Section 5.5.2.1);
- A novel graph-based algorithm for the clustering of 3D-feature trajectories, in which graph edges measure the dissimilarities between the RGBD features' distances (Section 5.5.2.3);
- The inclusion of a static scene pre-segmentation algorithm and a probabilistic method for the detection of over or under-segmentation (Section 5.3);
- A dense model reconstruction algorithm that makes use of the already clustered features (Section 5.6).

Finally we also present the integration of all the above into two separate pipelines using ROS as depicted in Figure 5.3 and Figure 5.4.

This chapter is organized as follows: System pipelines for segmentation of both types of the objects is presented in Section 5.2, pre-segmentation step for textureless objects in Section 5.3 and contact and push point estimation in Section 5.4. Actual segmentation algorithms are outlined in Section 5.5, which is followed by Section 5.6

about dense model reconstruction. Finally we present the experimental results and conclude with the discussion[1].

5.2 System

5.2.1 Textured Objects

The overall system schema for interactive segmentation of textured objects is depicted in Figure 5.3 and consists of four main steps. In the first, a 3D point cloud of a static tabletop scene with household items is captured by a depth camera (in this case using the Kinect). Points belonging to the table are separated from points belonging to the group of objects. The shape of the group of objects is used to infer a contact point and a push direction for the robot's manipulator.

In the second step we capture a sequence of Kinect RGB camera images of the scene, while the robot pushes its end effector into the group of objects. Shi-Tomasi features are extracted from the first camera frame and then tracked over subsequent frames using optical flow. Once object motion is detected, the robot continues moving its end effector a predefined distance along the push direction. Since we assume the robot arm is calibrated and we have a model of its geometry, we employ a self-filter to exclude features from the robot's arm.

In the third step, the recorded feature trajectories are clustered and assigned to object hypotheses. Finally, in the fourth and the last step the dense model is reconstructed using the region growing algorithm where the clustered point features are used as seed points.

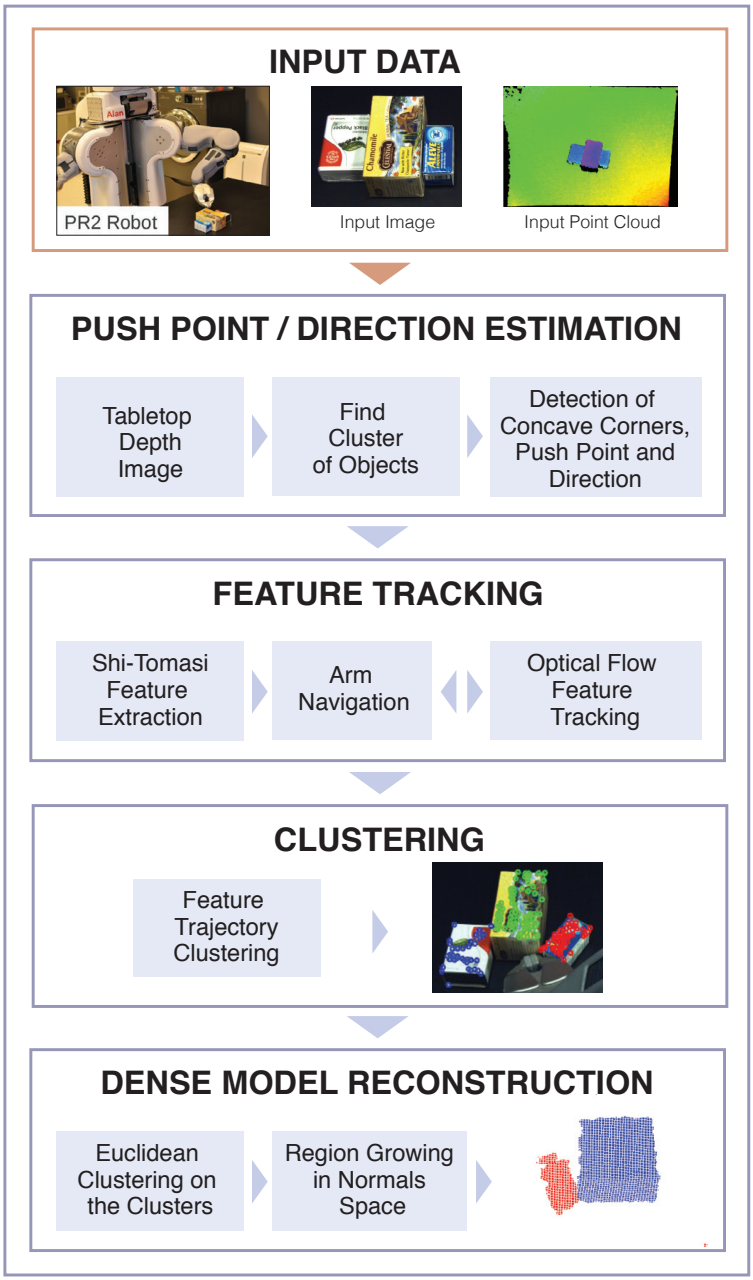


Figure 5.3: *This subsystem consists of four main nodes: a node for estimating the initial contact point and the push direction, a node that extracts 2D-features and tracks them while it moves the robot arm in the push direction, an object clustering node that assigns the tracked features to objects and finally, a dense model reconstruction node.*

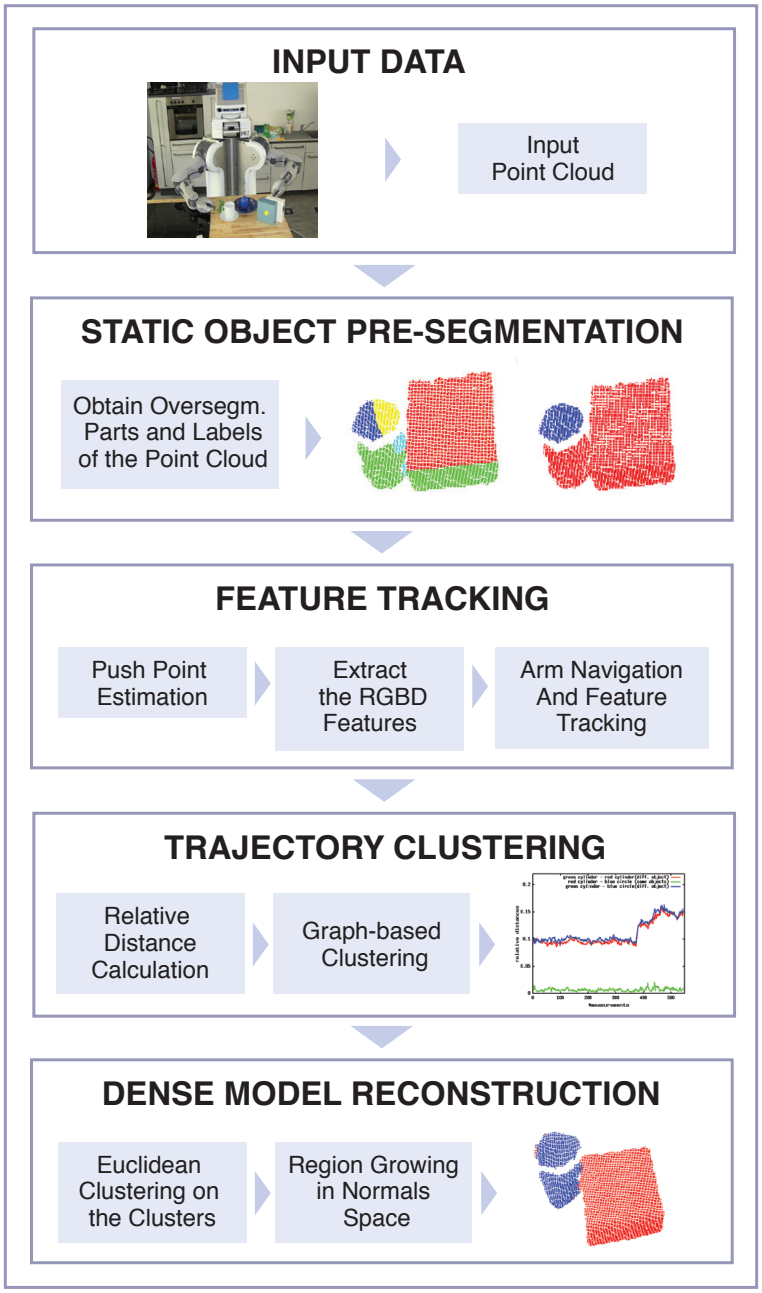


Figure 5.4: *System pipeline for the segmentation of textureless objects.*

5.2.2 Textureless Objects

Our approach for segmentation of textureless objects consists of five main steps as depicted in Figure 5.4 and demonstrated in an accompanying video[2]. In the first step we obtain an RGBD point cloud from the Kinect sensor. In the second step we perform static object pre-segmentation which results in a set of categorized object hypotheses O, with the category being either flat or round, and a list of object parts P_o that every object $o \in O$ consists of. Having obtained the object hypotheses O we infer which hypothesis is segmented correctly. For that we count the number of parts that the respective object hypotheses O consists of and then sample from the Poisson distribution according to the Equation 5.1. After obtaining the probability of the scene being segmented correctly we decide if the interactive segmentation algorithm should be used or not.

We use categorization of the objects as a prior for tracking by extracting and tracking line and corner RGBD features on the flat object hypotheses and circle and cylinder RGBD features on the round ones in the third step. Finally, we execute the arm motion movement in $1cm$ intervals until we reached a maximum of pre-defined number of pushes. All of the features are being tracked during the interaction and the trajectories of feature centroids are being saved. Based on relative distances between the feature centroids, the graph-based algorithm for the trajectory clustering is applied. The output of the algorithm is the number of objects belonging to a certain object hypothesis o and the association between the object number and the parts $p_1, \ldots, p_n \in P_o$ that belong to it (fourth step). In the fifth and the last step the dense model is reconstructed using the same region growing algorithm as in the textured objects case but with the difference that clustered RGBD features are used as seed points.

[1] System integration, graph-based trajectory clustering algorithm and dense model reconstruction were implemented as joint work with our colleagues Karol Hausman, Ferenc Balint-Benczedi and Zoltan-Csaba Marton [Hausman et al., 2013].
[2] http://youtu.be/Bu4LayrGC1s

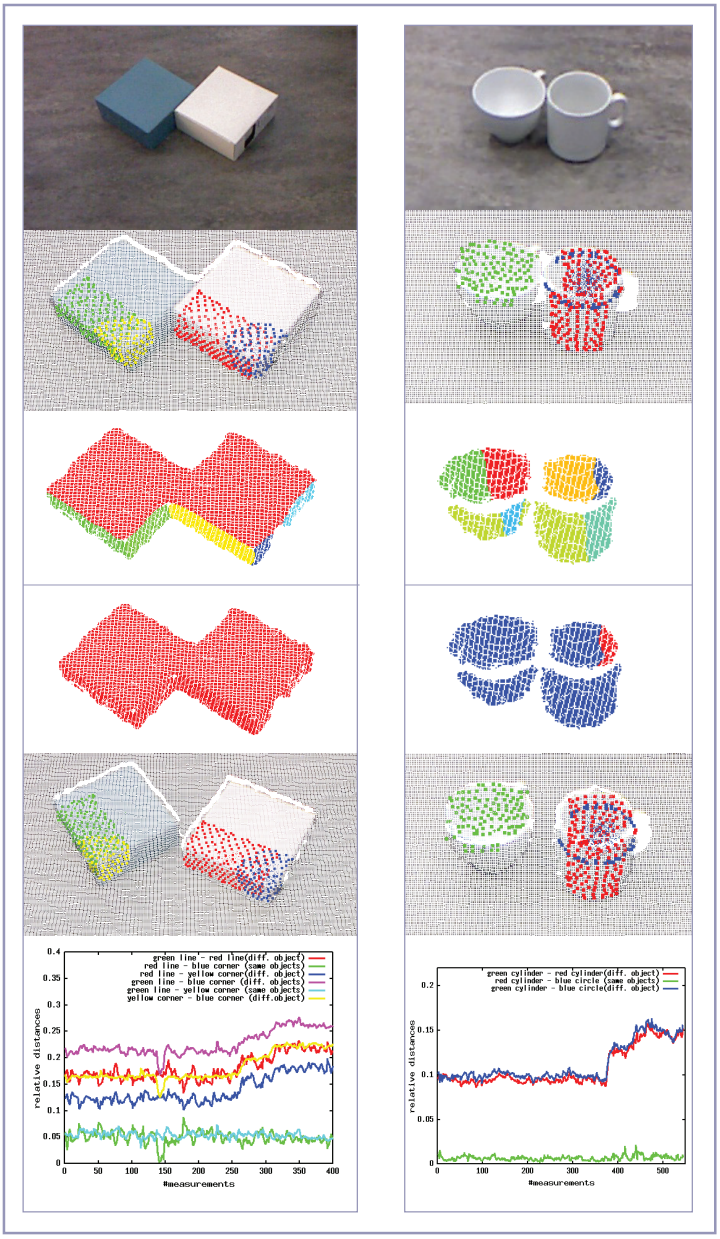


Figure 5.5: *Two test scenes in the left and right column respectively. First row: original scenes; second row: extracted RGBD features before the interaction; third row: parts P from the static segmentation; fourth row: object hypotheses O from the static segmentation; fifth row: tracked RGBD features after interaction; sixth row: relative distances between the tracked features. Plots with the ramp denote distances between features on different objects and plots with the constant values denote distance between features on the same object.*

5.3 Textureless Objects: Static Pre-segmentation

For the interactive segmentation of textureless objects we first apply a set of priors which aid us to select the right feature for the right type of the object. We make use of the classification method presented by Marton et al. [2012] which is based on part-graph-based hashing. The basic idea behind it is that segmenting objects accurately in a cluttered scene does not always yield the expected result, as seen in Figure 5.5 row 4, and can lead to classification failures, but over-segmenting is easily realizable [Malisiewicz and Efros, 2007; Lai and Fox, 2010; Mozos et al., 2011]. We use the classification approach described by Marton et al. [2012] for categorizing over-segmented object parts in cluttered scenes by considering combinations of these parts to compute features and classify these efficiently with the aid of hashing. The result is a set of labeled parts with geometric categories that can be grouped in order to obtain object hypotheses. Based on statistics computed from the training data on single objects, we can estimate how likely it is that an object hypothesis is correct.

In the rest of the section we summarize the part-graph-based hashing algorithm briefly and show how we use it to guide the interactive segmentation.

5.3.1 Decomposition into Part Graphs

In order to find the parts ($p_1, \ldots, p_n \in P_o$) in the point clouds we use the clustering criteria presented by Mozos et al. [2011], such that patches with a small curvature are considered, as shown in Figure 5.5 row 3. For each part we subsequently compute GRSD- (Global Radius-based Surface Descriptor [Kanezaki et al., 2011]) feature and store it for later use. We then extract the part neighborhoods by checking if the physical distance between two parts falls below a threshold of $2cm$ (considering Kinect noise level [WillowGarage, 2010a]), and build a connectivity matrix. Starting at each vertex of the connectivity matrix, we create all the possible groupings up to a certain size (eight parts in the case of single objects and four in the case of cluttered scenes) in order to obtain the "soup of segments", and create the groups' hash codes using isomorphic graph metrics. The hash codes are then used to further split the feature space ending up with a separate classifier (nearest neighbors in our case) for each hash code. During the classification phase we obtain confidence votes only from those

classifiers, which were created for the hash codes that are found in our scene. Based on these votes a decision is made upon the class of the segments. For a detailed description of this approach please refer to Marton et al. [2012].

5.3.2 Object Part Categorization

The classifier was trained on a subset of the dataset from Lai et al. [2011a] as presented in Marton et al. [2012]. The choice of the feature determined for each part, namely the GRSD- is motivated by the fact that we are dealing with novel objects not seen before by the classifier, so in order to successfully categorize them we need to use geometric features. Additionally, the low dimensionality and additive property[3] make GRSD- a suitable choice for such task.

Objects ($o_1, \ldots, o_n \in O$) are categorized in six geometrical categories: sphere, box, rectangular/flat, cylinder, disk/plate and other. Doing this we get a better a discrimination between different objects. After having the results for the six geometrical classes, we merge them together into different *object types* considering everything spherical and cylindrical being *round*, and disks/plates, flats and boxes as *flat* objects. With the category *other* we thus get three object types, whereas most household objects fall into the first two [Marton et al., 2011].

In this system we omit the category *other* and use the other two in order to determine if the interactive segmentation is needed, and if yes, which RGBD features to extract and track in the respective part of the point cloud in the given scene.

5.3.3 Verification of Correctness of Segmentation

Since the geometric categorization of parts does not give the correct grouping of these parts to form objects, simply grouping the parts of the same category together does not always separate the objects, especially if classification errors occur too. A method of voting for object centroids followed by a model fitting step was described by Mozos et al. [2011], but we assume having no CAD models for test objects in

[3] If the feature is additive, the descriptor that would be computed for the object is the same as the sum of the features of its segments.

this system. We would also have to consider 6DOF poses, complicating the approach considerably.

Whereas the segmentation of objects is not uniquely defined, there are still regularities in the number of parts they are broken up into. As shown in Figure 5.6, the distribution of the number of different object parts, generated in the training stage of the part-graph-based hashing algorithm, can be modeled as a Poisson distribution, with an average error of 1.7% (and at most roughly 9%).

The Poisson distribution described by Equation 5.1 describes the probability of different number of events occurring in a given interval, which we interpret here as the number of part boundaries encountered over the surface of the scanned object. The parameter λ is the mean of number of parts, which in our case is 0.876 for flat, 2.166 for round, and 3.317 for other object types.

$$P(k\ parts\ forming\ a\ single\ object) = \lambda^k \exp^{-\lambda} / k! \tag{5.1}$$

This simple model is used to judge if a group of parts of the same geometric category forms a single object or if the robot should try to interact with it. We cut the probabilities at 0.3 for flat and 0.15 for round objects.

Example: To demonstrate this, from the middle part of Figure 5.6 we can deduce that the flat object is most likely to consist of 1 or 2 parts. The test scene with 2 boxes (Figure 5.5) was categorized as one object (row 4), but in row 3 we notice that there are 6 parts in the scene. The probability for 1 object consisting of 6 parts is below the 0.3 value according to the Poisson distribution and clearly indicates an over-segmentation error and the need for the robot to segment this region interactively.

5.4 Contact Point Estimation and Pushing

To perform object segmentation based on the individual object motions induced by the robot, appropriate contact points between the objects in the scene and the robot's end effector must be determined. Furthermore, the direction the robot's end effector should move must be chosen.

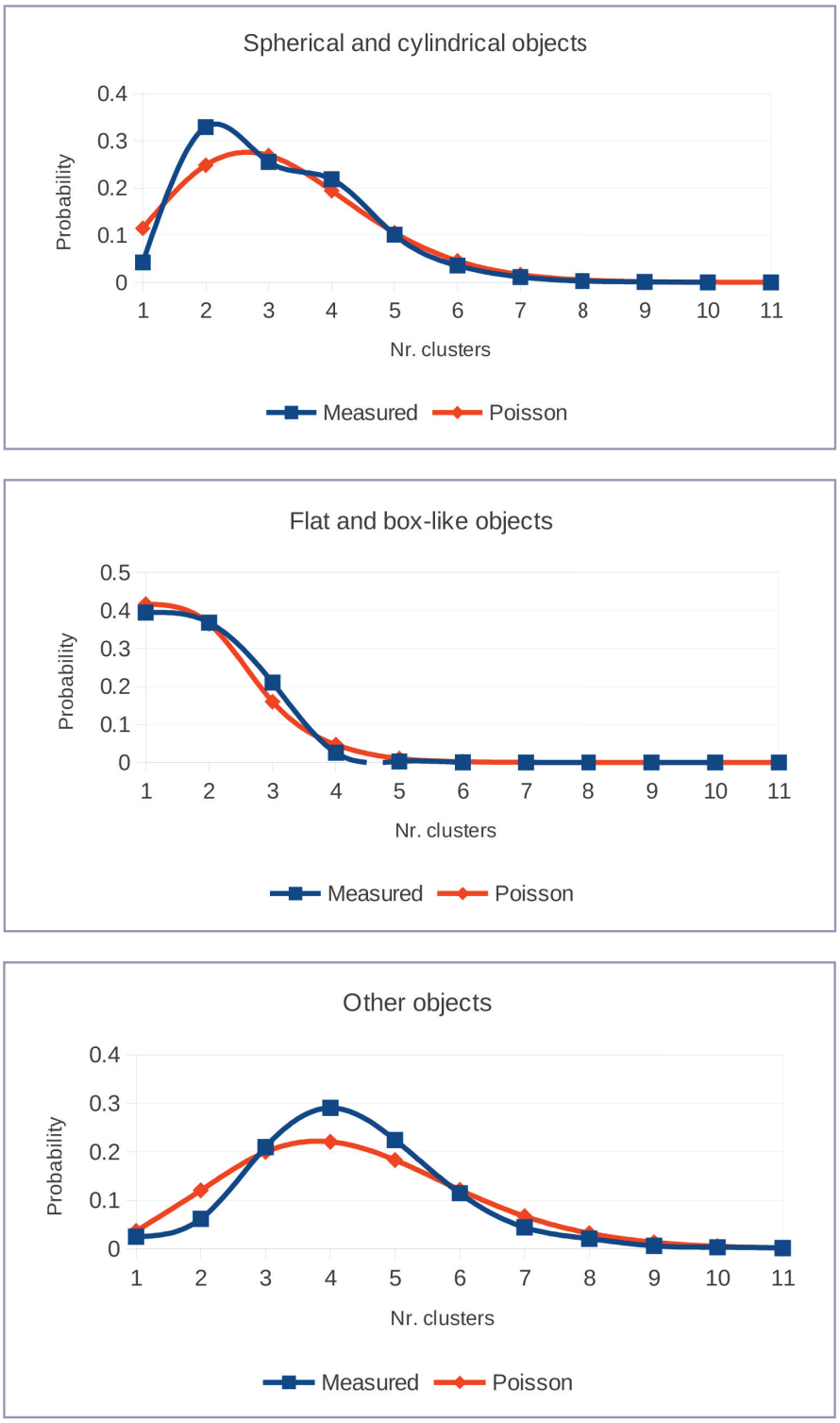


Figure 5.6: *Distribution of number of parts (see Figure 5.5 row 3) per object category and their approximation with a Poisson distribution. Courtesy@Marton.*

In this work, we consider a cluttered tabletop scene. Since most commonly encountered household items have convex outlines when observed from above [Marton et al., 2011], our system uses local concavities in the 2D contour of an object group as an indicator for boundaries between the objects. The robot separates objects from each other by pushing its end effector in between these boundaries. In the following, we describe a heuristic to determine a contact point and a push direction from depth-sensor data.

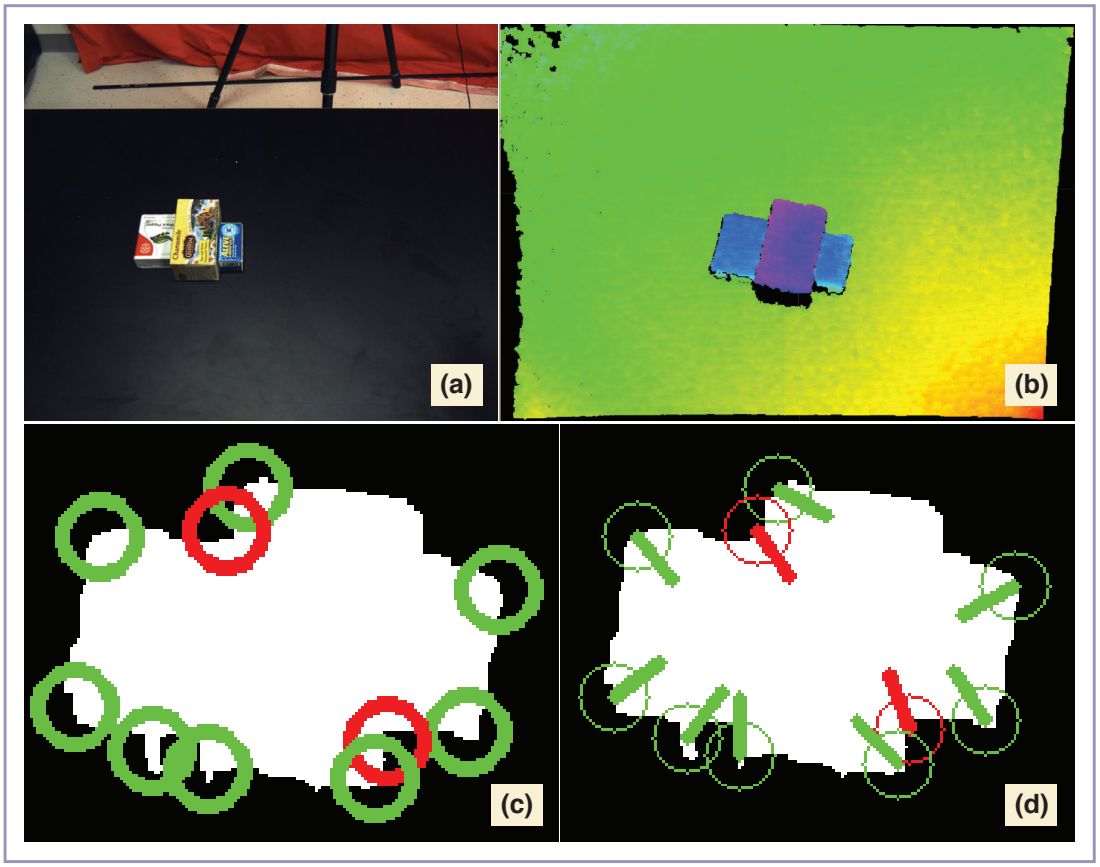


Figure 5.7: *Estimation of the contact point and the push direction. Top-left figure: original scene. Top-right figure: depth image as seen from the virtual camera positioned above the table. Bottom-left figure: Extracted contour of the object cluster, convex corners are shown in green, concave corners in red. Bottom-right figure: Direction of the dominant eigenvectors at the corners.*

5.4.1 Contact Points from Concave Corners

We restrict the problem of finding a contact point to the table plane. Our algorithm employs 2D-image processing techniques to select contact point candidates. The table plane is estimated from the depth-camera's point cloud data using RANSAC [Fischler and Bolles, 1981] and separated from the object points. The remaining cloud points are projected into a virtual camera view above the table. Since the projected cloud points are sparse, we employ standard morphological operators and 2D-contour search [Suzuki and Abe, 1985] to identify a closed region, R, corresponding to the group of objects. These steps are shown in Figure 5.7.

This region's outer contour is then searched for strong local directional changes by applying a corner detector and subsequently the corners that are placed at local concavities are selected. As in the Shi-Tomasi corner detector [Shi and Tomasi, 1994], we compute the corner response for each pixel location $\mathbf{p} = (p_x, p_y)$ based on the covariance matrix $\mathbf{Z}$:

$$\mathbf{Z}(\mathbf{p}) = \begin{bmatrix} \sum_{S(\mathbf{p})} I_x^2 & \sum_{S(\mathbf{p})} I_x I_y \\ \sum_{S(\mathbf{p})} I_x I_y & \sum_{S(\mathbf{p})} I_y^2 \end{bmatrix} \quad , \tag{5.2}$$

where I_x and I_y are the image gradients in x and y direction and $S(\mathbf{p})$ the neighborhood of $\mathbf{p}$. A corner detector response is recorded if $min(\lambda_1, \lambda_2) > \theta$, where λ_1,λ_2 are the eigenvalues of $\mathbf{Z}$ and θ is a given threshold. We also check for roughly equal eigenvalues, i.e. $\lambda_1 \approx \lambda_2$, ensuring that there are strong gradient responses in two approximately orthogonal directions. The local maxima of the smoothed corner responses, are the detected corners illustrated as circles in Figure 5.7.

The concavity of each corner is estimated using a small circular neighborhood. If a larger portion of this neighborhood is inside R rather than outside, the corner must be a concave part of R's contour and is shown red in Figure 5.7. This method effectively handles noise in terms of directional changes. Only the concave corners are considered contact point candidates, unless no corner is found fulfilling the above concavity criterion. This method computes potential contact points for only one group of objects, which the robot wants to break up for segmentation. If the scene contains multiple object groups the method will be applied to each group separately.

5.4.2 Push Direction and Execution

The push direction at a corner is set to be parallel to the eigenvector corresponding to the larger eigenvalue of the covariance matrix $\mathbf{Z}(\mathbf{p})$ in Equation 5.2. Intuitively, the dominant eigenvector will align with the dominant gradient direction. However, at a corner with two similar gradient responses in two directions, the eigenvector becomes the bisector. As only corners with roughly equal eigenvalues are chosen as potential contact point candidates, the eigenvector of each contact point candidate will bisect the angles of the contour at the corner location. as shown in Figure 5.7.

After determining the contact point candidates and the push directions in the 2D table plane, the end effector is moved within a constant small distance parallel to the table plane. A contact point is below an object's center of gravity and close to the friction vector between the object and the table, which avoids toppling over objects while pushing them. When there are multiple contact point candidates, the closest contact point to one of the end effectors and physically reachable by the robot arm, is selected.

5.4.3 Simulations

We carried out several simulations in the physics-based simulator Gazebo[4] to validate our corner pushing heuristic. To verify that pushing at corners is indeed more effective, we spawned different scenes in Gazebo with two or three closely placed objects. Objects were flat and round, in different orientations and arranged such that they were in solid contact or in single point contact. We then simulated pushing at these objects with a PR2 gripper at many different contact points along the bounding box of the objects and in many different directions. More precisely, we chose points along the bounding box spaced $2cm$ apart and for every such point, the gripper simulated a sequence of 2 pushes in 7 different directions 15° apart (See Figure 5.8). The starting gripper pose and the object poses before and after every push were recorded. Then Shi-Tomasi features with known but randomly determined locations were spawned artificially on the objects. Based on the recorded object poses, the locations of all the features were computed after every push. This enabled us to compute

[4] http://gazebosim.org/

the simulated optical flows. The feature trajectories so obtained were then clustered using Algorithm 1. The contact points for the pushes which resulted in a successful segmentation of objects were then observed and the number of successful corner pushes were counted. A push was classified as a corner push if the contact point is less than $1cm$ away from an object corner. We carried out 5 runs[5] on every of 24

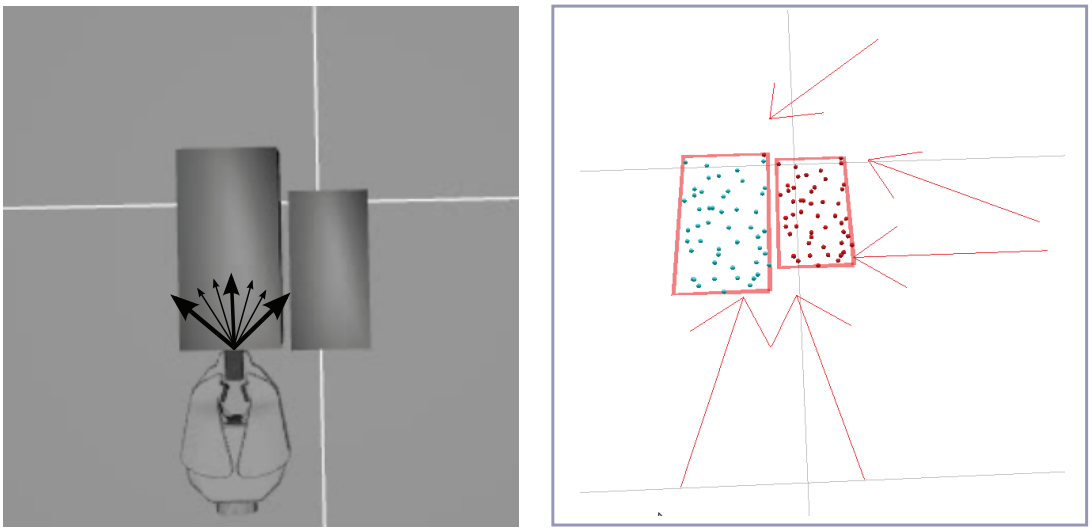


Figure 5.8: *Screenshots from the Gazebo simulation of a two object scene (left) and the corresponding visualization of the segmentation result. The black arrows in the left image show the 7 push directions for a single contact point. The dots on the objects in the right image represent features and their colors represent the cluster they were assigned to for a particular successful push sequence. The red arrows represent the starting gripper positions and directions of all the successful push sequences in a simulation run. Courtesy@Gupta.*

different scenes (11 scenes with 2 objects, 13 scenes with 3 objects), which resulted in an average of 381.5 push sequences for a scene out of which an average of 14.9 pushes were successful in segmenting all the objects in the scene correctly. Out of these, 7.25 pushes were corner pushes. There were an average of 10 object corners in each scene. From this it follows that there were on average 70 corner pushes and 311.5 other pushes. This gives the segmentation success of 10% for the corner pushes and 2.4% for the non-corner ones. The reason for the low overall segmentation result

[5]Please mind that since Gazebo uses an ODE engine which is based on linear complementarity problem constraint formulation and since the simulation is defendant on the CPU load, the runs are not fully deterministic.

is on the one hand in that the scenes in the simulation included the single contact points between the objects and on the other hand in that various non-favorable orientations per contact point were computed and executed. We observed that corner pushing was successful in all the scenes while side pushing was successful only when the objects were in single point contact. When the objects were next to each other and similar in size, pushing at the sides resulted in the objects moving together as a single rigid body, thus making the algorithm fail. In such cases, only corner pushes succeeded. These simulations thus prove the benefits of corner pushing irrespective of object arrangement.

5.5 Object Segmentation

Once the robot's end effector touches the objects, the resulting object motions are used to discriminate between the different items on the table. Features are tracked in the scene and the resulting feature trajectories are clustered. The clustering is based on the idea that features corresponding to the same objects must follow the same translations and rotations. Since the approach for feature clustering differs for textured and textureless objects we split the section in the following accordingly.

5.5.1 Textured Objects

The following assumptions with respect to textured objects were made:

- **Texture:** Each item has some texture over most of its surface, such that texture features can be used to appropriately represent an object for tracking.
- **Rigid Body:** Each item is a rigid body and not subject to deformations when interacting with the robot's end effector or other objects.

5.5.1.1 Feature Trajectory Generation using Optical Flow

We take advantage of the objects' texture properties by extracting $i = 1...N$ Shi-Tomasi features [Shi and Tomasi, 1994] at the pixel locations $\{\mathbf{p}_{i,0}\}_{i=1}^{N}$ from the ini-

tial scene at time $t = 0$, i.e. before an interaction with the robot took place. Feature locations are extracted using the same Shi-Tomasi feature detector as described in Section 5.4.1 but in this case on the 2D textured image (depicted in bottom row of Figure 5.1) When the robot's end effector interacts with the object, a Lucas-Kanade tracker [Bouguet, 2002] is used to compute the optical flow of the sparse feature set. Using the optical flow, each feature's position $\mathbf{p}_{i,t}$ is recorded over the image frames at time $t = 0...T$ while the robot is interacting with the objects. That is, for each successfully tracked feature i, a trajectory $S_i = \{\mathbf{p}_{i,t}\}_{t=0}^{T}$ is obtained. The features are tracked with the average time resolution of $1.047s$ which is the average time needed to process one image frame.

5.5.1.2 Randomized Feature Trajectory Clustering with Rigid Motion Hypotheses

After calculating the set of all feature trajectories $\mathscr{S} \equiv \{S_i\}_{i=1}^{N}$, the goal is to partition this set such that all features belonging to the same object are assigned the same object index $c_i \in \{1,..,K\}$, where the number of objects K is not known *a priori*.

In other work on moving object segmentation, clustering has been applied directly to optical flow vectors [Klappstein et al., 2008; Brox and Malik, 2010]. However, in this context, where the robot induces the motion, the objects tend to be subject to strong rotational motions, which cause strongly non-collinear optical flow vectors. Instead, we take advantage of the rigid body property of objects and assume that each subset of the features trajectories $\mathscr{S}$ belonging to the same object k are subjected to the same sequence of rigid transformation $A_k \equiv \{\mathbf{A}_{k,t}\}_{t=0}^{T-1}$, i.e. we cluster features with respect to how well rigid transformations can explain their motions. As the objects only move on the table plane, we restrict a possible rigid transformation $\mathbf{A}$ to be composed of a 2D-rotation $\mathbf{R}$, a 2D-translation $\mathbf{t}$ and a scaling component s, i.e. $\mathbf{A} = s \cdot [\mathbf{R}|\mathbf{t}]$. The scaling component compensates for the changes in size of the projected objects in the camera image. The actual scaling is not linear due to the perspective view, however, the error resulting from this linearization is small as the objects are displaced only in small amounts.

The clustering algorithm we propose is outlined in Algorithm 1, and combines a divisive clustering approach with RANSAC-style model hypothesis sampling. At the core

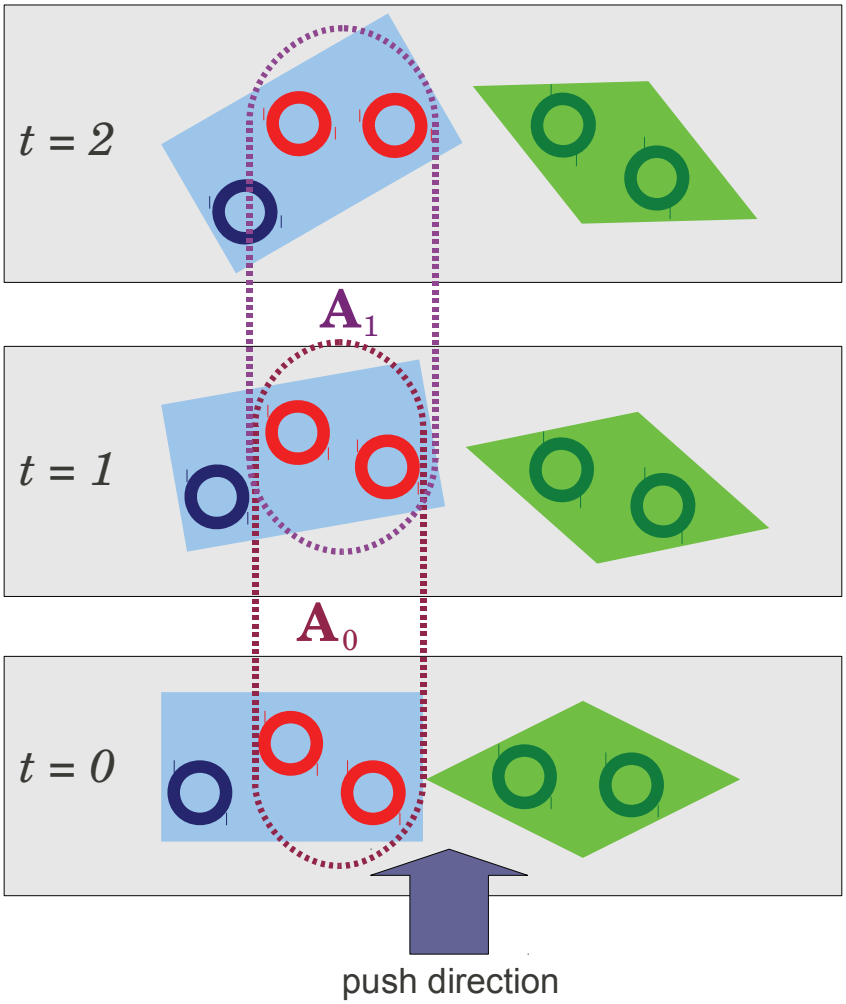


Figure 5.9: *Feature trajectory clustering with rigid motion hypotheses: Each feature i, depicted as a circle, is tracked over each time step t, forming a trajectory of feature positions S_i. After the robot finished its push motion, two features u and v, depicted as red circles, are randomly selected. From their trajectories S_u and S_v, a rigid transformation $\mathbf{A}_{k,t}$ is calculated that represents the rigid motion of u and v for each time increment from t to $t+1$. If u and v are on the same object, all other features will move according the sequence of rigid transformations $A_k = \{\mathbf{A}_{k,t}\}_{t=0}^{T-1}$, which serves as the rigid motion hypotheses for an object (e.g. the blue box). As the dark blue feature belongs to the same object as u and v, its motion can be explained by this motion hypothesis, and will thus be assigned to the same object. The motions of the dark green features located on a different object are poorly modeled by this motion hypothesis, and thus trigger the algorithm to create another motion hypothesis.*

of the algorithm (lines 4–12, see also Figure 5.9), we randomly draw 2 tracked features u,v and estimate a sequence of rigid transformations $A_{1,t}$ from their optical flow motions as first model hypothesis. The feature trajectories S_i that can be explained well by $A_{1,t}$ are considered "model inliers" and are removed from set of feature trajectories. From the remaining set, again 2 features are drawn to create a second model hypothesis $A_{2,t}$ and all inliers are removed. This process repeats until there are not enough features left to create a new model hypothesis. This process results in K hypotheses.

Algorithm 1: Randomized feature trajectory clustering. Mind that for the sake of clarity we do not write out the subscript m in the text explaining this algorithm.

1 Input: Set of feature trajectories $\mathscr{S} \equiv \{S_i\}_{i=1}^{N}$ where $S_i = \{\mathbf{p}_{i,t}\}_{t=0}^{T}$
2 Output: object cluster count K, object cluster assignments $\mathbf{c} = \left[c_i\right]_{i=1}^{N}$ where $c_i \in \{1,..,K\}$
3 **for** $m := 1$ *to* M **do**
4 $\quad k_m := 1$, $\mathscr{S}_m := \mathscr{S}$
5 $\quad$ **while** $|\mathscr{S}_m| \geq 2$ **do**
6 $\quad\quad$ draw 2 random trajectories $S_u, S_v \in \mathscr{S}_m$
7 $\quad\quad$ generate sequence of rigid transformations: $A_{k_m} \equiv \{\mathbf{A}_{k_m,t}\}_{t=0}^{T-1}$ from (S_u, S_v)
8 $\quad\quad$ **for** S_j *in* $\mathscr{S}_m$ **do**
9 $\quad\quad\quad$ sum squared residuals w.r.t to A_{k_m}: $r_{k_m,j} := \sum_{t=0}^{T-1} \|\mathbf{p}_{j,t+1} - \mathbf{A}_{k_m,t}\mathbf{p}_{j,t}\|_2^2$
10 $\quad\quad\quad$ **if** $r_{k_m,j} < THRESHOLD$ **then**
11 $\quad\quad\quad\quad \mathscr{S}_m := \mathscr{S}_m \setminus \{S_j\}$
12 $\quad\quad k_m := k_m + 1$
13 $\quad K_m := k_m$
14 $\quad$ **for** S_i *in* $\mathscr{S}$ **do**
15 $\quad\quad$ Assign each trajectory to best matching rigid transformation sequence: $c^*_{m,i} := \text{argmin}_{\{1,..,k_m,..,K_m-1\}}\, r_{k_m,i}$, where $r_{k_m,i} := \sum_{t=0}^{T-1} \|\mathbf{p}_{i,t+1} - \mathbf{A}_{k_m,t}\mathbf{p}_{i,t}\|_2^2$
16 Select best overall matching set of rigid transform sequences: $m^* := \text{argmin}_m \sum_{k_m=1}^{K_m} \frac{\sum_i r_{k_m,i} \cdot \mathbf{1}_{\left[c^*_{m,i}=k_m\right]}}{\sum_i \mathbf{1}_{\left[c^*_{m,i}=k_m\right]}}$
17 Return: $K := K_{m^*}$, $\mathbf{c} := \left[c^*_{m^*,i}\right]_{i=1}^{N}$

We bias the sampling of the 2 points (line 6) such that drawn feature pairs are not likely to be further apart than the object size $OS = 0.1m$ that the robot can grasp. For this, the first feature u is chosen uniformly and the probability p for choosing

a feature i as the second point is proportional to the normalized Gaussian density function of the distance between $\mathbf{p}_i$ and $\mathbf{p}_u$:

$$p(i) \propto \exp\left(-\frac{\| \mathbf{p}_i - \mathbf{p}_u \|_2^2}{2\sigma^2}\right), \tag{5.3}$$

where σ is set to half of the object size OS in image space.

In line 7, a rigid transformation $\mathbf{A}_{k,t}$ is computed from the trajectories S_u and S_v at each time increment from t to $t+1$. A 4-DOF transformation $\mathbf{A}_{k,t}$ can be computed directly using geometrical considerations outlined by Horn [1987] from the two 2D-feature point locations $\mathbf{p}_u$ and $\mathbf{p}_v$ at t and $t+1$, such that:

$$\mathbf{p}_{u,t+1} = \mathbf{A}_{k,t}\mathbf{p}_{u,t} \quad \text{and} \quad \mathbf{p}_{v,t+1} = \mathbf{A}_{k,t}\mathbf{p}_{v,t} \quad . \tag{5.4}$$

We use the sum of squared residuals over all time increments, $r_{k,j} = \sum_{t=0}^{T-1} \|\mathbf{p}_{i,t+1} - \mathbf{A}_{k,t}\mathbf{p}_{i,t}\|_2^2$, as a measure of how well a feature trajectory S_i fits a transformation sequence A_k, where each residual is the difference vector between the actual feature location $\mathbf{p}_{i,t+1}$ and $\mathbf{A}_{k,t}\mathbf{p}_{i,t}$, the feature location predicted by $\mathbf{A}_{k,t}$. This measure is used to discriminate between inliers and outliers for the model generation process (line 10), as well as for the best assignment c_i^* of a trajectory to a model hypothesis (line 15). Note that the final assignment may not necessarily correspond to the model hypothesis for which the trajectory was an inlier if a later generated hypothesis explains its motions better. Furthermore, using a model to predict the features' motions at each time step, as compared to considering only the total movement induced by the robot, is effective at discriminating between objects that start or stop moving at different time steps during the robot interaction.

As each trajectory pair that is used for the model hypothesis generation is chosen in a randomized fashion, it can happen that a pair of features is chosen such that they are not on the same object. This can cause an erroneous partitioning process of the feature trajectory set, resulting in wrong model hypotheses. However, this problem can be overcome with a high probability by sampling from the whole hypotheses generation process M-times, where each set of model hypotheses is indexed by the iteration m in Algorithm 1. This is explained in detail in Section 5.5.1.3. We choose the best m according to the score function (line 16), which is the sum of summed

squared residuals between each trajectory and its best matching model hypothesis, normalized by the number of feature trajectories assigned to this hypothesis. This measure thus favors sets of hypotheses that predict a smaller number of objects and where the number of features per object is roughly equal. Often erroneous hypotheses are only supported by few outlier features and are suppressed.

Even though we used the clustering algorithm for the clustering of the 2D features, the algorithm scales to 3D space as well.

5.5.1.3 Trajectory Clustering Complexity Analysis

Instead of sampling M-times the trajectory model generation process, one could generate a model hypothesis for all $\binom{N}{2} \approx \frac{N^2}{2}$ possible feature pairs (N - number of all extracted features as above), as done in quality-threshold clustering [Heyer et al., 1999]. However, for a small maximal number of objects K, M can be small such that the computational complexity of our algorithm is much lower as shown in the following.

The probability that a draw of a pair of trajectories $\mathscr{S}_u$, $\mathscr{S}_v$ (line 7) is the result of the motion of the same object can be approximated if we neglect the bias in Equation 5.3 and instead assume uniformly random draws from all detected feature trajectories. Given the true number of objects K and the number of feature trajectories N_k on an object $k \in \{1,...,K\} := \mathscr{K}$, the probability to select any 2 feature trajectories from the same object in the first draw $w = 0$ is

$$P_{k,0}(\text{2 traj. on object } k) = \frac{N_k \cdot (N_k - 1)}{N \cdot (N-1)} \quad . \tag{5.5}$$

Thus the probability to have 2 feature trajectories selected on *any* of the K objects together in the first draw $w = 0$ is:

$$P_0 = \sum_{k \in \mathscr{K}} \frac{N_k \cdot (N_k - 1)}{N \cdot (N-1)} \approx \sum_{k \in \mathscr{K}} \left(\frac{N_k}{N} \right)^2, \quad N \gg 1 \quad . \tag{5.6}$$

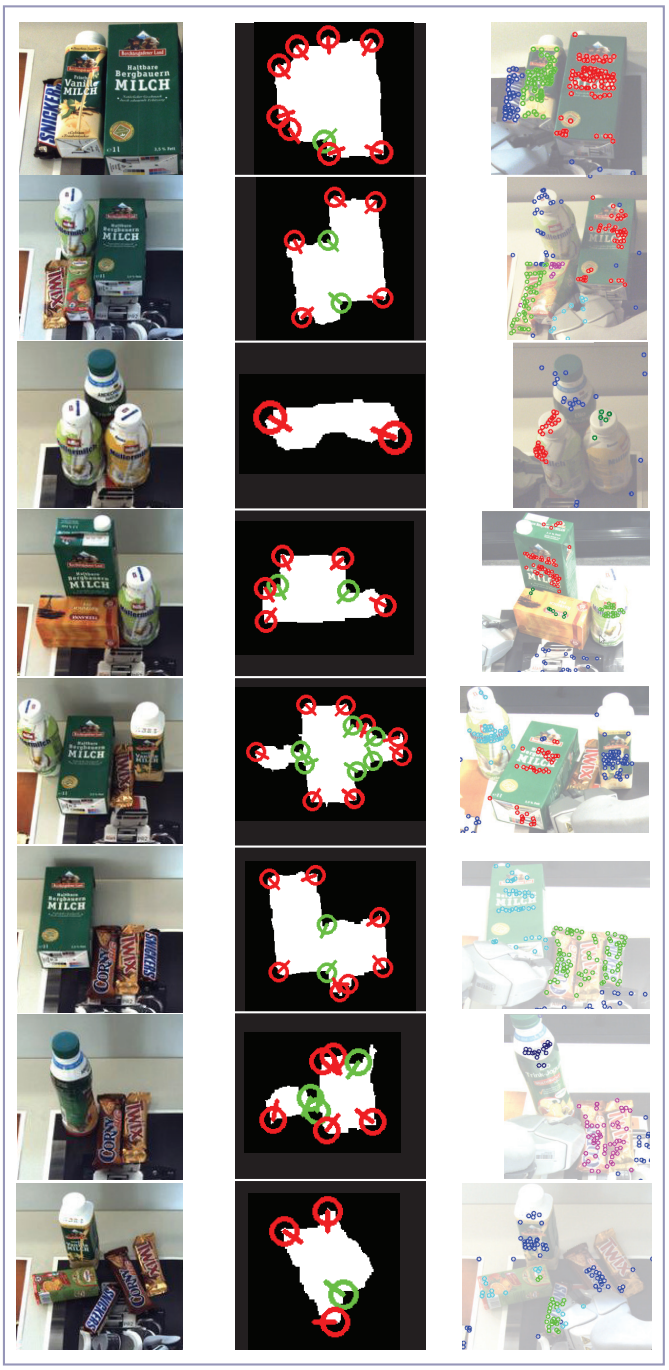


Figure 5.10: *Test scenes 1 to 8 from top to down. Left column: original scenes, middle column: contact point estimation, right column: segmentation after the first push cycle. Please note, that the 2D contours in the middle column are generated from the virtual view above the table which may slightly change the perspective. Features on the objects that did not get pushed are in the same cluster (denoted by the same color) as background features.*

If we assume that N_k is the same for all objects, i.e.
$N_k := \bar{N}_k = \frac{N}{K}$, this simplifies to $P_0 = \frac{1}{K}$.

Once a draw is made, all trajectories matching the model hypothesis defined by the draw are assigned to that model and are removed from $\mathscr{S}_m$. The next draw $w = 1$ is made from the remaining $N - N_k$ trajectories of the remaining $K - 1$ objects, given that the trajectory assignment (line 10-11) discriminates sufficiently between the trajectories belonging to that model and those that do not. Analogously the probability for drawing two feature trajectories from any of the remaining objects in draw w, conditioned that all previous draws were correct, is:

$$P_w = \sum_{k \in \mathscr{K} \setminus \mathscr{K}^-} \left(\frac{N_k}{N - \sum_{k' \in \mathscr{K}^-} N_{k'}} \right)^2 \overset{N_k = \bar{N}_k}{=\!=} \frac{1}{K - w} \quad , \tag{5.7}$$

where $\mathscr{K}^-$ is the set of modeled objects whose trajectories have been removed from $\mathscr{S}_m$.

The probability that for all draws the drawn feature trajectory pairs are together on an object thus is:

$$P = \prod_{w=0}^{K-1} P_w = \frac{1}{K!} \quad . \tag{5.8}$$

The probability P directly corresponds to the inlier probability in RANSAC. Thus, one can similarly estimate the number of times M that above drawing process should be executed in order to find a set of correct motion model hypotheses with a given probability $\alpha < 1$. That is, the algorithm returns a good segmentation with probability α. As $1 - \alpha = (1 - P)^M$ is the probability that in none out of M drawing processes all K drawn trajectory pairs were together on an object, M can be calculated as:

$$M = \frac{\log(1 - \alpha)}{\log(1 - P)} = \frac{\log(1 - \alpha)}{\log(1 - \frac{1}{K!})} \quad . \tag{5.9}$$

For example for $K = 4$ objects and $\alpha = 0.95$, $M = 72$ sampling runs are required.

However, in our experiments, $M = 20$ proved to be sufficient, since we bias the drawing of trajectories towards features that are initially close to each other, such that the probability P_w is higher than in random sampling. It is important to notice that this bias as controlled by σ in Equation 5.3 may not be too strong, as close-by feature pairs decrease the accuracy when estimating a motion model for an object from their trajectories. This is because the feature displacements are tracked only with pixel precision such that the rotation estimate of close-by feature pairs is subject to larger noise, which can result in splitting an object in the segmentation.

5.5.2 Textureless Objects

In this subsection we describe the selected RGBD features suitable for the tracking of textureless objects and the particle filtering-based tracking library. The features are estimated on the above (Section 5.3.2) classified list of object hypotheses O from the RGBD point cloud. RGB and the depth measurements in the point cloud are time synchronized and registered. We employ 3D circle and 3D cylinder point cloud features for the round objects and 3D line and 3D corner point cloud features for the flat objects. The rationale behind this selection of features is that they are all fast to compute and yet distinctive enough for tracking with the proposed tracking algorithm. The latter uses a combination of the visual appearance and the geometrical structure of the feature to compute the likelihood function of the feature hypothesis.

5.5.2.1 RGBD Features

In order to obtain a 3D line point cloud we first find object edge candidates in the cluttered scene using curvature values computed in the input point cloud from the Kinect sensor. Next we fit a line model to the object edge candidates using RANSAC [Fischler and Bolles, 1981] and finally pad the line with neighboring points on the object within a radius of $5cm$. 3D corner point clouds are determined using the 3D variant of the Harris corner detector as implemented in the Point Cloud Library (PCL)(pointclouds.org) and padded with neighboring points on the object within a radius of $5cm$ as well. Padding of both features is necessary in order to guarantee

computation of a better likelihood function needed by the tracker as explained in the following subsection. The features are shown in Figure 5.5 rows 2 and 5.

To obtain a 3D cylinder point cloud, we also use a RANSAC model which is based on the fact that on a cylinder surface, all normals are both orthogonal to the cylinder axis and intersect it. We consider the two lines defined by two sample points and their corresponding normals as two skew lines, and the shortest connecting line segment as the axis. Determining the radius is then a matter of computing the distance of one of the sample points to the axis. By setting the cylinder axis perpendicular to the table results are more robust, but is not mandatory. Finally, the generation of the 3D circle is also done using RANSAC by projecting a sample point into the 3D circle's plane and computing the distance between this point and the point obtained as an intersection of the line from the circle's center with the circle's boundary, whereas the line is passing through the projected sample point. The features are shown in the right column of Figure 5.5 rows 2 and 5.

5.5.2.2 Particle Filtering-based Tracking of RGBD Pointclouds

The feature point clouds extracted above are then passed to the particle filter-based tracker as reference models. The tracker consists of four steps: i) the above described reference model selection (RGBD features), ii) pose update and re-sampling, iii) computation of the likelihood and iv) weight normalization. In the pose update step we use a ratio between a constant position and a constant velocity motion model which allows us to achieve efficient tracking with a lesser number of the particles. In the re-sampling phase we utilize Walkers Alias Method [Walker, 1977]. The likelihood function l_j of the hypotheses in the third step is computed as in Equation 5.10 and is based on the similarity between the nearest points pair of the reference point (p_j) cloud and the input data (q_j). Similarity is defined as a product of a term describing the points pair's euclidean distance $l_{euclidean}$ and a term describing points pair's match

in the HSV (Hue, Saturation, Value) color space l_{color}. α and β are the weight factors set to 0.5 in our case.

$$\begin{aligned} l_j &= l_{euclidean}(p_j, q_j) l_{color}(p_j, q_j) \\ l_{euclidean}(p_j, q_j) &= \frac{1}{1 + \alpha |p_j - q_j|^2} \\ l_{color}(p_j, q_j) &= \frac{1}{1 + \beta |p_{j,hsv} - q_{j,hsv}|^2} \end{aligned} \tag{5.10}$$

To obtain the model's weight we sum over likelihood values for every points pair in the reference model as follows: $w_i = \sum_j l_j$. This likelihood function assures a combined matching of model's structure and visual appearance. In the final step we normalize the previously computed model weight by applying a relative normalization as described by Azad et al. [2011]. The real-time operation of the algorithm is made possible through various optimization techniques such as downsampling of the point clouds, openMP parallelization and KLD-based (Kullback-Leibler Divergence) sampling [Fox, 2001] to select the optimal number of particles.

Why not to track object parts? To answer this question we refer the reader to scene 1 in Figure 5.5, row 3 where top surfaces of both boxes were grouped into one segment. Had we taken this segment as a reference cloud the tracking algorithm would fail due to its limitation to generate multiple reference clouds during tracking.

5.5.2.3 Trajectory Clustering

The tracked features' 3D trajectories (see Figure 5.5 row 6) are clustered using Algorithm 2 in order to find the feature-object associations. We treat each of the n RGBD features as a node in a graph, where edge weights represent the maximum number of consecutive violations of the relative distance variation threshold ($d_{threshold}$), i.e. *breaks* (optionally, also pose changes can be checked for better performance). The final connection matrix is obtained by removing the edges which have weights that exceed a given percentage ($p_{threshold}$) of the theoretic maximum number of frames. The distance between features which did not vary are then clustered together.

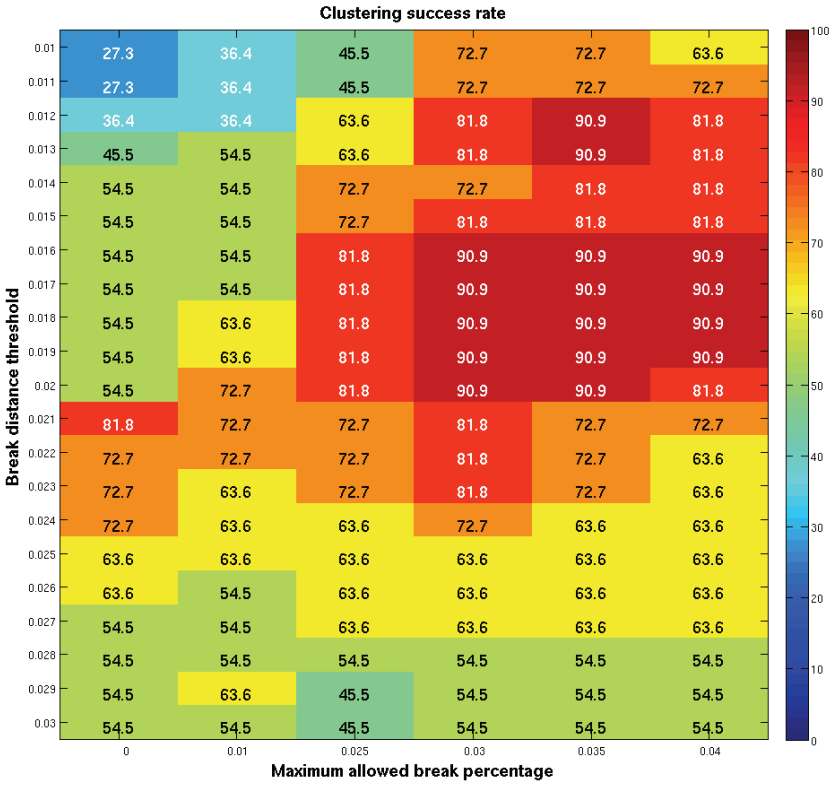


Figure 5.11: *Clustering success rate on 17 scenes for different values of $p_{threshold}$ (maximum allowed break percentage) as a function of $d_{threshold}$ (break distance threshold).*

Algorithm 2: Graph-based trajectory clustering algorithm. A *break* between features means that the relative distance between them exceeded the given threshold. Courtesy@Marton.

/* number of tracked features n and number of time steps m, relative distance variation threshold $d_{threshold}$, max allowed percent of consecutive breaks $p_{threshold}$, and the set of positions of each feature T */
Input: n, m, $d_{threshold}$, $p_{threshold}$, $T = \{t_1 ... t_m\}$
/* relative distances at t_1 */
1 $D_{reference} = \text{pairwiseL2}(t_1)$
/* nr of consecutive breaks between features */
2 $C_{breaks} = \text{zeros}(n,n)$
/* relative distances at t_1 */
3 $T_{breaks} = \text{zeros}(m,n,n)$
/* count number of consecutive breaks */
4 **foreach** $t_i \in T$ **do**
  /* relative distances at t_i */
5 $D_i = \text{pairwiseL2}(t_i)$
  /* deviation of distances */
6 $E_i = |D_i - D_{reference}|$
  /* breaking feature pairs */
7 $B_i = \{(f_1, f_2) | E_i[f_1, f_2] > d_{threshold}\}$
8 **foreach** $(f_1, f_2) \in B_i$ **do**
9 $C_{breaks}[f_1, f_2]++$ /* increment counter */
10 **foreach** $(f_1, f_2) \notin B_i$ **do**
11 $C_{breaks}[f_1, f_2] = 0$ /* reset counter */
12 $T_{breaks}[i] = C_{breaks}$ /* save counter */
/* maximum percentage of consecutive breaks */
13 $M_{breaks} = \max(T_{breaks})/m$
/* final adjacency matrix */
14 $A = \text{getConnections}(M_{breaks} <= p_{threshold})$
/* number of clusters based on Laplacian */
15 $nr_{clusters} = \text{nrZeros}(\text{eigenValues}(\text{diag}(\text{degrees}(A)) - A))$
/* get features clustered by connectivity */
Output: $F_{clusters} = \text{connectedComponents}(A)$

Figure 5.11 shows an evaluation of the clustering algorithm on 17 scenes from Figure 5.14. The use of $p_{threshold}$ is clearly advantageous, and the method works well for a range of the $p_{threshold}$ and the $d_{threshold}$ parameters. Since too low values for $d_{threshold}$ over-segment the features, values over $1.5cm$ are used, and the possible under-segmentations solved by applying the whole method iteratively until all the objects are clearly separated.

5.6 Dense Model Reconstruction

In order to generate useful input for object recognition or object grasping, we in the end densely reconstruct the segmented objects. Considering the connected features $c_i \in \{1, .., K\}$ in case of textured objects and $F_{clusters}$ in case of textureless objects as being part of the same object, we reconstruct the dense model of the object using region growing in normal space, which also makes use of the borders found at depth discontinuities, as shown in Algorithm 3. The idea for the region growing constraints is based on the segmentation described by Mishra and Aloimonos [2009], where the authors make use of a predefined fixation point and a border map. Since we already know the features that are part of the object, we can easily define a seed point for the region growing. In order to find the best possible seed point, we separate the connected features using euclidean clustering, calculate each of the resulting clusters' centroid, and then start growing from these. An important condition of the region growing is the assumption that objects are often composed of convex parts [Jacobs, 2001]. Therefore, we make sure that during region growing two points are assigned to the same region R_i if the angle eps_{thresh} between the vector connecting them and the points normal is close to obtuse (considering the sensor noise level [Willow-Garage, 2010a] 89° were used). Once all region-feature pairs have been identified, we reconstruct the dense model. Since in the trajectory clustering step we already identified the features that belong to the same object, having multiple regions for the same object is easily dealt with by merging those regions for which the corresponding features belong to the same object into dense models R_j.

5.7 Results

5.7.1 Textured Objects

We evaluated the segmentation of textured objects on real scenes using PR2 robot. Depth and RGB images were taken from a Kinect sensor mounted on the robot's head (See Figure 5.1).

Algorithm 3: Region growing with normals & boundaries. Courtesy@Balint-Benczedi.

```
/* set of features F_clusters or c_i, distance threshold d_roi_thresh, angle threshold
   eps_thresh, seed queue sq, regions list R, current region R_i, list of
   processed points processed                                                    */
Input: F_clusters or c_i, d_roi_thresh, eps_thresh
foreach f_i ∈ F_clusters or c_i do
    p_s,i := centroid(f_i) sq.add(p_s,i) /* select a seed point and add it to a queue */
    processed(p_s,i) = true
    R_i := {p_s,i} /* initialize region                                          */
    while sq.notempty() do
        N := {q_j || dist(q_j, R_i[c]) < d_roi_thresh}
        /* select neighborhood                                                   */
        foreach q_j ∈ N do
            if processed(q_j) = true then
                continue
            if boundary(q_j) = true then
                stopgrowing = true
                R_i ← R_i ∪ {q_j} processed(q_j) = true
                break
            if deg(p_s,i q_j, norm(q_j)) > eps_thresh then
                R_i ← R_i ∪ {q_j}
                processed(q_j) = true
            else
                break
        if stopgrowing = false && ∀q_j ∈ N boundary(q_j) = false then
            sq ← N
    R ← R_i
foreach R_i, R_j ∈ R do
    if f_i f_j ∈ same object then
        R_i ← R_i ∪ {R_j}
```

Output: Dense models R_j

5.7.1.1 Random vs. Corner-based Pushing

We arranged eight tabletop scenes with the cluttered background shown in Figure 5.10 and evaluated the segmentation success rate given corner-based pushing and random pushing. In the latter mode we randomly sample poses from the set of reachable poses along the object pile contour. In the corner-based pushing experiments we also evaluated the correctness of detected contact points and push directions. After contact point and direction initialization, the robot entered into a push-

	Scene 1				Scene 2				Scene 3			Scene 4			Scene 5				Scene 6				Scene 7			Scene 8				
	1	2	3	4	1	2	3	4	1	2	3	1	2	3	1	2	3	4	1	2	3	4	1	2	3	1	2	3	4	5
Corner pushing	3	4,3	5	7	4,7	5	6,7	7,7	4	5	6,3	2,7	3	7	2,7	5	5	6,3	2,7	6	7,7	8,7	2	10	10	2,3	4,7	9,3	10	10
Random pushing	4,7	5	7,3	7,3	4,7	5,3	9,7	9,7	2	4	5,7	2	5,3	8,7	4,7	5,3	7	7,7	3	5,7	7,7	10	4	10	10	3,7	6	8,3	10	10

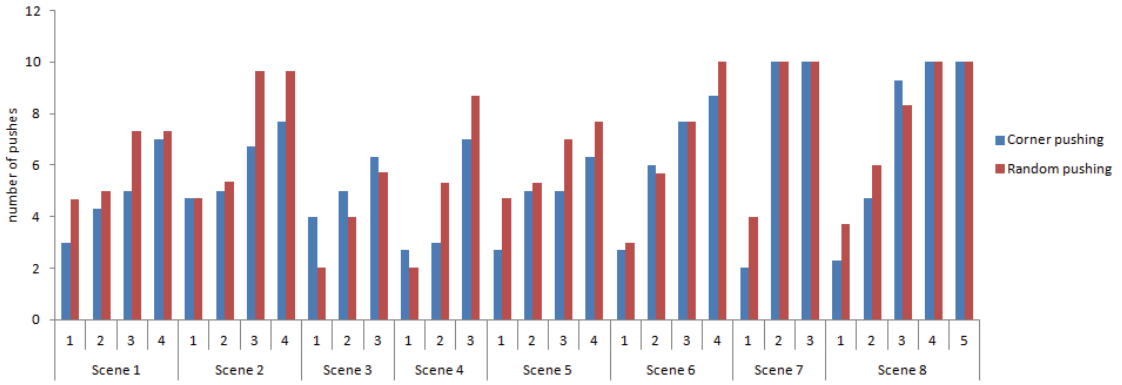


Figure 5.12: *Results of the segmentation of objects depicted in Figure 5.10 using random vs. corner-based pushing. The tabular (upper) part of the figure denotes the average number of pushes over 3 runs needed to segment the respective object in the respective scene. Number 10 (maximum number of pushes allowed) means the robot failed to segment the object. The same statistics is also depicted as a bar chart in the bottom part of the Figure for clarity. X-axis represents the scene and the object number, Y-axis the number of pushes.*

cluster cycle until one or more objects were successfully segmented (the number of pushes was not exclusive to single object) or we reached a maximum 10 number of pushes. For every push the robot's end-effector traveled for $1cm$ along the push direction.

For each of the scenes we carried out three segmentation runs and present the average results in Figure 5.12. In every run the success of the segmentation was inspected visually by the human operator and the objects were removed from the scene by the operator upon the successful segmentation. In all 24 runs for the corner-based pushing all the contact points and the push directions were successfully chosen and executed. As shown in Figure 5.12 an informative, corner-based pushing results in a faster and more reliable segmentation of objects. Using random pushing, there were 10 runs in which the robot failed to segment the whole scene (total of 26 unsegmented objects). For the corner-based pushing we only had 3 unsegmented scenes and total of 10 non-segmented objects, while exerting 0.6 pushes less on average as in the case of random pushing. Across all runs using corner-based pushing 89% of all objects were segmented successfully. The most frequent source of failures for the random pushing is depicted in the right part of Figure 5.15 where the push motion was induced such that all objects moved rigidly with respect to each other. In terms of average computational costs per scene, corner detection and direction estimation took $0.2s$, arm to corner movement $5s$ and Shi-Tomasi feature extraction $0.1s$. Every next push took $0.3s$ and the clustering of every next image frame requires $1.047s$. The total average time to segment 1 scene was $\sim$1 minute.

5.7.1.2 Grasping

We ran a grasping experiment on the scene 8 (Figure 5.10) and use an associated point cloud for the calculation of the object pose. To compute the latter we take the set of 3D points corresponding to the the set of 2D features from the successfully segmented cluster and apply an Euclidean clustering to remove possible outliers. We then compute the 3D centroid to obtain the object position and then use the Principle Component Analysis to compute the orientation. The result of this experiment is presented in a video [6].

[6] http://youtu.be/4VVov6E3iiM

5.7.2 Textureless Objects

The system for segmentation of textureless objects was evaluated on 17 scenes in different configurations as illustrated in Figure 5.14. The scenes are numbered 1-17 and arranged according to the legend shown in Figure 5.13. Though our system can iteratively cope with multi-object scenes, we performed the evaluation on two-object scenes with the finite number of scene configurations that can occur. These configurations can be split in three different ways, namely: i) size, ii) shape, and iii) arrangement. A scene may consist of two objects of different sizes or the same size. The objects may be either both flat or round or a combination of these two. They may also occur in different arrangements; completely separated, only touching, one on top of the other, or in solid contact. Solid contact refers to both objects being in contact with each other, whereby the contact area is larger than a single line (scene number 4 in Figure 5.14). Some configurations are infeasible for our approach. For example a flat object and a round object cannot be of the same size, or round object on top of another round object cannot be pushed (one mug on top of another mug). It is also not possible to have a round object that is in solid contact with another round object. For this case we consider solid contact as being two objects touching with more than one line, for example in scene number 17 where also the handle of the mug touches the juice box.

It is important to emphasize that the above devised conventions refer to the scenes after a push. The scenes before interaction were designed such that it is difficult or impossible to segment them using static segmentation techniques.

Average time to segment one scene from Figure 5.14 amounted to 12.5s with the pre-segmentation taking 1.5s, feature extraction 3.5s, pushing 6s (tracking runs at 25fps for up to 10 features) and dense model reconstruction 1.5s. Apart from tracking all modules perform linearly with the number of features and objects respectively and can thus easily be used for larger and more complex scenes. For all the scenes the push point estimation algorithm was used, the only exception being the 'on top' arrangements for which the algorithm does not generalize. For this reason and since the scope is on the priors from the static segmentation, RGBD features for textureless objects and the final dense model reconstruction, we performed the experiments by

	arrangement	shape		
		flat-flat	round-round	round-flat
different size	separated	1	9	14
	touching	2	10	15
	on top	3	11	16
	in solid contact	4	-	17
same size	separated	5	12	-
	touching	6	13	-
	on top	7	-	-
	in solid contact	8	-	-

Figure 5.13: *Legend for the different scene configurations. The scenes are shown in Figure 5.14.*

manually inducing motions into the corners of the scenes. In our future work we will address finding a generalized push point algorithm.

All the experiments were performed three times for each of the 17 scenes. All the results are presented in Table 5.1 which shows the segmentation success rate for every scene. The corresponding figures for this data can be found in Figure 5.14. The algorithm was never able to segment the scene number 8 and performed poorly for scenes 6 and 13. In these cases the contact surface of the two objects is large and the objects are of the same size. Erroneous reconstruction happens due to a lack of a sufficiently good boundary estimation near the touching surface, and therefore the region growing does not terminate. This could be alleviated by integrating texture/color-based segmentation methods, which we plan to investigate in the future.

It is important to note that the overall segmentation was successful in more than 82% of the experiments. Table 5.2 shows that the more objects differ and the less in contact they are the more successful the segmentation becomes. Our algorithm performs extremely well in the 'on top' arrangement which is very challenging for the static segmentation techniques.

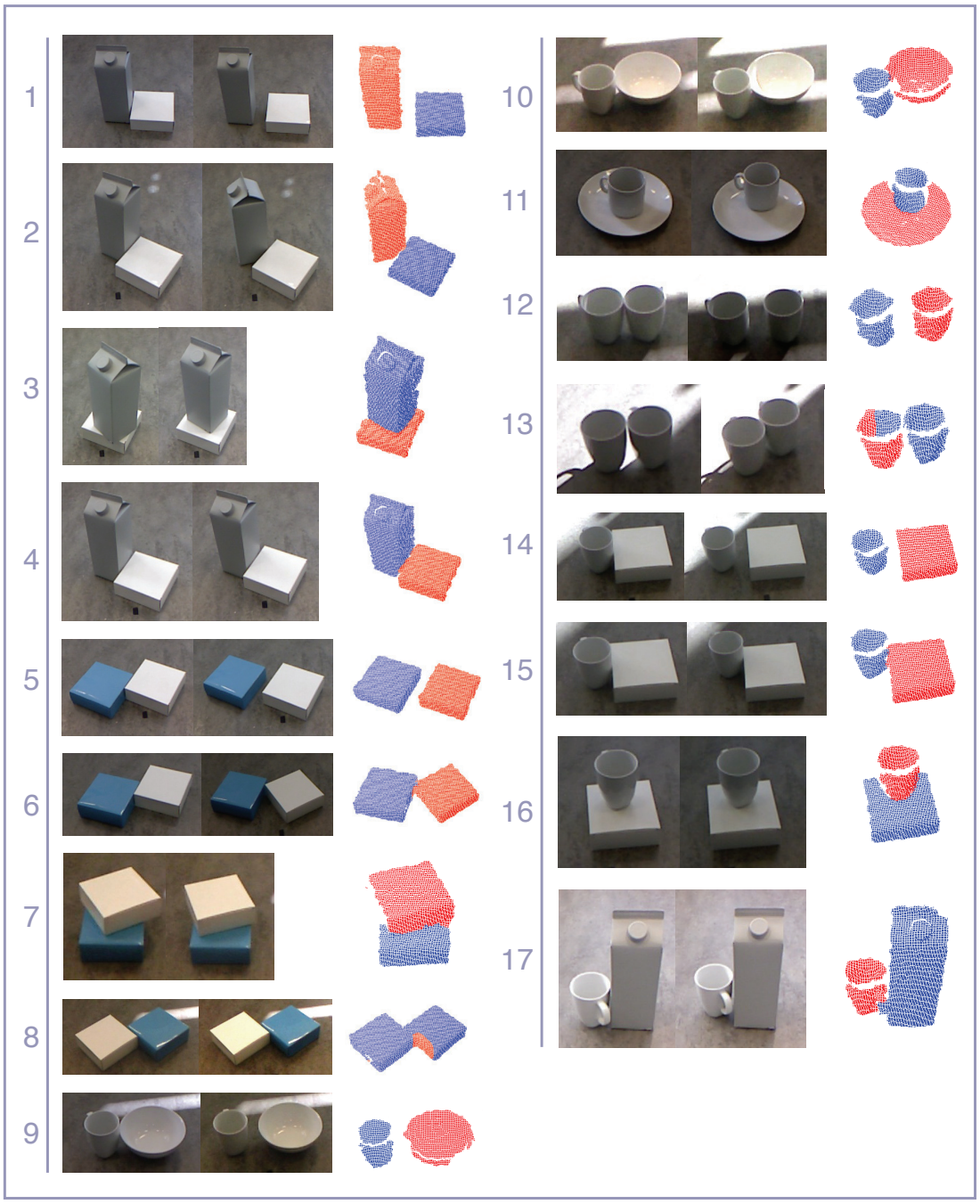


Figure 5.14: *Results of the segmentation for 17 scenes. 1st/4th image column: image before the push for scenes. 2nd/5th column: image after the push for scenes. 3rd/6th column: point cloud after dense model reconstruction for scenes.*

Scene number	1	2	3	4	5	6	7	8	9
Success rate[%]	100	100	100	100	100	33,3	100	0	100
Scene number	10	11	12	13	14	15	16	17	
Success rate[%]	100	66,7	100	33,3	100	100	100-	66,7	

Table 5.1: *Segmentation results for all 17 scenes. For each scene there were 3 experiments conducted.*

	total	diff. size	same size	flat-flat	round-round	round-flat	apart	in contact	on top
Success rate[%]	82,4	93,9	61,1	79,2	80,0	91,7	100	66,7	91,7

Table 5.2: *Segmentation success rates of different scene configurations.*

5.8 Discussion

To exemplify some of the failed cases for the segmentation of textured objects we would like to direct reader's attention to Figure 5.15. One failure was caused by unsuccessful tracking of the features through the image sequence, for instance when the robot's arm occluded initially detected features (e.g. pepper box in the left scene). A similar effect was observed, when an object was rotated due to the robot interaction and features on its vertical surface were occluded by the object itself. In the right scene, a semi-transparent and a reflective object was used. The failure was caused because the features were lost during tracking or they moved entirely inconsistently as the reflection pattern changed.

For the textureless objects we would like to draw the reader's attention to all the scenes with the round objects (Figure 5.14). It can be noted that the Kinect sensor from the used viewpoint (mounted on the head of the human size PR2 robot) always captures mugs as two spatially non-connected parts. In order to robustly merge these two parts using segmentation algorithms operating on point clouds or images of static scenes, model-based segmentation algorithms are required. While that constitutes a feasible solution, the system presented in this thesis can easily deal with such scenes without a model by clustering the two parts of the mug since they move rigidly with respect to each other.

For the scene in right column of Figure 5.5 we can observe that there is only one feature on the left object. All the clustering algorithms trying to explicitly cluster at least one pair of features with the constant relative distance over time would fail in

Figure 5.15: *Failure cases exemplified. In the left-top scene the "black pepper" object became occluded by the robot arm. In the right-top scene the features on the semi-transparent object were tracked unsuccessfully. Bottom row: failed segmentation from the random pushing experiment where the robot pushed objects such that they all moved rigidly with respect to each other*

this case. Using the graph-based clustering method we are able to disconnect the two nodes of the graph and infer that there is a single feature-object association.

Chapter 6

Knowledge-linked Object Recognition

6.1 Introduction

A robot acting as a household assistant must be capable of recognizing the hundreds of objects of daily use that are present in its working environment. It also has to be able to recognize new objects, for example, when emptying a shopping basket to put the purchased items where they belong [Klingbeil et al., 2011a]. For the latter task the robot has to retrieve semantic information, such as the product type, typical storage location, perishability and other characteristics.

In this chapter we present the design and implementation of the Objects of Daily Use Finder (*ODUfinder*) perception system that can deal with some aspects of this challenge. The system consists of two major components, an object modeling (*ODUfinder-m*) and an object recognition (*ODUfinder-r*) component. The system (shown in Figure 6.1) detects and decodes barcodes on objects by implementing an open source ZBar barcode reader library [Brown et al., 2011], retrieves semantic information about objects from a large (over seven million products) product information website Barcoo[1], builds the appearance models of the textured objects and finally detects and recognizes those in typical kitchen scenes. The models for perceiving the objects to be detected and recognized are acquired autonomously using the robot's camera (Kinect in this case) but could also be loaded from the large object catalogs such as Google images. In the system configuration described here, the robot is equipped with an object model library containing approximately 3500 objects from Barcoo. The *ODUfinder-r* achieves an object detection rate of 10 FPS and recognizes objects

[1] www.barcoo.com

reliably with an accuracy rate of over 80%. Object detection and recognition is fast enough so that it does not cause delays in the execution of the robot's tasks.

Using the *ODUfinder-m* (Figure 6.1 top) the robot autonomously builds up the visual appearance and the semantic model of the textured object. To do so it leverages the open source ZBar barcode reader library [Brown et al., 2011] for product barcode identification. The library reliably extracts product data encoded with many barcode formats (symbologies) from camera frames in real time. Having successfully read the barcode, the robot then in real time queries the Barcoo website and extracts information such as the object type, its relation in the taxonomy of object classes, object picture, etc., which we (at the moment) manually align with the taxonomy in the knowledge processing system KnowRob [Tenorth and Beetz, 2009] and thus SOM$^+$maps. Barcoo is with over seven million object classes one of the largest standardized product information store in the world.

The *ODUfinder-r* system employs a state-of-the-art object perception technique Scale Invariant Feature (SIFT) [Lowe, 2004] using a vocabulary tree [Nister and Stewenius, 2006], which we extend in that the *ODUfinder-r* detects candidates for textured object parts by over-segmenting image regions and then combines the evidence of the detected candidate parts to infer the presence of the object. This extension substantially increase the detection rate as well as the detection reliability, in particular in the case of partial obstruction and in certain lighting conditions like specular reflections on object parts. In a nutshell this chapter provides the following main contributions:

- A barcode detection and recognition library ZBar that reliably extracts product data encoded with many barcode formats (Section 6.3.1.3);
- An over-segmentation-based recognition of textured objects (Section 6.4);
- An application of a vocabulary tree matcher to real perception problems (Section 6.4);
- An access to one of the largest objects of daily use information catalogs Barcoo with standardized descriptions.

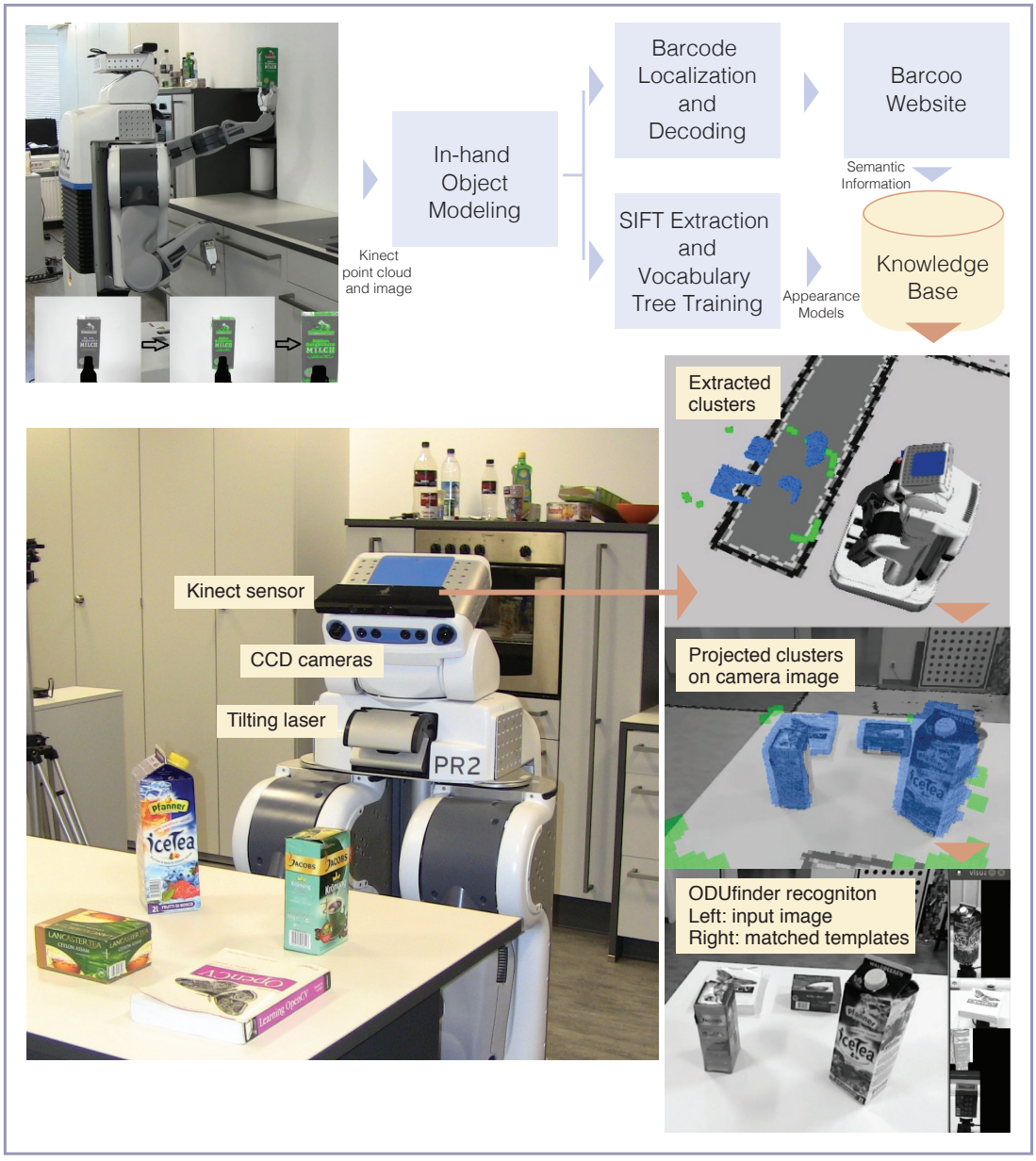


Figure 6.1: *Top row: System diagram for* ODUfinder-m. *PR2 robot builds up an object appearance model, retrieves its semantic information and stores both in the knowledge base. Bottom row: PR2 robot recognizing objects lying on the tabletop using Kinect sensor and* ODUfinder-r. *Right column depicts extraction of clusters from point clouds (top), projection of clusters onto camera image and Region-Of-Interest extraction (middle) and, finally,* ODUfinder-r *recognizing objects (bottom).*

ODUfinder system is out-of-the-box and open-source available in ROS[2] and can be easily deployed on any kind of robot equipped with a 3D sensor and a camera with auto-focus that are calibrated with respect to each other.

In terms of related work, Nakayama et al. [2009b] present the AI Goggles system, which is a wearable system capable of describing generic objects in the environment and of retrieving the memories of these objects by using visual information in real time without any external computation resources. The system is also capable of learning new objects or scenes taught by users. As the core of the system, a high-accuracy and high-speed image annotation and retrieval method supporting online learning are considered. The authors use color higher-order local auto-correlation (Color-HLAC) features and the Canonical Correlation Analysis (CCA) algorithm in order to learn the latent variables.

Arbeiter et al. [2010] implemented a framework for 3D perception and modeling. The proposed algorithm can be used to reconstruct a 3D environment or learn models for object recognition on a mobile robot. Both color and time-of-flight cameras are used, and 2D features are extracted from color images and linked to 3D coordinates. Those coordinates then serve as input for a modified fastSLAM algorithm that is capable of rendering environment maps or object models.

A self-referenced 3D modeler is presented in [Strobl et al., 2009], where the authors demonstrate that an ego-motion algorithm can simultaneously track natural, distinctive features and provide 3-D modeling of the scene. The use of stereo vision, an inertial measurement unit and robust cost functions for pose estimation further increased system's performance.

Incremental learning and recognition of objects is done in an unsupervised manner in [Triebel et al., 2010], but the authors focus mainly on chairs, and it is not clear how well multiple objects could be reliably detected without any prior information. Moreover, scalability is hard to assess since only one view is analyzed at a time.

Much of the recent barcode literature focuses on mobile platforms. In [Adelmann, 2006] a binarized scan line based approach is used to read product barcodes. Similarly for [Wachenfeld et al., 2008], which also relies on assumptions specific to the

[2]http://www.ros.org/wiki/objects_of_daily_use_finder

EAN/UPC symbology. In [Rocholl et al., 2010], EAN/UPC data is extracted from blurry images by guessing digits and comparing a single scan line with a mathematical blur model.

The remainder of this chapter will proceed as follows: in the next section we discuss the system's architecture. Object modeler is explained in Section 6.3, followed by Section 6.4 focusing on the *ODUfinder's* capability to recognize objects. In Section 6.5 we present integration of *ODUfinder* into an ensemble of perception expert methods and in Section 6.6 we present the integration with the knowledge base. In Section 6.7 we discuss the results of experiments and, finally, in the end we conclude and give suggestions for future research.

6.2 Perceptual Pop-Out

The *ODUfinder-m* is depicted in Figure 6.1 top. We assume that the objects stand on horizontal, planar surfaces and the scenes they are part of can be cluttered or the objects are more or less isolated. The robot then detects the horizontal plane and the unknown object candidates as the perceptual pop-outs (Figure 6.2). Next, robot computes the grasp points as presented in [Ciocarlie et al., 2010], grasps the object and articulates it in front of the robot's cameras as described in Section 6.3.1.1. While the object is being rotated the robot learns its visual appearance (Section 6.3.1.1) and concurrently tries to localize and decode its barcode (Section 6.3.1.3). Finally, the object information is extracted from Barcoo and stored into the knowledge base (Section 6.6).

With *ODUfinder-r* (Figure 6.1 bottom) the robot simultaneously takes a 3D scan and captures an image of the scene in front of it. The robot detects object hypotheses as in *ODUfinder-m* above. These object hypotheses are then back-projected into the captured image as regions of interest and searched for objects using the Vocabulary Tree matcher (See Section 6.4).

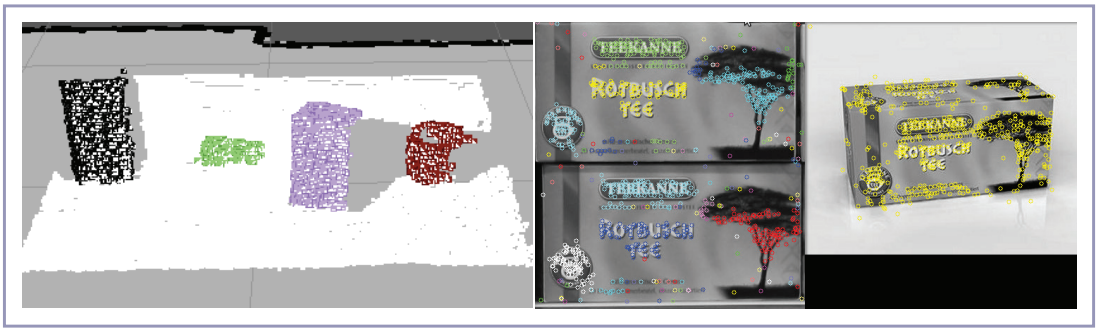


Figure 6.2: *Left: Region of interest extraction using back projection of 3D points, Right: Over-segmentation using a region-growing based approach.*

6.3 Objects of Daily Use Finder

6.3.1 Object Modelling

ODUfinder-m consists of three essential modules (Figure 6.1 top) for i) in-hand object manipulation, ii) barcode localization and decoding and iii) learning of object appearance models. In the following we discuss these modules in details.

6.3.1.1 In-hand Object Modelling

We assume that the robot is positioned in front of the horizontal plane at the approximate table height and has the head-mounted Kinect sensor pointed at the table. The object model acquisition process is depicted in Figure 6.3 and best explained through the following steps:

- extract the horizontal plane and the object clusters [Klank et al., 2009];
- calculate object grasp points on object's cluster [Ciocarlie et al., 2010];
- grasp the object, bring it in the frustum of the camera and set it upright;
- rotate the object around the up-right (z) axis in 30° steps;

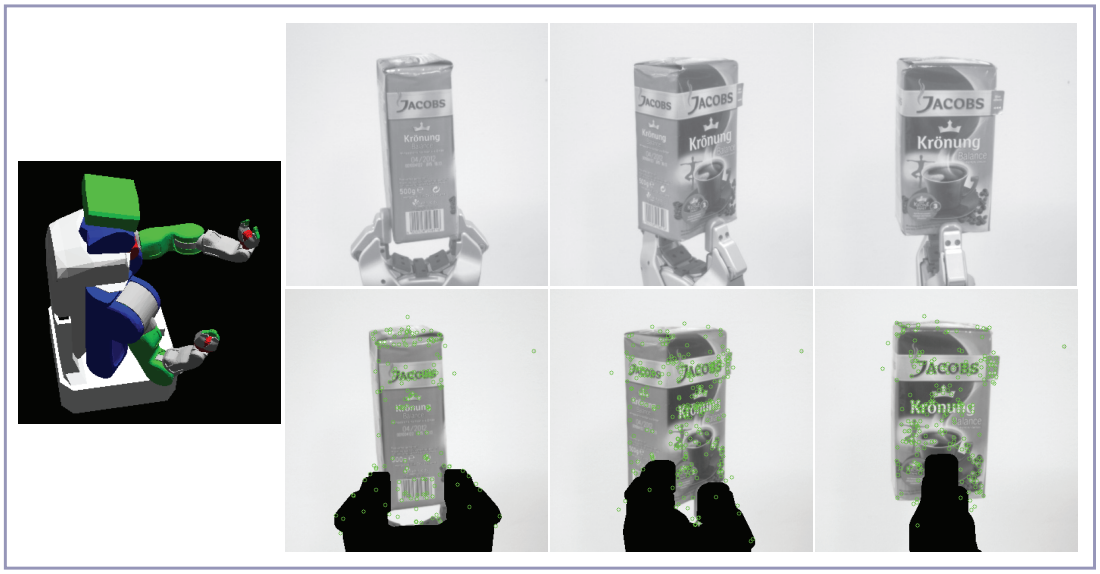


Figure 6.3: *Robot (left-most image) is manipulating an object in front of the camera (top row). Bottom row: Extraction of keypoints and masking of robot's parts.*

- mask out parts of the robot and extract the object template using SIFT keypoints (Figure 6.3). We use depth sensor information to filter out noisy keypoints from the environment as well as the keypoints belonging to the robot itself;
- build documents from the keypoints, quantize them with the existing vocabulary tree and add them to the database (Section 6.3.1.2);
- find the barcode and query the object information from Barcoo (Section 6.3.1.3);
- repeat above four steps until object has been rotated for $2\pi rad$ (note that our PR2 robot is equipped with the continuous revolute wrist joint).

6.3.1.2 Vocabulary Tree-based Recognition of Textured Objects

As already pointed out before we perform object recognition of textured objects by computing the set of SIFT descriptors for all distinctive pixels in through perceptual pop-out computed region of interest and then determine the object model in the library that best explains the set of SIFT descriptors of the region of interest. Each object view contains the set of SIFT descriptors of the distinctive pixels.

Unfortunately, comparing a region of interest with every object view in the object model library is prohibitively expensive. To this end, as proposed by Sivic and Zisserman [2003], we consider object recognition as a document retrieval problem, which enables us to use fast data structures and retrieval algorithms and apply them to object recognition problems for large libraries of object models.

Vocabulary Tree. We employ vocabulary trees that were developed by Nister and Stewenius [2006] for retrieving similar images in very large image libraries. The vocabulary tree of branching factor K and depth L is a tree data structure where the nodes in the tree represent a set of SIFT descriptors. The root node of the vocabulary tree represents the SIFT descriptors of all views of all object models in the library. If a node n in the vocabulary tree represents the set of SIFT descriptors $\mathcal{N}$ then its children nodes represent the partitioning of $\mathcal{N}$ into k subsets represented by the children nodes $cn_1 \ldots cn_k$, where the SIFT descriptors within a children nodes are similar and the ones of different children nodes dissimilar (see Figure 6.4).

Thus, by taking a SIFT descriptor *sd* and classifying it hierarchically through the vocabulary tree using the defined distance measure on the SIFT descriptors we quickly find the set of SIFT descriptors that are most similar in the object model database as the leaf nodes, whose representative SIFT descriptors have the smallest distances to *sd*. We apply vocabulary trees for TF-IDF (Term Frequency Inverse Document Frequency [Robertson, 2004]) indexing, a method used in document retrieval to find documents that best fit a given textual user query. For efficiency, *sd* is not compared to all features in a given node, but to the centroid of its features.

The SIFT descriptors in the vocabulary tree also have a reference to the object model in which they occur. Thus, when *sd* matches a leaf node it votes for the object models that the SIFT descriptors of the identified leaf belong to.

The children nodes $cn_1 \ldots cn_k$ of $\mathcal{N}$ are computed by applying k-means clustering to the SIFT descriptors of node n. Since the TF-IDF algorithm works on words (the equivalent of leaf nodes), we use a vocabulary tree to convert the keypoint descriptors into words, where each word is an integer value corresponding to the number of the leaf node.

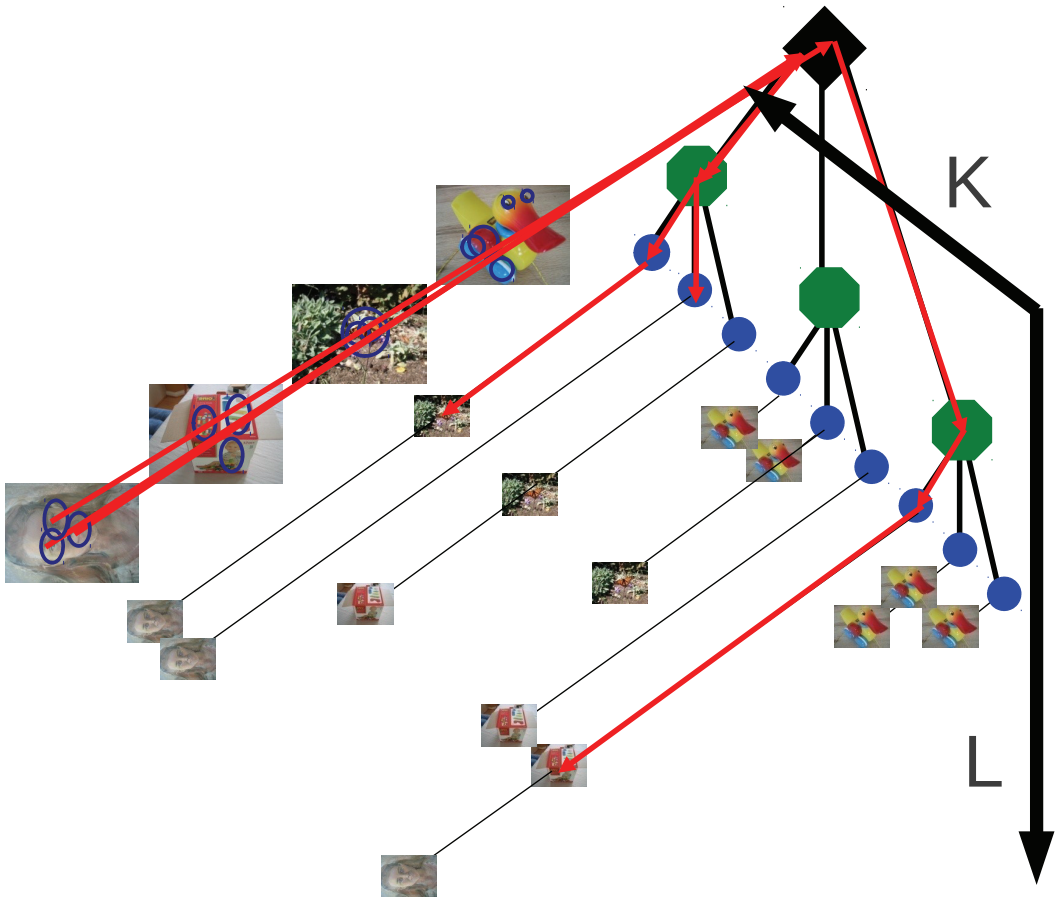


Figure 6.4: *Example of a vocabulary tree and filling with the training data. Courtesy@Nister.*

Building the Database. In our approach we use a similar database as described by Nister and Stewenius [2006]. In order to be able to detect objects the database only stores the quantized SIFT features of the images, but not the images themselves. The following steps describe the building process of the database:

- In order to extract the visual SIFT features from the images we use an open-source implementation [libfastsift] of the standard SIFT algorithm as initially described by [Lowe, 2004]. Each SIFT feature is characterized by a 128 dimensional descriptor vector, 2 image coordinates, a scale and an orientation value. In the current implementation we only use the descriptor vectors for the detection process and the image coordinates for visualization.

- After we have the vocabulary tree, we quantize feature descriptors to single words. For every image, we take all SIFT features, we quantize them with the vocabulary tree and we group the resulting words into one document for every image. In this way each document is composed of a list of all quantized features corresponding to a single image.

- After generating all image documents, we insert them into a specialized database as proposed by Nister and Stewenius [2006]. The database is then trained with the TF-IDF algorithm. After this training the database can be queried with documents generated from input camera images in order to find the best database matches between objects in the image and objects in the database. The database documents, along with specific database information, are stored in a binary format in order to allow for fast loading of the database. Additional information, like image file names, textures and feature coordinates, is also saved for visualization purposes.

6.3.1.3 Barcode Recognition

Product barcodes are recognized by processing each camera frame with the ZBar library (Figure 6.5). The library arranges scan passes over the image, extracts bar/space (element) edges directly from the grayscale image, searches for specific patterns in the measured width ratios and returns data about any decoded barcode instances (symbols).

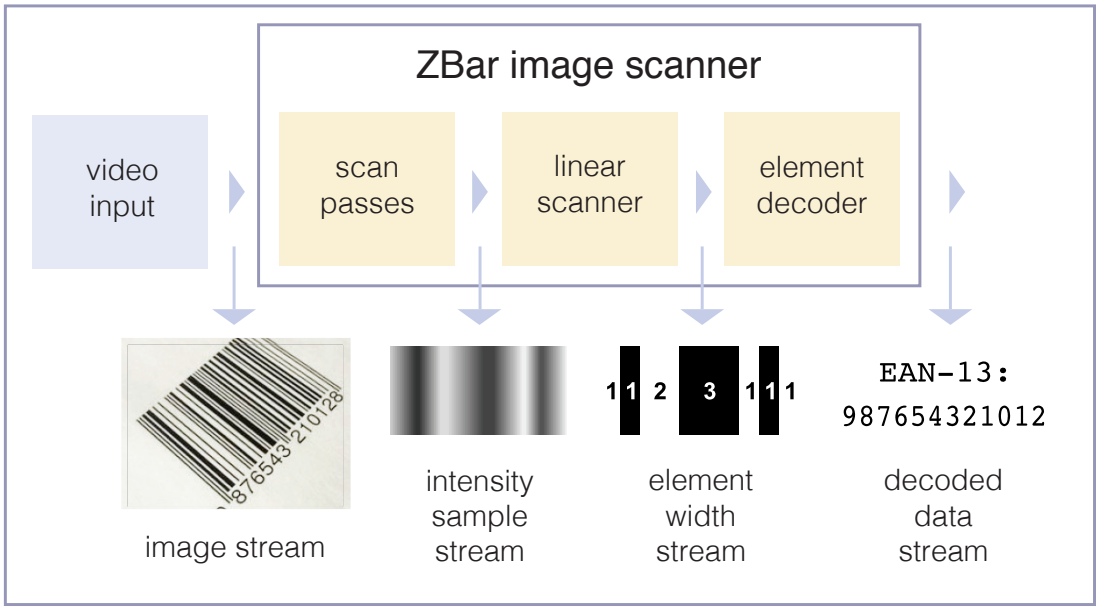


Figure 6.5: *Pipeline used by ZBar to recognize barcodes. Data streams between the processing stages with minimal buffering. Courtesy@Brown.*

All barcode image processing is implemented using a causal, streaming approach; there are no full image processing steps and minimal intermediate buffering is required for each stage.

Image Scanner. The image scanner analyzes incoming video frames and returns any detected barcode information. The barcode search starts by decomposing the two-dimensional image into one-dimensional scan passes, generated by iterating the image pixels using a simple axis-aligned grid (Figure 6.6).

Each scan pass is an independent stream of pixel intensity values, which may be compared to the data generated by a laser or wand scanner. Note that the scan grid makes no assumptions about the location of barcodes in the image; however, for symbols with a large aspect ratio, it does assume the symbol is approximately aligned to the image axes.

The density of passes in the scan grid is configurable, allowing a trade-off between processing time and redundancy, which affects decode latency and sensitivity to symbol orientation, while still allowing full resolution in the scan direction.

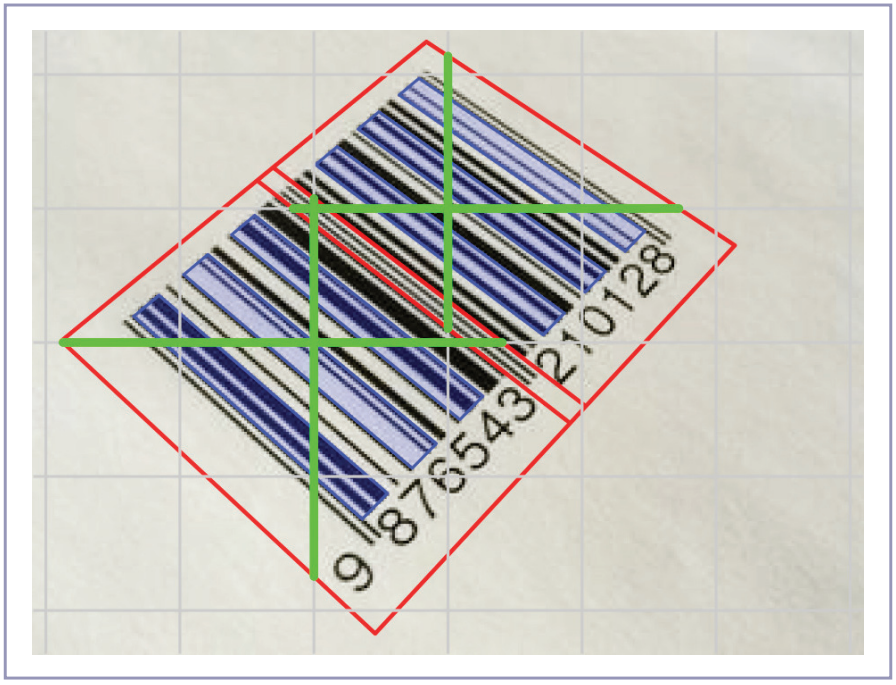


Figure 6.6: *Example scan grid overlaid on an EAN-13 symbol. The two independent halves of the symbol are outlined (red), as well as the individual characters (blue). Each scan pass is streamed to the linear scanner. Successful scan passes are highlighted (green). Note that a typical scan grid uses a much denser stride (1-3 pixels). Courtesy@Brown.*

Intensity samples are streamed to the linear scanner for edge detection and the resulting element width stream is fed to the decoder. Finally, the image scanner collects any decoded data and handles temporal redundancy by applying hysteresis and duplicate suppression before returning the results.

Linear Scanner. The linear scanner looks for edges in the intensity stream and generates a running sequence of alternating bar (dark segment) and space (light segment) element widths. Edge detection (Figure 6.7) begins with a simple IIR low-pass filter to remove some noise from the signal:

$$\widehat{y}_i = \alpha y_i + (1-\alpha)\widehat{y}_{i-1} \tag{6.1}$$

where y_i is the new intensity sample and α is the constant smoothing factor for the filter. This filtered intensity stream feeds a standard one-dimensional differential edge detector (similar to [Marr and Hildreth, 1980]). Derivatives are calculated as:

$$\widehat{y}'_{i-1/2} = \widehat{y}_i - \widehat{y}_{i-1} \tag{6.2}$$

$$\widehat{y}''_{i-1} = \widehat{y}_i - 2\widehat{y}_{i-1} + \widehat{y}_{i-2} \tag{6.3}$$

corresponding to the kernels $y' = [-1\ 1] * y$ and $y'' = [1\ -2\ 1] * y$.

Primary noise filtering is achieved by application aware thresholding of $\widehat{y}'$; barcode edges occur together, with similar strengths and spacing proportional to their size. To leverage this, the threshold level, T_i, is selected dynamically, based on the last edge rate, and decays linearly to a minimum level, $T_{\min}$, at a rate based on the last reported element width:

$$T_i = \max\left(T_{\min}, \gamma\widehat{y}'_j\left(1 - \tau\frac{x_i - x_j}{x_j - x_k}\right)\right) \tag{6.4}$$

$$|\widehat{y}'_i| \geq T_i \tag{6.5}$$

where x_i is the location of the current sample, x_j and x_k are the locations of the previous two edges and $\gamma, \tau \in [0,1]$ are constants that determine the relative edge sensitivity and rate that the threshold returns to minimum, respectively.

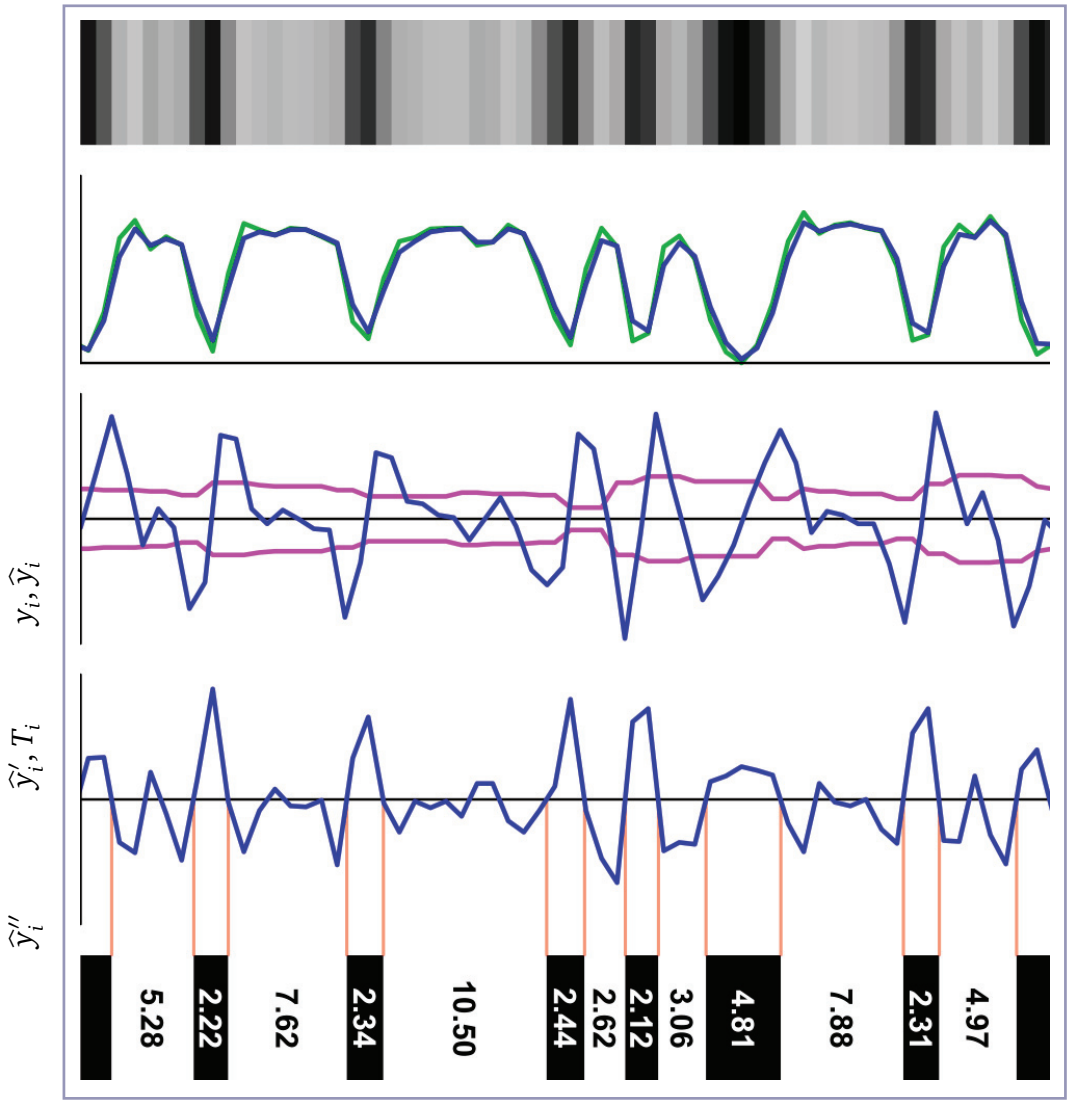


Figure 6.7: *Edge detection performed by the linear scanner: (top) input signal and low-pass filter, (top-middle) first derivative and threshold, (bottom-middle) filtered zero crossings of the second derivative, (bottom) output widths. Courtesy@Brown.*

After thresholding, the selected zero-crossings of $\widehat{y}''$ are interpolated to locate edges with sub-pixel precision and non-maximal suppression selects the most salient transition for each edge.

One of the goals of the ZBar library is to minimize the number of parameters presented to users. Therefore, the scanner constants, α, γ, τ and T_{min} were selected during scanner design and are hardcoded into the library implementation. The values apply across a broad range of use cases and were determined empirically, using a relatively small set of images (around 100) from various sources.

Decoder. The element decoder examines the stream of element widths from the linear scanner, looking for specific ratios defined by the relevant barcode specifications. When a valid pattern is detected, the decoded data and associated symbology identification metadata are returned to the image scanner.

Many linear product identification symbologies are supported by the library, including: the GS1 EAN/UPC family [GS1], GS1 DataBar [DataBar] and Code 128 [Code128], among others. Each supported symbology is implemented as a separate finite state machine (FSM). These FSMs operate in parallel on the same element width stream, enabling auto-discrimination between symbologies.

Decode for a given symbology typically proceeds by looking for one or more "finder patterns" – ratio combinations unique to the symbology that identify a specific location within a symbol – usually begin/end delimiters, sometimes a central finder. Subsequent characters are then expected at specific intervals. The decode is complete when another delimiter is encountered. Any unexpected or illegal patterns reset the FSM back to the initial search state.

When comparing element width ratios, the decoder avoids direct comparison between bars and spaces, which is overly sensitive to variations in exposure and printing process. Instead, the decoder adopts a more robust approach suggested by most of the barcode literature, which prefers to compare bar and space *pairs* or bars to bars and spaces to spaces, as these combinations are invariant to consistent erosion and dilation effects caused by exposure variation.

Decoder reliability is assured variously (depending on the specific symbology) through a combination of character self-checks, symbol completeness, data checksums and the redundancy of collecting multiple scan passes.

In Section 6.6 we show how to query, parse and store the information from Barcoo into the SOM$^+$map.

6.4 Object Recognition

In this section we show how we have specialized vocabulary tree for the purpose of object recognition in the context of robot perception. Our principal aim was to improve the capability of the proposed method for identifying objects in real scenes, which required taking different lighting conditions, obstruction and clutter, and the uncertainty and noise associated with physical sensors acting in the real world, into consideration.

In order to find an object in the extracted ROI we have to generate a database document in the same way as described in Section 6.3.1.2. We first extract the SIFT features from the received image and we quantize the descriptor vectors to words with the vocabulary tree. A single document is formed from all words of the input image and we can query the database with it. The database returns the best N matches with their respective scores (between 0 and 2, where 0 is best and 2 is worst).

This approach performs well as long as the objects are isolated and the approach from previous subsection extracts one object per ROI. If two or more objects are in the ROI, and especially if more than one of them is also loaded in the database, the performance decreases. This happens because the database retrieval mechanism tries to find an image containing all of the objects together and, although the objects can still be detected, their scores are low and very similar. This makes it very difficult to tell which match truly corresponds to the object in the image.

In order to improve recognition performance in such cases we present a clustering of features of the input image in 2D space (the position of the feature in the image). We determine the visually distinctive pixels using SIFT features and apply region growing algorithm to determine the clusters. Region growing starts from a point that

does not belong to any clusters and incrementally adds points that are in a predefined radius r around the original point. The process is repeated for all newly added points. This results in clusters that represent the strongest texture "islands" in the image (Figure 6.2 right).

For our application, the quality of the segmentation results heavily depends on the appropriate setting of the radius parameter r. In order to improve performance, we adaptively chose the radius length in relation to the level of texturedness of the camera image using a scaled and shifted logistic sigmoid function:

$$r^2(x) = (r_{max}^2 - r_{min}^2)(K(1 - logsig(x - A))) + r_{min}^2 \tag{6.6}$$

where logsig is defined as:

$$logsig(x) = \frac{1}{1 + e^{-x}}. \tag{6.7}$$

Argument x is the number of keypoints in the image. The parameters r_{min} and r_{max} denote the maximum and the minimum values of the radius. The parameter A denotes the value of x, where the value of the function is the average of the minimum and maximum value of the radius. The constant K denotes the speed at which the function approaches its minimum and maximum values. These 4 parameters are determined empirically and are valid for images of roughly similar sizes. In the experiments below we use the following values: $A = 800$, $K = 0.02$, $r_{min} = 200$, $r_{max} = 600$.

In this way we can find rich-textured sub-regions in the object candidate image. It is difficult to make the clustering algorithms find the exact regions of the objects, but our experiments show, that this is indeed not necessary. If we adjust the clustering algorithm to over-segment, we get several clusters per object. These clusters correspond to the strongest textures of the objects and are, in most cases, enough to identify the whole object (see Figure 6.8).

The next step is to generate a document for every cluster size greater than the predefined size $S_{cluster}$ and query the database with those documents. Typical values for the $S_{cluster}$ are between 20 and 30, because smaller clusters are unlikely to produce meaningful results. Thus, every cluster has its own ranking of the most probable matches and we need to merge the results. In order to combine the results from every cluster

Figure 6.8: *Detection of objects by partial textures. Left part shows that only a "Jacobs" sign is sufficient, while the right part implies the same for a "Kronung" sign.*

into one final list of matches, we sum the scores ($clusters_{scores}$) which result from matching of every cluster against every image in the database. In this way, if several clusters vote with a high score for a specific image in the database, we understand that it is very likely that we have found the right object in the image. Note that if we had two objects in one input image, which also have respective entries in the database, then we will get more than two clusters from the input image (thanks to the over-segmentation) and the database retrieval mechanism will not search for the documents containing both objects, but rather only for parts of the objects, which will result in far more distinctive scores.

The final consensus is that, as our segmentation method tends to over-segment, the *ODUfinder-r* considers the image regions that could spatially lie on the same objects as multiple evidence for the respective objects and combines the evidences provided by the individual regions. Obviously the visual region-based object model appearance is particularly appropriate to handle partly obstructed objects and those which might have parts that cause reflections.

6.5 Integration of ODUfinder in the Perception Server for Generic Object Recognition

In order to be able to recognize all possible objects that the robot may encounter in a household, having only one or few perceptual mechanisms will not suffice. Many algorithms have thus been developed to solve these problems for different subsets of objects, with varying accuracy and reliability, with different requirements for computational resources, and under different context conditions. Some of them require prior object models while others can do without, some infer only general categories, others exact instances without the knowledge of the broader categories these objects fall into. The approaches also differ in the type of sensors used, in speed, in that not all of them report 6D poses, in the number of objects they can deal with at once, etc. The realization of robot perception systems that can perceive the range of objects to be manipulated in a typical human environment with the accuracy and reliability needed for grasping them successfully in real everyday settings, however, still poses a very hard research problem. To overcome this problem we helped creating a perception server system that can deal with a truly large range of objects [Marton et al., 2011]. The perception server is depicted in Figure 6.9 and *ODUfinder* represents its classification library. In the following we briefly summarize its architecture.

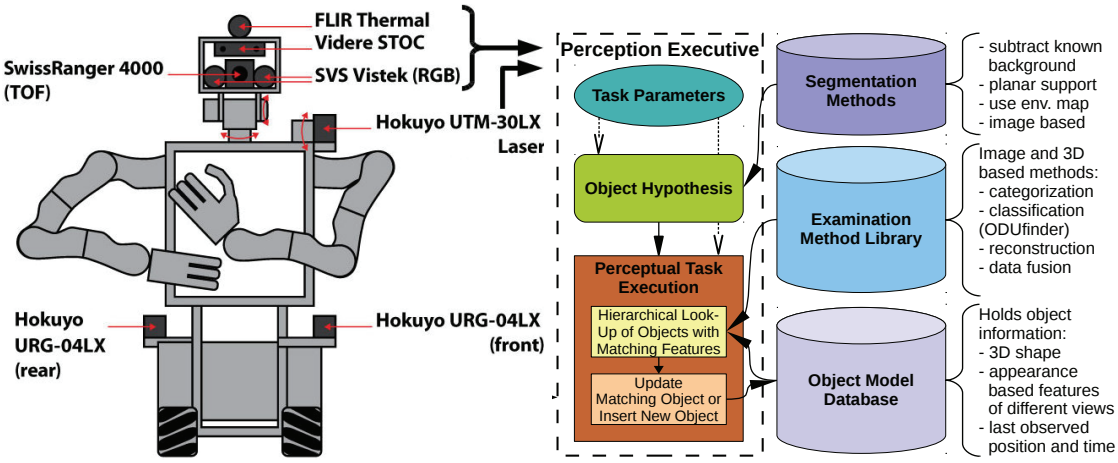


Figure 6.9: *Perception server architecture: from sensor data to objects.*

The images and 3D information coming from the robot's sensors are processed by the *Perception Executive* and the gathered data is interpreted according to the task at hand (searching for a specific object or identifying all objects). First, to limit the

search-space for object locations, a set of possible locations is extracted and the corresponding sensor readings (3D clusters, 2D Regions-Of-Interest) are considered to represent object candidates. These object hypotheses are then processed as needed in order to associate the percepts to the correct object in the *Object Model Database*.

When an object is being sought for, the system selects a set of features, whose values uniquely describe the object amongst all the objects in the database. The same features' values are then computed using the examination modules for each object hypothesis, and the first one that presents matching ones is selected as the target object for e.g. grasping. In the case that no geometric model is associated with the database object in question, we compute it on demand and feed it to the grasp planner.

If computational resources allow all object hypotheses can be checked against all objects in the database – and new objects, or new positions or views of known objects, can be detected. In this case, the features for each object hypothesis are computed one by one, according to a hierarchy, and the possible object identities are filtered in each step. The selection of the feature to be used in each step is hand encoded as of now, but an expansion of the single-object case is envisioned to be extended for finding the most discriminative features in each step.

This process is repeated until either an object is found whose stored features match all observed features, or until there are no matching objects left in the database, signaling that a novel object was observed. In the ambiguous case, when all features were computed and there are still multiple matching objects left from the database, the system takes no action and leaves the object hypothesis unclassified.

Since the computation of the features for an object hypothesis is prohibitively expensive, the aim of the procedure is to also minimize the number of objects that have to be compared against as drastically as possible in each step, and allow a large number of objects to be handled efficiently. Given enough descriptive features, this method can scale well in the context of objects of daily use in human living environments. We consider this approach to be a move away from bottom-up, rigid pipelines, towards a more flexible setup. This enables the robot to specialize to the current situation, producing shorter processing times as not all the methods are needed all the time.

In order to be able to learn more and more about an object, multiple detections with different sensors and from different points of view are needed. To check if two per-

cepts belong to the same object or not, we use a simplified version of the probabilistic framework for identity resolution [Blodow et al., 2010], which is based on positions of objects.

6.6 Integration with the KnowRob and SOM$^+$ Map

In order to enable the robot to reason about the types and properties of the objects it has detected, the output of the recognition system needs to be linked to semantic information about the objects, and to other descriptions of these items that may exist in the system, for example specifications of action parameters. We use the KnowRob ontology [Tenorth and Beetz, 2009] as an *interlingua* to integrate these different sources of knowledge. KnowRob provides formal specifications of object classes as well as their properties and relations. As long as different parts of the robot, e.g. the action executive and the object recognition module, refer to these classes for describing objects, they can make use of information in the other modules.

The Barcoo website already provides a taxonomy of more than seven million object classes that describe for instance *Cheese* as a sub-class of *Dairy* and *Food*. In addition, individual object detections are annotated with a number of properties such as the object picture, price or nutrition facts of the object. By (currently manually) aligning this taxonomy with the KnowRob ontology, the provided information becomes available to the robot for abstract reasoning about the objects it has detected.

Figure 6.10 illustrates the inference steps that can be performed via *tell-ask*-interface as presented in Chapter 4.2. The reasoning process combines encyclopedic knowledge about the refrigerator (upper left part), common-sense knowledge that a refrigerator is the storage place for perishable goods, spatial knowledge describing an instance of a refrigerator at a certain location in the environment, and knowledge about the pancake mix object that was automatically generated from Barcoo's website.

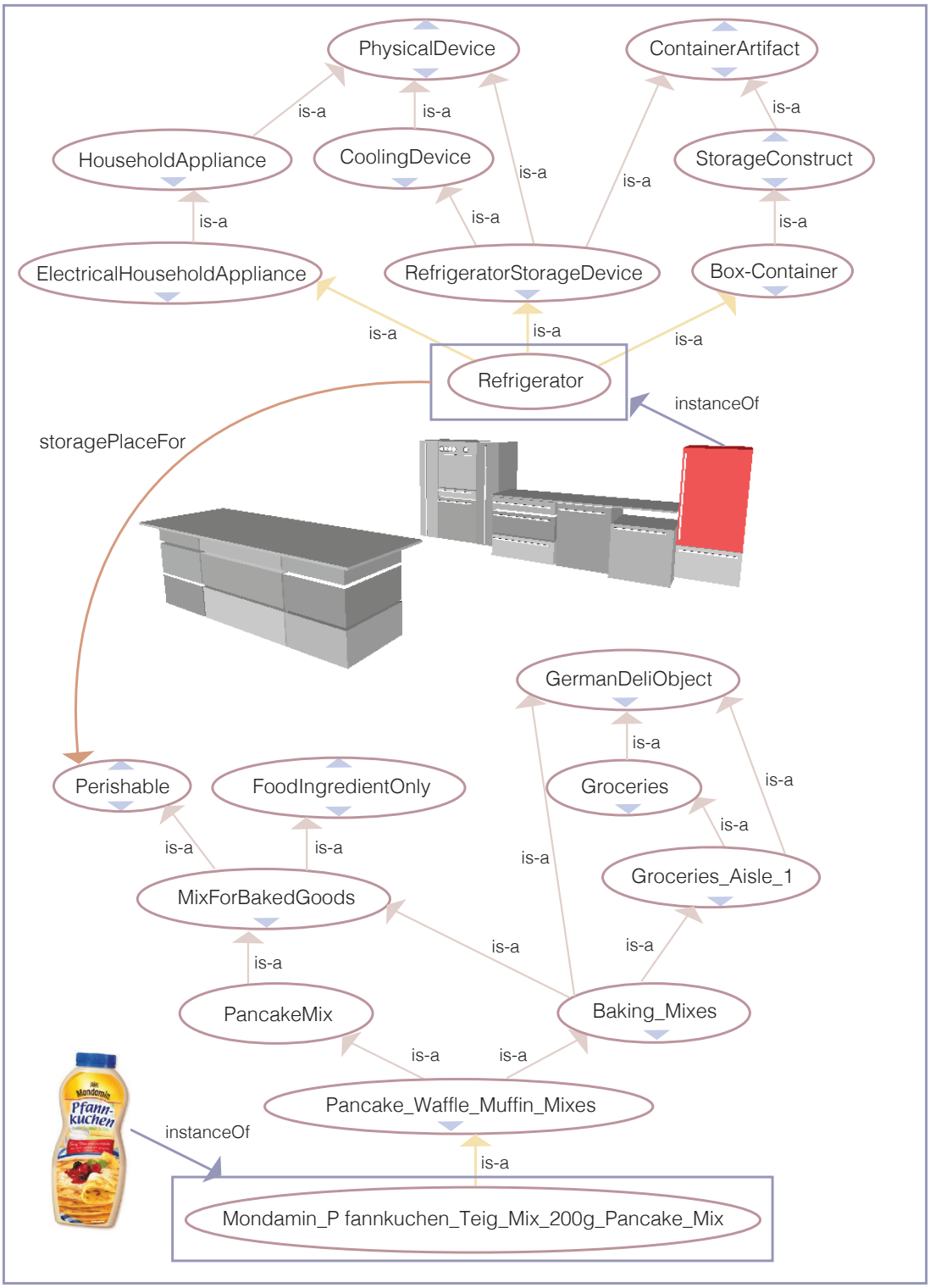


Figure 6.10: *Example of object taxonomy in the KnowRob knowledge base (and thus SOM$^+$ map). Courtesy@Tenorth.*

6.7 Results

We evaluated *ODUfinder* with respect to its runtimes, success rate of barcode and object instance recognition and, finally, with respect to being able to learn and incorporate new object models on the fly.

6.7.1 Barcode Recognition

We evaluated the ZBar library by running the test on barcodes of 30 objects of daily use. During testing we positioned the camera $5cm$ away from the barcode center and then rotated the object around camera's optical axis into 3 possible configurations: $0°, 45°$ and $90°$. We ran three batches of tests under varying lighting conditions: in the morning, in the afternoon and under the artificial light. The averaged results are presented in Figure 6.11, which clearly shows the effect of using an axis aligned scan grid: all barcodes were detected when the symbol was oriented with the scan grid, but the detection rate decreased as the symbol was rotated, with the worst case at 45°, where only a few of the scan passes were able to complete a path through the symbol, compromising the redundancy that ZBar relies on for robust recognition.

6.7.2 Database Training

To provide an insight into the performance of *ODUfinder-m* we have trained a vocabulary tree and built a database with 3500 textured objects from Barcoo. K and L parameters for the structure of the vocabulary tree were 6 and 6 and the Table 6.1 provides the rest of the profiling. Please note the cluster query time under $100ms$ which ensures the real time operation of the system.

Source	**nr. images**	**nr. features**	**training time**	**cluster query time**
Barcoo	3500	2500000	1h	90ms

Table 6.1: *Profile data for the generated database of 3500 objects.*

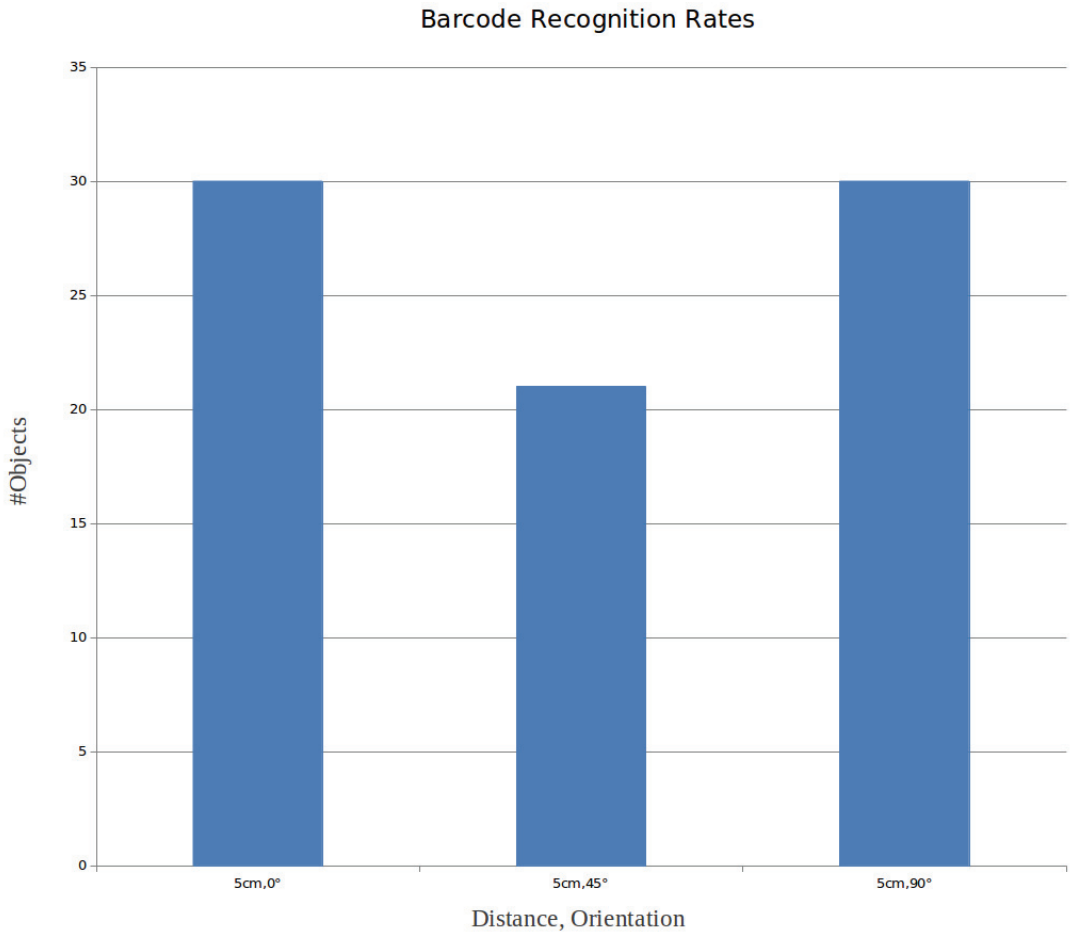


Figure 6.11: *Barcode recognition evaluated on 30 objects.*

Figure 6.12: *Test objects.*

6.7.3 Recognition of Objects based on Known Views

In order to evaluate our approach as a whole, we performed the detection and recognition test in the assistive kitchen laboratory (see left column of Figure 6.13). The test was carried out on a total number of 13 objects located at 4 different scenes (denoted with Scene 1 to Scene 4). The robot was programmed to navigate to each of the scenes and capture point cloud and image from several different views by traversing along the free paths around the scenes. The basic *Planar Support* approach could not have been applied for the scenes 2 and 4 as the supporting planes are too high, thus impossible to scan with either of our robots.

The vocabulary tree and corresponding database with descriptors were trained and built from images from the SemanticDB database and 10 more images of products from the Barcoo web site[3]. The parameters K and L were both set to 5, resulting in a 1 minute training time of the datasbase for the 65000 features extracted from 170 images. In this configuration the querying for 1 object cluster took 50 ms. Setting the score value of the database retrieval mechanism to the experimentally determined

[3]Note that for the Barcoo object we initially only obtain one image per object

Scene	#Views/#Known	#Failures	#Unknown	Success
Scene 1	52/42	10	10	80.8%/100%
Scene 2	11/11	5	0	54.5%/54.5%
Scene 3	24/24	2	0	91.6%/91.6%
Scene 4	12/12	0	0	100%/100%
Total	99/89	7	10	82.8%/92.1%

Table 6.2: *Detection of objects and identification of unknown views using SIFT with vocabulary trees.*

value of 1.0 enables us to classify all measurements that exceed this value as unknown.

As per Table 6.2 scene 1 contained 1 object (green milk box) for which vacabulary tree was trained on the Barcoo image of the object. This known face of the object was initially not in the robot's field of view which resulted in the recognition success rate of 80.8%. For the scene 2 we attribute all failed cases due to the *ODUfinder* not being able to extract the regions of interest. In the scene 3 the robot failed to separately cluster the book and the can which in turn resulted in the wrong regions of interest and finally in the failures in the recognition. The right most column of Table 6.2 shows success rates with and without unknown views.

6.7.4 Improved Detection through Incremental Learning

In the case that an object hypothesis is detected in the same position in the map's coordinate frame in subsequent scans we assume that it is the same object as the one identified previously. For the localization we use an AMCL-based framework combining robot's odometry and laser readings in a priori built 2D map [Pfaff et al., 2006]. Localization's absolute error margin lies at 0.02 m on average, thus we consider object hypotheses to represent the same object if they are not further away than 0.05 m. If the subsequent call of the recognition function of *ODUfinder* returns no matching views for the given object hypothesis, we store the current observation as a new view.

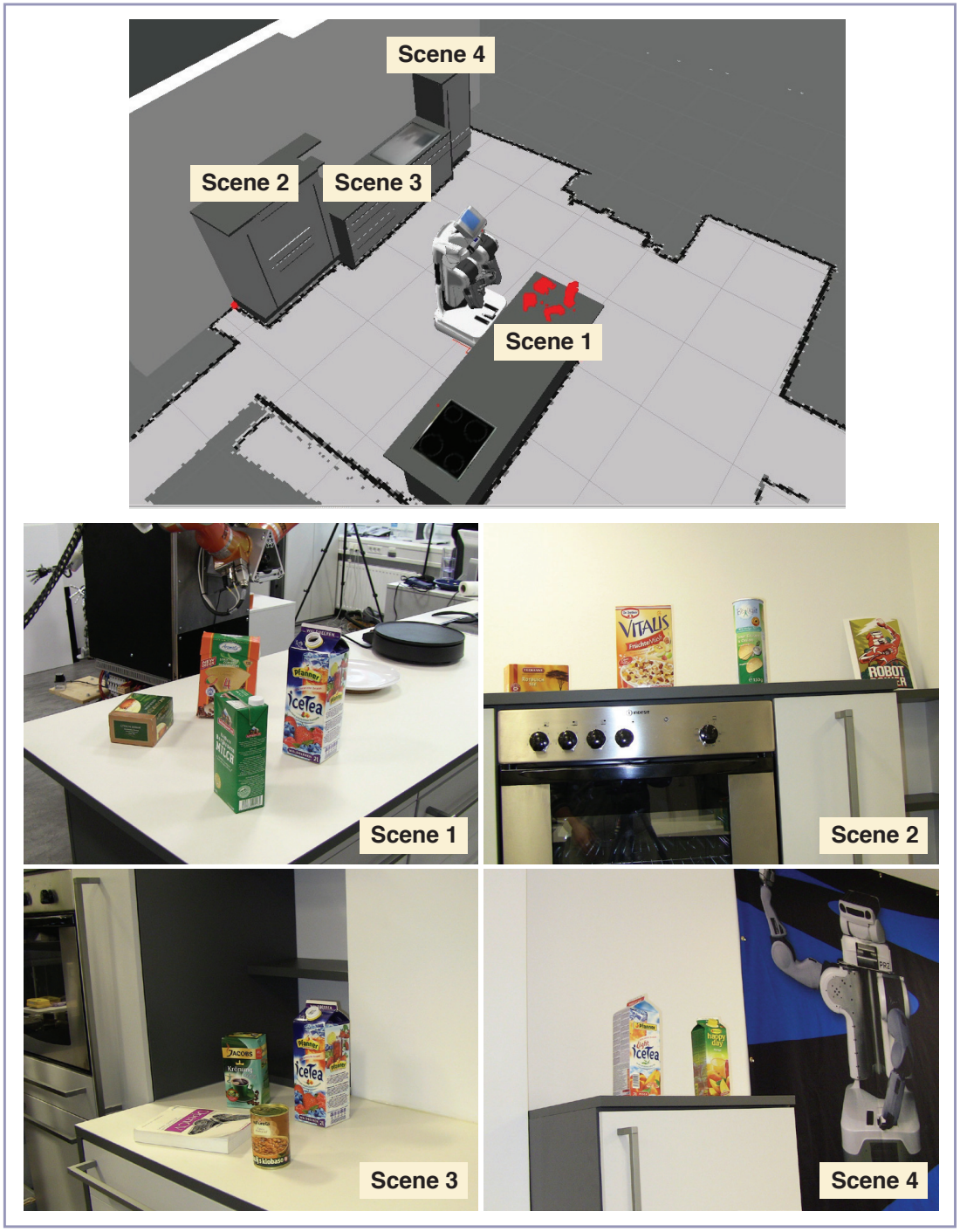


Figure 6.13: *We performed the final evaluation test on a total number of 13 objects located at 4 different scenes in our kitchen lab (denoted with Scene 1 . . . Scene 4). The robot was programmed to navigate to each of the scenes and capture point cloud and image from several different views by traversing along the free paths around the scenes.*

To demonstrate the capability of our system to acquire new object models on the fly we set up the Scene 1 with one unknown object (green milk box) which in fact generated all 10 un-classified views reported in the first row of Table 6.2. Knowing that these do not match anything in the database, we can introduce them as new object models. The assumption we are making here is that the scene remains static, thus the cluster cloud and the defined region of interest at the given 3D position in the world coordinate frame contain the same object. The vocabulary tree is for the moment re-trained every time the new uknown object is detected.

Scene 1	**#Views/#Known**	**#Failures**	**#Unknown**	**Success**
Before	52/42	10	10	**80.8%**/100%
After	52/52	2	0	**96.2%**/96.2%

Table 6.3: *Improved detection for Scene 1 from Figure 6.13 before and after the vocabulary tree was re-trained and database rebuilt with the templates for green milk box. All in all, more views got the correct label.*

After this we performed another test run on the Scene 1 with the re-trained tree and the updated database of SIFT descriptors and were able to reduce the number of not detected objects down to 2 as shown in Table 6.3.

Since most large databases (e.g. Barcoo website) offer only single pictures of objects, incremental learning is an important feature for a perception system that needs to develop over time.

6.8 Discussion

This chapter has presented a perception system for autonomous service robots acting in human living environments, coined the *ODUfinder*. The perception system enables robots to detect and recognize textured objects of daily use, it ensures real-time and robust operation and is modular with respect to the integration of new components (e.g. detection of texture-less or translucent objects). The system is available as open-source and widely used by several research, and in case of ZBar, also industrial communities. We have also shown that fusing together semantic information available through SOM$^+$maps and those extracted from commonly available resources such as

product information websites or web shops gives the robot the power to perform its actions in sensible and reasonable way.

Several research issues remain to be solved. In case of highly mounted shelves or cluttered planar surfaces, our system is unable to generate regions of interest which in turn decreases the success of SIFT matching. For this case we plan to incorporate our interactive segmentation work presented in the previous chapter to validate regions of interest. For dealing with textureless and transparent objects we plan to incorporate object recognizers by Hinterstoisser et al. [2012] and [Lysenkov et al., 2012] respectively. Finally, to effectively fuse various object detection and recognition results we already started exploring UIMA [Ferrucci and Lally, 2004], which is an architecture used in the famous Jeopardy quiz by Watson computer. UIMA architecture makes it possible to analyze large volumes of unstructured information in order to discover knowledge and models that are most relevant for the robot's tasks at hand.

Chapter 7

Knowledge-enabled Scene Understanding

7.1 Introduction

Autonomous robots performing everyday manipulation tasks have to make many decisions that require the combination of perception and knowledge processing. As an illustrative example for knowledge-enabled perception, consider a robot that is to set the table together with a human as outlined in our master plan in Section 1. In order to implicitly coordinate its course of action with the human, the robot has to fetch missing items. Based on what the robot sees on the table and the time of the day, the robot is to probabilistically infer what meal the table is set for, what is likely to be eaten, and, based on this, which utensils are likely to be required.

In this chapter we effectively combine results of Chapter 4, Chapter 5 and Chapter 6. The original version of the system presented herein has been presented in [Pangercic et al., 2010].

We propose the logic programming system K-CoPMAN (Knowledge-enabled Cognitive Perception for Manipulation) that can test and satisfy knowledge preconditions for everyday manipulation. K-CoPMAN fulfills three main functions:

1. Providing the robot with abstract symbolic knowledge about perceived scenes. K-CoPMAN acquires and stores perceptual data during robot operation, associates data structures with symbolic names that can be used for perceptually grounded knowledge processing. These perceptions extend static perceptual data like environment maps introduced in Section 4 and objects introduced in Section 6. There are two main perceptual mechanisms: task-directed and passive perception. *Task-directed perception* provides information necessary for accomplishing manipulation tasks – in-

formation about the object to be acted on and the scene context. The *passive perception* is to make the robot environment-aware by also memorizing objects that are not task-relevant at the time of perception. Object information is stored in K-CoPMan at different levels of detail, ranging from raw, sub-symbolic data to symbolic descriptions.

2. Using abstract symbolic knowledge for accomplishing perception tasks. K-CoPMan enables the robot to employ knowledge processing functionalities to simplify perceptual tasks by using (symbolic) models of context, situations, and goal-directed behavior. Using knowledge processing mechanisms and the belief state (memory) of the robot, the robot can for instance point the camera at places where it believes objects to be or exploit the fact that objects inside a cupboard are invisible unless the door is open.

3. Answering types of queries that require the combination of knowledge processing and perception. For instance, K-CoPMan enables the robot to infer the items that are missing on a table set for a particular meal, the items that have to be put away in order to clean a table, or the items that have to be put into the fridge. Inferring these information requires using a combination of perception and knowledge processing mechanisms.

Technically, K-CoPMan is realized as an interface layer to open-source SWI Prolog [Wielemaker et al., 2012]. Prolog combines fast inference and computation with declarative, logics-based semantics. Lightweight Prolog inferences can even run in feedback loops up to 10 Hz to make the robot action-aware. Prolog's foreign language interface thereby facilitates the integration of perception routines written in other programming languages like C/C++.

The remainder of this chapter starts with an overview of the software architecture (Section 7.2). Then, the perception server and the library of perception routines are explained (Section 7.3). Section 7.4 describes the integration of the perceptual mechanisms with the knowledge processing system KnowRob [Tenorth and Beetz, 2009]. We conclude with a demonstration scenario, discuss and evaluate one example query.

7.2 K-CoPMan System Overview

K-CoPMan is an extension of KnowRob and extends KnowRob in two important ways. First, it adds a set of predicates that abstract away from the robot's perceptual mechanisms and transforms the perceptual tasks and their results into a logical representation suitable for knowledge processing and decision-making. Second, K-CoPMan provides a continual update mechanism for the part of the knowledge base that represents the dynamic world state. This mechanism is to make the robot *environment-aware*, i.e. to always have a rough estimate of the current state of the world. For example, in our application, objects on tables and kitchen counters are declared as a relevant dynamic aspect of the world that should be monitored continually. K-CoPMan keeps track of the positions of objects on different tables and asserts these percepts as logical facts.

A robot programmer can use KnowRob to define concepts needed for robot control in terms of first-order logical statements. For example, to write plans for joint human-robot table setting tasks, the programmer might want to define the concept of items that are missing on a table in the following way: Missing items on a table where people intend to have a meal *m* are those items that are predicted to be needed for this meal, but cannot be perceived to be already on the table. Having this definition, the programmer can write a plan fragment such as: *keep putting a missing item on the table until no further items are believed to be missing*. In this code fragment, the missing item is a knowledge precondition of the plan step that has to be achieved by computing which items in the environment satisfy the above concept definition.

In this setting, K-CoPMan's task is to test the perception-related parts of the concept definition. Thus, K-CoPMan translates the conditions to be checked into parametrized perception routines and interprets their results in order to check the conditions. It also controls and supervises the perception processes that are spawned from the information requests, and it stores and manages the results returned by the perception processes.

In order to perform competent perception, it is often helpful to make use of other knowledge stored in KnowRob. In this example, checking the condition requires the robot to identify the right table, which is accomplished using the semantic environment map stored in KnowRob as presented in Chapter 4. It allows, for example, to

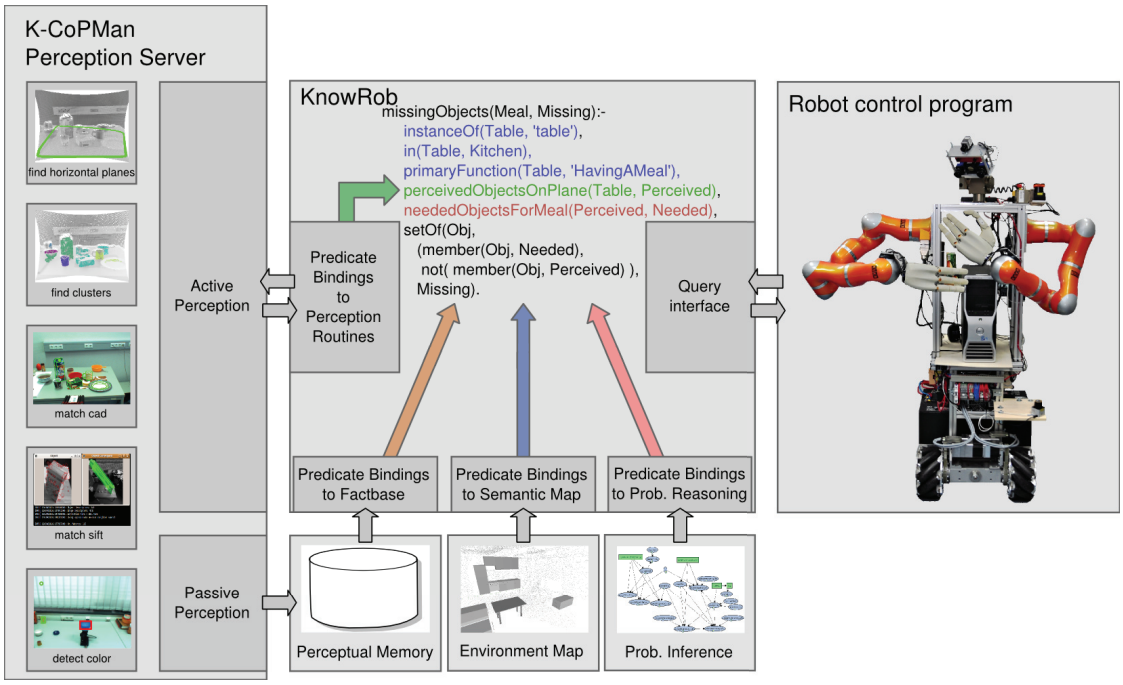


Figure 7.1: K-CoPMan's *building blocks.* ***Left)*** *Perception server used in* K-CoPMan *with state-of-the-art perception routines.* ***Middle)*** KnowRob *with predicates for evoking of perception routines and extension plugins for first-order probabilistic reasoning and knowledge on static objects.* ***Right)*** *TUM-Rosie with logical control program. Courtesy@Tenorth.*

query for objects of type “Table”, especially for those that are used for having meals. Similarly, reasoning with perceived information requires the system to explicitly deal with the uncertainty that results from sensors being unreliable, inaccurate, and only providing incomplete information about the world. This functionality is provided by a predicate library that realizes probabilistic first-order reasoning.

7.2.1 K-CoPMan Components

Figure 7.1 shows the embedding of K-CoPMan into the overall robot control system and the software components of K-CoPMan within this system. The core of K-CoPMan is the utilization of the perception server (see Section 7.3). The perception server calls the respective perception routines, monitors and manages the perception processes they execute, and stores to and updates the K-CoPMan data store according to the perception tasks and their results. The second component is the implementation of the K-CoPMan predicates. The implementation translates information needed to compute the truth value of a predicate into parametrized calls of perception routines and interprets the results returned by these routines in terms of the information requested. The third component is the passive perception component, which continually acquires point cloud data obtained from laser sensor sweeps (Chapter 7.3 provides specifics). As a fourth component, K-CoPMan uses KnowRob’s *tell-ask*-interface to communicate with the robot control program. The method *knowrob-query(q)* returns a boolean value depending on whether or not q is implied by the “virtual” knowledge base. knowrob-query-var(var, q) returns the bindings of the query variable *var* which renders the logical expression of the query true. The fifth component consists of KnowRob extension libraries for perceptual memory management, first-order probabilistic reasoning and static environment mapping.

7.2.2 Example Scenario

Let us now consider our example task of bringing missing items to a breakfast table in more detail. Inferring the missing items is a very complex task and requires the integration of heterogeneous information: Where is the table? What is already on the table? What should be there? Where to find the missing items?

Figure 7.2 describes the specification of the missingObjects predicate. The first three conditions in the predicate require the variable Table to be a table in the environment and to have a primary function of having a meal on it. This condition can be met by employing the robot's semantic map of the environment to identify the tables in the environment (visualized in red). The fourth condition tests the set of objects in a given region of interest, which, in our case, is the top of the table, denoted by the variable Table. This condition is checked with the perceptual mechanisms of the K-CoPMan perception server which sets up a perception task to detect, categorize and recognize all the objects on the table and binds the result of this perception task to the Prolog variable Perceived. The next condition specifies the items that are probably needed on the table. To identify these, we use the first-order probabilistic reasoning component. Schematically, KnowRob converts the predicate neededObjectsForMeal into a query *P(on(Obj,Table) | Perceived*$_1$*, ..., Perceived*$_n$), which is then computed for all possible objects. Given the result of this probabilistic query, KnowRob binds the set of objects for which the probability value exceeds some threshold θ to the Prolog variable Needed (e.g. $\theta = 0.5$ or lower, depending on how conservative we want to be). The last condition then determines the missing items Missing as those items that are in the set Needed but not in the set Perceived.

The specification of the predicate perceivedObjectsOnPlane is most relevant, as it actually uses the capabilities of K-CoPMan.

```
perceivedObjectsOnPlane(Plane, Perceived) :-
   onPlane(Plane),
   setOf(Obj-Hyp,
         (  on(Obj, Plane),
            category(Obj,Cat),
            uniqueId(Id),
            objectInstace(Obj,KnownObj),
            Obj-Hyp = [Id,Obj,Cat,KnownObj]),
         Perceived).
```

The condition collects all object hypotheses generated by the perception routine by producing a unique Id for each hypothesis, associating with it the raw sensor data Obj that belongs to the hypothesis, categorizing the object hypothesis (Cat), and checking whether the hypothesis is a known object instance KnownObj, and if so,

which one. The perceptual routines needed for the realization of this condition are explained in Figure 7.3 and the definitions of the predicates in terms of these perception routines can be found in Section 7.4.2.

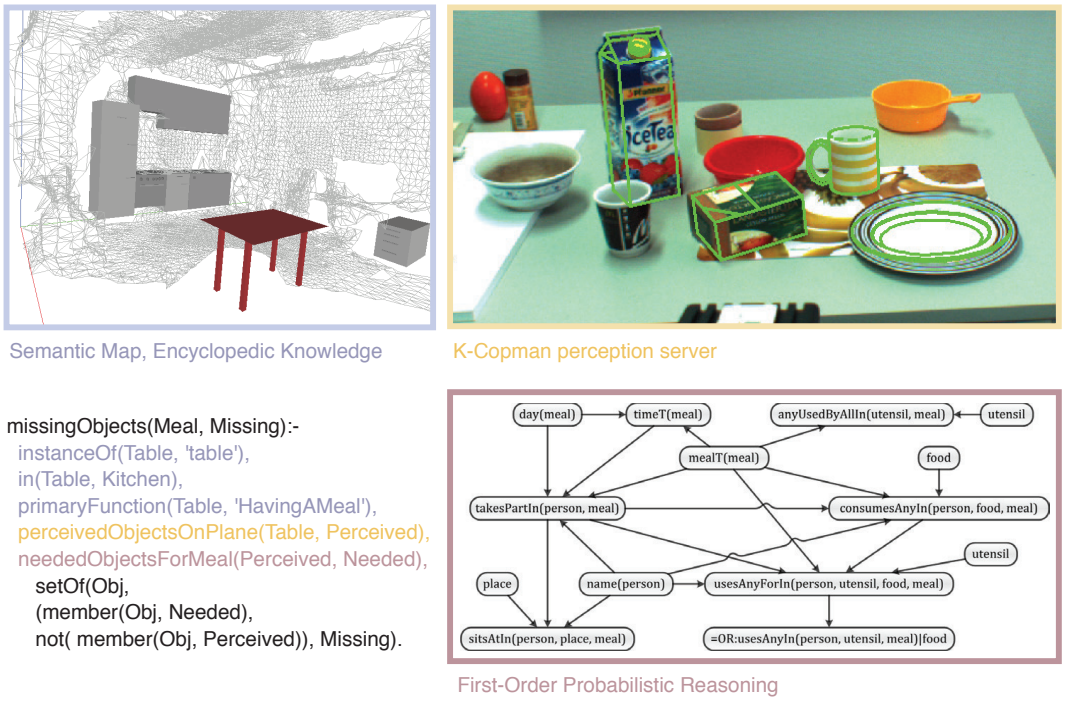


Figure 7.2: *Query to the* K-CoPMan *system for items that are missing on a table with respect to a particular meal. The system first locates the table, perceives the objects on it, queries the probabilistic inference engine for items that are supposed to be on the table and determines those that are missing. BLN graphical model is explained in Section 7.5.*

7.3 Perceptual Models

Let us now explain the perception routines used by K-CoPMan, the passive perception, and the storage and management of perceptual data in more detail.

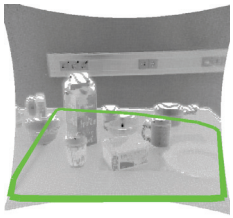
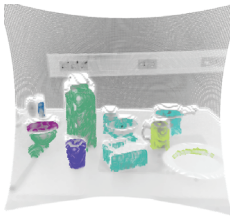
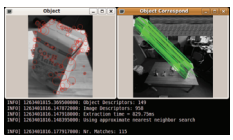
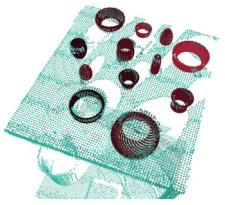
op	abstract routine call	functionality	example results
find-hor-planes	*plane-hyp* ← perceive an object cat. *plane* ori. *horizontal* size $\geq 0.25m^2$	The routine *find-hor-planes(pointcloud)* estimates surface normals based on local neighborhoods performs region growing on the points with approximately vertical normals. The routine then estimates the best horizontal plane using sample consensus and the minimal bounds thereof (for details see [Klank et al., 2009] and Chapter 6.2).	
find-clusters	*obj-hyp* ← perceive an object cat. *pcd-cluster* on *hor-plane*	The routine *find-clusters(pl)* is called with the symbolic name *pl* of a horizontal plane as its parameter and returns a set of names of object hypotheses that are perceived as being supported by *pl* as its result. Each hypothesis name is associated with a subset of point cloud data, which are marked in different colors in the picture on the right.	
match-cad	given *obj-hyp* examine *obj-hyp* object-identity object-pose	*match-cad(obj-hyp, 2D-image)* gets an object hypothesis *obj-hyp* and a 2D color image as its input and performs CAD model matching on the image region that corresponds to *obj-hyp*. The routine returns the object identity of the matching model in the object database and determines the pose of the object. (see [Ulrich et al., 2009]).	
match-sift	given *obj-hyp* examine *obj-hyp* object-identity object-pose	*match-sift(pcd-cluster, 2D image)* finds objects in a 2D image using *ODUfinder* as presented in Chapter 6.	
reconstruct-object	given *obj-hyp* examine *obj-hyp* object-identity surface-of-revolution	The routine *reconstruct-object(pcd-cluster, rotation-axis)* [Blodow et al., 2009] detects surfaces of revolution in point clouds reliably and efficiently. Symmetry assumptions can be hypothesized and verified in order to complete the model from a single view, i.e. to generate data on the occluded parts of the object. These complete models can be used for grasp analysis.	

Figure 7.3: *Some of the perception routines used by the* K-CoPMan *as implemented in the Perception Server (Section 6.5), their procedure call interface, their functionality and an example result. For the full list see [Marton et al., 2011].*

7.3.1 Perception Routines

The K-CoPMan uses a set of perception mechanisms for images and point cloud data, including the detection of horizontal planes, point cloud clustering, CAD model matching, and SIFT-based classification. The mechanisms that are most important for this chapter are listed in Figure 7.3, which shows the name of the routine, how it can be called abstractly, a short functional description, and some sample results. These routines can be used both for task-directed perception and for the passive perception.

7.3.2 Passive Perception

The passive perception component of the perception server (Chapter 6.5) is a key mechanism used in K-CoPMan, which makes the robot environment-aware. It searches the point clouds of the shoulder laser scanner for regions of interest (using SOM$^+$s as a prior), such as tables or cupboards. Whenever it finds such a region, it clusters the point cloud data in order to segment objects standing on top of it. A unique identifier is generated for each of these clusters and asserted to the knowledge base (*Object Model Database*), together with information on the region the time at which it was perceived. The identifier can later be used in conjunction with the perception server to further examine the cluster, e.g. to categorize/classify the corresponding object.

The example in Figure 7.4 illustrates the information that is saved for point cloud clusters. Until the object type is determined, K-CoPMan only knows that it is a Thing, the region of interest it was detected in (here: roi2), the position of the cluster center, and the corresponding point cloud data.

7.3.3 Perceptual Memory

The perceptual memory stores all percepts, making them accessible to future queries. For performance reasons, computations are performed on demand. The passive perception module, for instance, only segments the observed point cloud data and saves the clusters in the memory. Any further processing, such as the classification of the

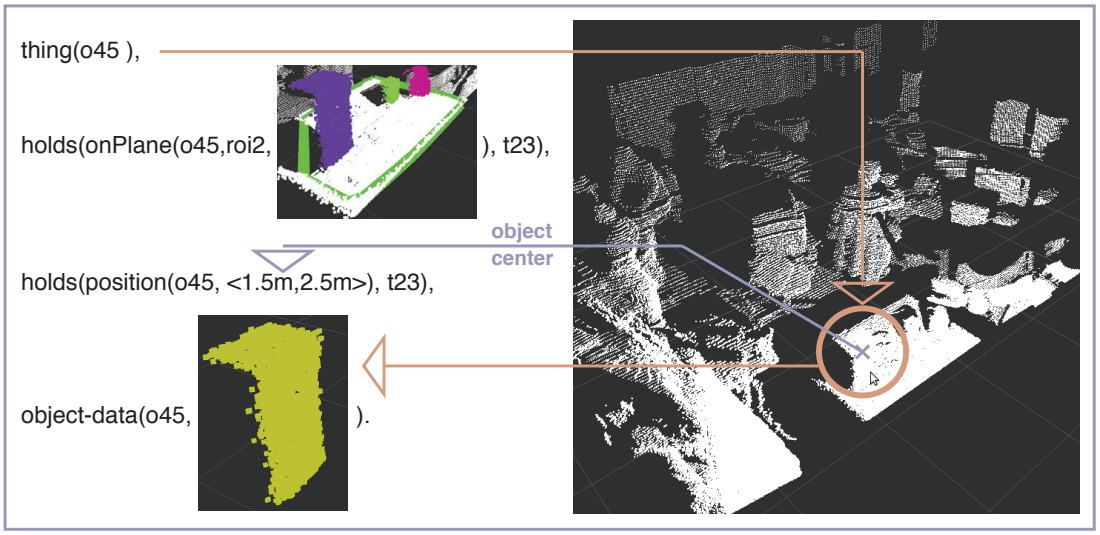


Figure 7.4: *Information stored in the symbolic knowledge base about a (not yet fully classified) object that was detected on a table.*

observed objects, is postponed until the information is required for queries involving the respective object identifiers.

7.4 Integration with the KnowRob

The knowledge processing part in K-CoPMan is based on KnowRob which is, as already alluded before, specialized in integrating sensor data into the knowledge processing system to perform reasoning on observations from the real world. For K-CoPMan, we extended KnowRob with an interface to the perception server described in Chapter 6.5. This allows for direct reasoning on the perceived objects and their properties, and for applying perception routines to them.

The perception routines described in Section 7.3 are embedded into Prolog using the foreign language interface (FLI). Prolog predicates are linked to the functions in the perception server and evaluated by calling the corresponding perception routine.

7.4.1 Computable Relations

External data can easily be integrated using *computable* relations, which allow to determine whether a relation between object instances holds not only on the static knowledge in the system, but also by querying external data sources. Computables are calculated on demand during the reasoning process.

In the K-CoPMan system, computables use attached perception routines to check if a relation holds or not. For instance, the relation typeOf(Obj, Type) is evaluated internally by the K-CoPMan predicate categorize(Id, Type), and the *class* property of an object is determined by a SIFT-based classification method. In addition to loading data into the system, *computable* relations can also be used to calculate qualitative spatial relations based on the objects' positions, e.g. to determine whether an object is on a table. For these relations, the query is not passed to the perception server, but to a small Prolog program that reads the object positions and dimensions and checks whether the relation holds.

7.4.2 K-CoPMan Predicates

In the following, we list the most relevant K-CoPMan predicates for the implementation of the *perceivedObjectsOnPlane* predicate:

holds(onPlane(Obj,Plane),ti) is true if Obj refers to the raw data of an object hypothesis detected by the perception server when looking at plane Plane at time instance *ti*. The predicate is implemented using the perception routines *find-hor-planes* and *find-clusters* (see Figure 7.3).

holds(position(Obj,Pos),ti) is true if *Pos* is the center of mass of the last detection of the object hypothesis *Obj* before *ti*.

holds(spatial-rel(Obj$_1$,Obj$_2$),ti) is true if the object hypotheses Obj_1 and Obj_2 were last detected at the positions Pos_1 and Pos_2, and if these positions satisfy the constraints for *spatial-rel*, e.g. *left-of*. At the moment, we use hard-coded rules to define the spatial relations that depend on the pair of objects at hand but we plan to expand this.

categorize(Obj, Cat) evaluates to true if the point cloud cluster identified by *Obj* can be classified as *Cat* by one of the perception routines in K-CoPMan. Depending on the perception routine, *Cat* can either be a geometric category, e.g. a cylinder, or an object class like a milk box.

7.5 Probabilistic First-Order Reasoning

In order to cope with non-deterministic domains, we integrated statistical models, in particular statistical relational models, into our knowledge processing system. By abstracting away from concrete entities and instead representing general principles (of statistical nature) about a domain, statistical relational models represent meta-models for the construction of concrete probability distributions – represented as graphical models – for a given domain of course, i.e. a concrete set of entities that are of interest (see [Getoor and Taskar, 2007]). Specifically, we use Bayesian Logic Networks (BLN) [Jain et al., 2009], a formalism that combines statistical knowledge (in fragments representing conditional probability distributions) with logical knowledge (sentences in first-order logic). For a given set of entities, a BLN can be instantiated to obtain a ground mixed network [Mateescu and Dechter, 2008] or auxiliary Bayesian network that represents a full-joint probability distribution over the relevant propositions about these entities. Given a model structure and a sufficient amount of relational data – taken directly from our relational knowledge processing system – the parameters of a BLN with given dependency structure can easily be learned, yielding a quantitative representation of statistical dependencies inherent in the data.

To realize our example application (Section 7.2.2), we constructed a model that represents statistical knowledge about table settings. For this model, we used synthetic training data which was generated based on a stochastic process that considered the preferences and habits of six individuals. The model considers the types of meals, the people participating in them (whose preferences the model reflects), the places at which these people sit, the food and drinks they consume as well as the utensils they use to do so. The latter two types of objects constitute the same categories as available in the perception server of K-CoPMan. The model's conditional probability fragment structure is shown in Figure 7.2 (right). Given a partial table setting for one

or more persons, the model can be used to infer the probability with which further utensils or food and drinks might be required. Using an appropriately chosen probability threshold, we can thus flexibly perform the task of completing a table setting based on the information we are given.

7.6 Results

We apply K-CoPMAN to the autonomous mobile manipulation robot TUM-Rosie[1] depicted in Figure 7.1 (right), which is to perform everyday manipulation activities such as setting the table in a kitchen environment. K-CoPMAN controls and uses the sensor system shown in Figure 7.5. A pair of high-resolution color cameras, a stereo-on-the-chip camera, and a time-of-flight sensor on a pan-tilt sensor head are used for task-directed perception. In addition, a tilting laser scanner mounted on the robot's shoulder continually acquires depth maps of the scene in front of the robot (which are mostly used by the passive perception module).

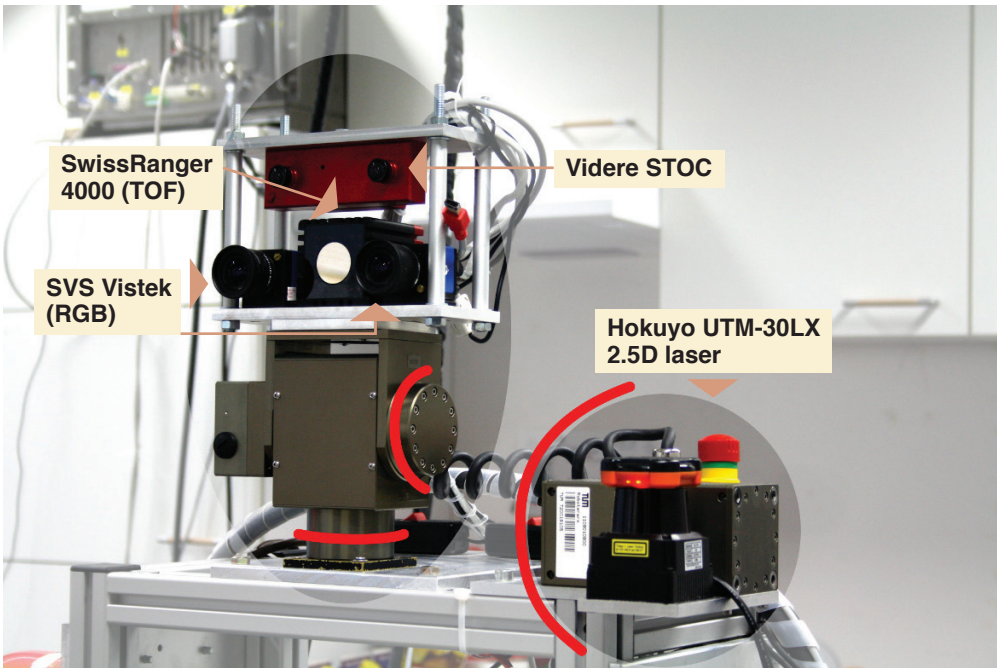


Figure 7.5: *Setup of the sensor head.*

[1]The system was later tested on a TUM-James robot as well with the head-mounted Kinect sensor.

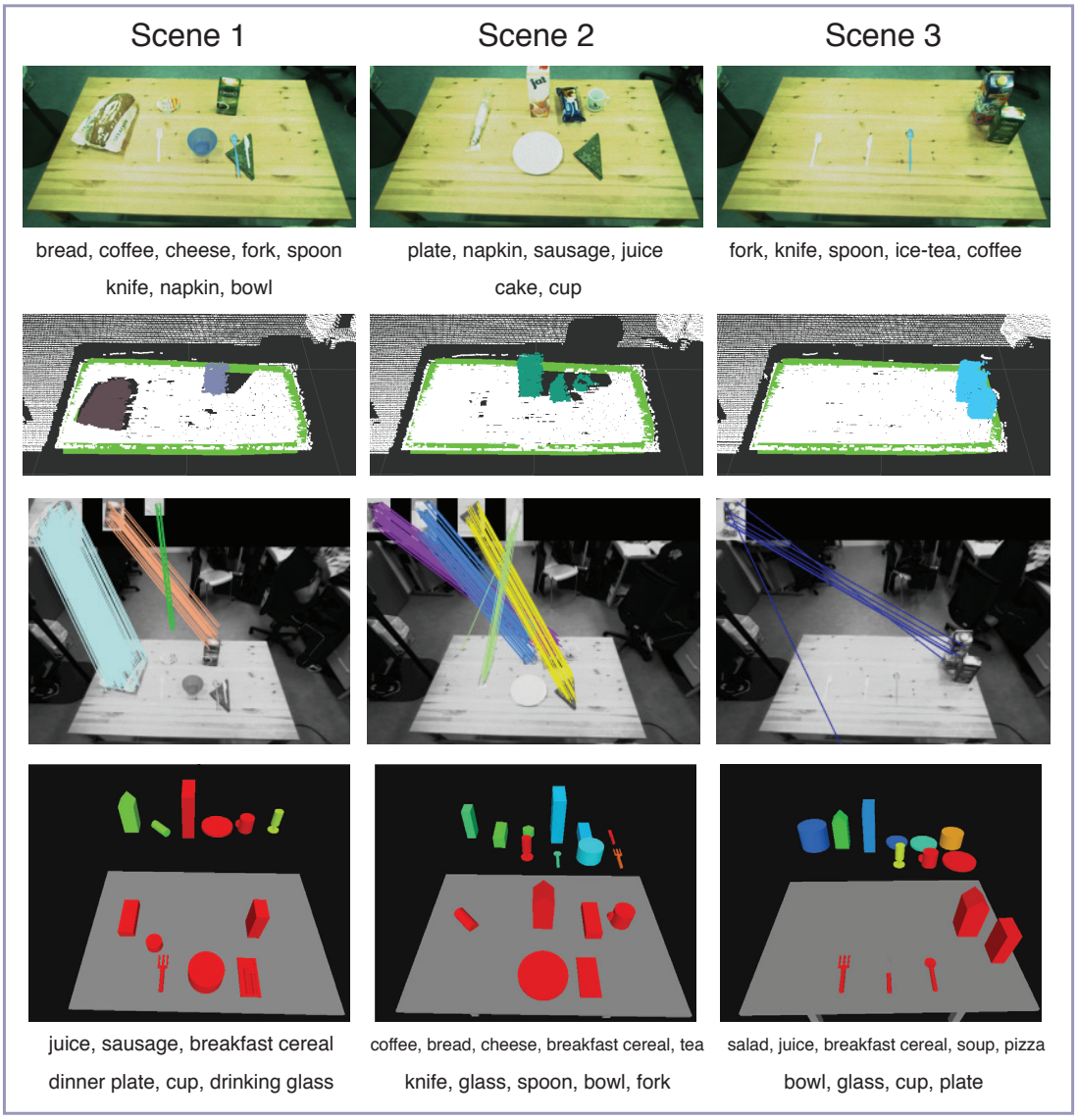


Figure 7.6: *Evaluation results for meal type breakfast.* ***1st row:*** *Snapshots of test scenes;* ***2nd row:*** *object hypotheses;* ***3rd row:*** *detection of objects using* matchsift *routine;* ***4th row:*** *results of probabilistic inference for* missingObjects query. *Below enlisted objects correspond to the inferred ones (visualized off the table) in left-to-right rear-to-front order. Part of the figure courtesy@Tenorth.*

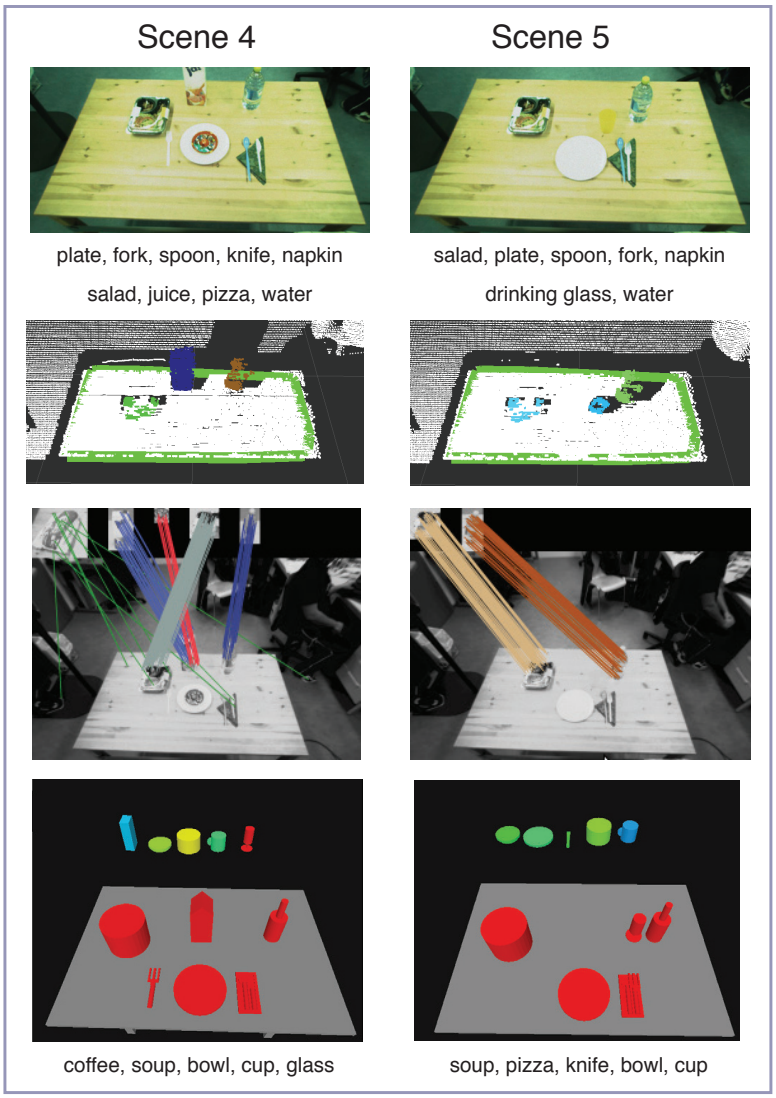


Figure 7.7: *Evaluation results for meal type lunch.* ***1st row:*** *Snapshots of test scenes;* ***2nd row:*** *object hypotheses;* ***3rd row:*** *detection of objects using* matchsift *routine;* ***4th row:*** *results of probabilistic inference for* missingObjects query. *Below enlisted objects correspond to the inferred ones (visualized off the table) in left-to-right rear-to-front order. Part of the figure courtesy@Tenorth.*

To validate our proposed framework, we performed several experiments on the table scenes depicted in the first row of Figure 7.6 and Figure 7.7. As we will show, the integrated system can help a robot system make the decisions required for competent operation in the presence of uncertainty. In addition to the example of inferring missing objects, we will present further queries showing the advantages of integrating perception, knowledge processing and probabilistic reasoning.

Scenes 1-3 in Figure 7.6 show incomplete breakfast settings, whereas scenes 4 and 5 in Figure 7.7 are incomplete lunch settings. The task is to infer which items need to be added to complete the setup. The first row shows the incomplete setup and the lists of objects they involve. In the second row, the table surfaces and clusters identified in the point cloud data are drawn. The clusters were projected onto 2D images and classified with the *match-sift* routine (third row). In the remaining, unoccupied parts of the images, we searched for further objects using combinations of our perception routines.

The set of perceived objects was read into the *KnowRob* system and passed as evidence to the probabilistic reasoning engine which, based on the model described in the previous subsection, infers the table setting that is most likely to be desired. The bottom row visualizes the perceived objects (visualized on the table) and inferred objects (visualized off the table) as instantiated in *KnowRob*. The hue indicates probability: Red corresponds to 1.0, with orange, yellow, green and blue denoting declining probabilities in this order.

As an example query, consider the fourth scene in Figure 7.7. In terms of the functions and predicates the model considers, the query for potentially missing entities translates to a probabilistic query as follows,

$$
\begin{aligned}
&P(usesAnyIn(P, ?u, M), consumesAnyIn(P, ?f, M) \mid mealT(M) = Lunch \wedge \\
&\quad usesAnyIn(P, Plate, M) \wedge usesAnyIn(P, Knife, M) \wedge \\
&\quad usesAnyIn(P, Fork, M) \wedge usesAnyIn(P, Spoon, M) \wedge \\
&\quad usesAnyIn(P, Napkin, M) \wedge consumesAnyIn(P, Salad, M) \wedge \\
&\quad consumesAnyIn(P, Pizza, M) \wedge consumesAnyIn(P, Juice, M) \wedge \\
&\quad consumesAnyIn(P, Water, M) \wedge takesPartIn(P, M)) \\
&\approx \langle\langle\ Glass: 1.00, Bowl: 0.85, Cup: 0.51, \ldots\rangle, \\
&\quad \langle\ Soup: 0.82, Coffee: 0.41, Tea: 0.14, \ldots\rangle\rangle
\end{aligned}
$$

where *P* is some person participating in the meal *M*, who is assumed to be using/consuming the objects that were detected, and we ask for the probabilities of corresponding *usesAnyIn* and *consumesAnyIn* atoms. The results above (listed in order of probability) are certainly sensible, given that the presence of a spoon generally implies that something like soup is likely to be consumed, and therefore that a bowl/soup plate is likely to be required. Also, a glass is necessary for the drinks to be consumed.

Further applications of the system, beyond inferring missing objects, are the recognition of an activity/meal based on the objects that were perceived, the detection of misplaced objects (by applying the predicates for computing spatial relations on the perceived objects), and even the identification of potentially superfluous objects (i.e. objects that have a low probability given the other objects).

7.7 Discussion

We presented K-CoPMan, a system that integrates novel perception routines and knowledge processing mechanisms for autonomous robot manipulation. The system abstracts perceptual facts from the real world, utilises symbolic knowledge to boost up perceptual capabilities, and blends in the combination of both in order to answer complex queries such as *what items are missing on the table for a meal*. We verified our approach by showing several queries that demonstrate how the system can contribute to informed decision making.

Since the detected objects are formally represented in the knowledge base, queries can combine object information with background knowledge that describes, for example, their main functionality. For instance, the following query searches for objects that can be used to cut food and that are lying on the table:

```
?- type(Obj, ObjType),
   subClassOf(ObjType, 'KitchenUtensil'),
   onPlane(Obj, T),
   type(Obj, 'Table'),
   primaryFunction(ObjType, 'CuttingFood').
Obj=knife1
```

Chapter 8

Demonstrations

Most of the theoretical contributions of this thesis have also been implemented on our TUM-James and TUM-Rosie robots in a form of a public demonstration where we programmed the robots to i) make pancakes, ii) shop for and store groceries and iii) serve drinks (see Figure 8.1). While the first two demonstrations were carried out in the assistive kitchen laboratory (in front of a large crowd of world renowned roboticists), the last one was firstly carried in the assistive kitchen at Bosch RTC in Palo Alto and secondly during the exhibition at IEEE/RSJ International Conference on Intelligent Robots and Systems 2011 (IROS2011). Running such demonstrations or as we like to call them "feasibility studies" is a way for us to on the one hand get a grasp of the actual problems that household robots will have to tackle as oppose to trying too hard to make them up and thus risk wandering into the false assumptions. On the other hand this also enables us to survey the hardware and the range of available software components and thus judge what their respective limitations are.

Figure 8.1: *Robots carrying out demonstrations.*

For instance, there is a general belief at the moment that the 2D localization for even indoor floors is a solved topic [Fox, 2001]. While this might hold as long as the robot's tasks do not go beyond navigation and obstacle avoidance, the localization is certainly too inaccurate for a very fine and dexterous manipulation where precision rates under $1cm$ are needed. Our suggested solution for this case was thus to act in a sense-act-refine loop and to get localized relative to the scene or objects in question as oppose to doing so in the global map. And this is just the tip of the iceberg, the list of challenges thus also includes perception of objects in varying lighting conditions, limited field of view for the majority of the available sensors (e.g. Kinect), under-actuated hands, slow motion planning algorithms, temperature-dependent behavior of cooking substances (e.g. pancake mix), lack of tactile sensors that would enable an execution of certain tasks in partial contact with the environment, disambiguation when translating the natural instructions into the robot executive programs, lack of clean semantic information when using the object models from WWW and, last but not least, lack of well defined interfaces between software components. In the rest of this chapter we will break down those challenges that became especially prominent in the given demonstrations and elaborate on our proposed solution. The consensus that we can already share in the preface is that most of the issues either stem from years of the segregated development of the respective fields and taking of false assumptions (e.g. in computer vision lighting and blur effects seldom get any attention), and underachieving and expensive hardware.

The algorithms proposed in this thesis are implemented as libraries and expose their functionality through ROS nodes. The communication with other components such as robot's drivers, localization, etc. is made via ROS topics, services and actions. This choice enabled us to use the code from other individuals and groups, and to also relatively well break our code into the computation, communication, coordination, composition, and configuration parts [Prassler et al., 2009].

8.1 Robots Making Pancakes

In this demonstration[1], the robots retrieve instructions for making pancakes from the World Wide Web and generate robot action plans from the instructions. This task

[1] http://www.youtube.com/watch?v=gMhxi1CJI4M

is jointly performed by two autonomous robots: The first robot opens and closes cupboards and drawers, takes a pancake mix from the refrigerator, and hands it to the second robot. The second robot cooks and flips the pancakes, and then delivers them back to the first robot. While the robot plans in the scenario are all percept-guided, they are also limited in different ways and rely on manually implemented sub-plans for parts of the task.

The purpose of this experiment is to show the midterm feasibility of the service robots entering into household environments and more importantly the better understanding of how we can realize control systems with these capabilities by building such a system. We tested various hypotheses such as whether the localization accuracy of a mobile robot suffices to perform tasks such as cabinet door opening, whether successful percept-guided behavior for sophisticated manipulation actions can be generated, whether objects can be detected, recognized and grasped reliably and whether robot plans can be generated from web instructions made for human use.

While the whole demonstration is a result of a large group effort and thoroughly described in a separate publication [Beetz et al., 2011], we will in this section focus on our contributions and discuss the potential of the underlying technologies as well as the research challenges raised by the experiment. For the demonstration we build a SOM$^+$map of the assistive kitchen laboratory as described in Section 4 and use a perception server (Section 6.5) and *ODUfinder*(Section 6) in particular to detect and recognize a pancake mix. While a SOM$^+$map is depicted in Figure 4.2 and includes poses of handles and articulation models of cupboards, we for this experiment use a product information website to learn object attributes (see Figure 8.2).

Using a SOM$^+$map and an object to map association via afore described ontology and reasoning (Figure 6.10) our robots can find objects faster and more reliable. SOM$^+$'s knowledge base namely describes pancake mix as a perishable item and it further contains information about refrigerators, namely that they are household appliances, cooling devices, container artifacts, and that they are storage places for perishable goods. Using perceptual pop-out (Chapter 6.2) inside the refrigerator limits the search space and thus reduces the false positive rate in terms of recognized objects. Utilizing pre-stored knowledge about the poses of handles, knobs and articulation models, our robots are less prone to grasp wrong fixtures in the kitchen and less prone to fail while opening cupboards due to handle slippage or errors in

Figure 8.2: *Picture of a bottle of pancake mix obtained from an online shop.*

continuous estimation of the articulation models. Owing it to the insufficient localization performance, we only use handle and knob poses stored in the SOM$^+$map to find their approximate poses. To actually find and grasp those features, we still have to perform additional re-perception step in the coordinate frame of the target object (e.g. cupboard to be operated) and also perform all mobile manipulation tasks such as door opening relative to the robot's base odometry frame.

8.1.1 Future Challenges

Despite a very encouraging result this demonstration revealed a plethora of further research challenges. For instance we learned that there is a range of perception tasks that the robot must accomplish: it must detect objects, recognize, localize, reconstruct them, it has to calibrate the tools in its hand, it has to monitor the deformation of the pancake. Also the objects and stuff that are to be perceived vary a lot: some objects are textured, others have identifiable forms, others are transparent and others, like plates, are indistinguishable from each other. Objects can also be heavily occluded and positioned in a clutter. For the mapping part we believe that approaches like Kinect Fusion [Izadi et al., 2011] yield enough details and enough accuracy, however a challenge remains in how to continuously update the map as objects move and reliably (industrial strength grade) bridge the gap to the knowledge retrieved from www. For the latter we propose human in the loop approaches such as e.g. proposed by Pitzer et al. [2011]. Robot perception has to thus go far beyond the library of methods that is currently used in the control software and be from the beginning on part of the robot's perception-reasoning-action loop.

In terms of manipulation we learned that robot's hands do not have to necessary get more dexterous for tabletop manipulation but rather more compliant [Deimel and Brock, 2013] and cheaper. Further we found that robots like TUM-James and TUM-Rosie are by enlarge too big and too bulky for actual mobile manipulation in real homes and thus constrained environments [Rühr et al., 2012]. While making smaller and more dexterous robots is well under way [UnboundedRobotics, 2013], there is also a great need for whole body kinematics algorithms and manipulation of objects with both arms (e.g. using hand-over strategy) and both together in dynamic environments.

The demonstration was a feasibility study, and we had to deal with many of the issues. Many aspects have been solved specifically and some actions have been hand-coded. One important aspect of the experiment was that we integrated previously independent research efforts and validated that these efforts can be combined effectively and thus contribute to a successfully integrated solution for robotic household tasks.

8.2 Robots that Shop for and Stores Groceries

In this single-robot demonstration the robot is tasked to simulate a shopping and store object away tasks[2]. Similarly to the experiment above we want to show the midterm feasibility of the service robots entering into household environments and in addition also public sectors such as grocery stores. In addition to the above listed hypotheses to be tested, we also want to test readiness of TUM-James' bi-manual manipulation skills and whether a manipulation can aid and improve robot's perception. Using algorithm for the inference of organizational principles adds another dimension to our work in that robot can actually adapt and personalize its behavior to the human companions.

While the whole demonstration is again a result of a larger group effort and in full documented in a following publication: [Pangercic et al., 2011b], our contribution was major. The robot thus for instance uses the interactive segmentation approach from Chapter 5 by poking the objects located on the shopping shelf. Such generated priors are then fed to the *ODUfinder* (Chapter 6) where the object is recognized, grasped and put into the shopping basket. In this version of the demonstration the objects are not tightly crammed together since this would impose an inevitable grasping constraint. Upon bringing the objects *home* which is simulated by emptying the shopping basket on the empty horizontal surface in the assistive kitchen laboratory, the robot runs an algorithm that determines an organizational principle of an environment (in this case the TUM assistive kitchen laboratory, Chapter 3.1) and infers where to best place the objects. Finally, it grasps the objects and actually stores them away into the allocated place as also depicted in Figure 8.3. To represent both kitchen and shopping store environments we again use SOM$^+$maps as described in Chapter 4. Algorithm to infer the organizational principles has been contributed by Schuster et al.

[2] https://www.youtube.com/watch?v=gbIDPqb_2iM

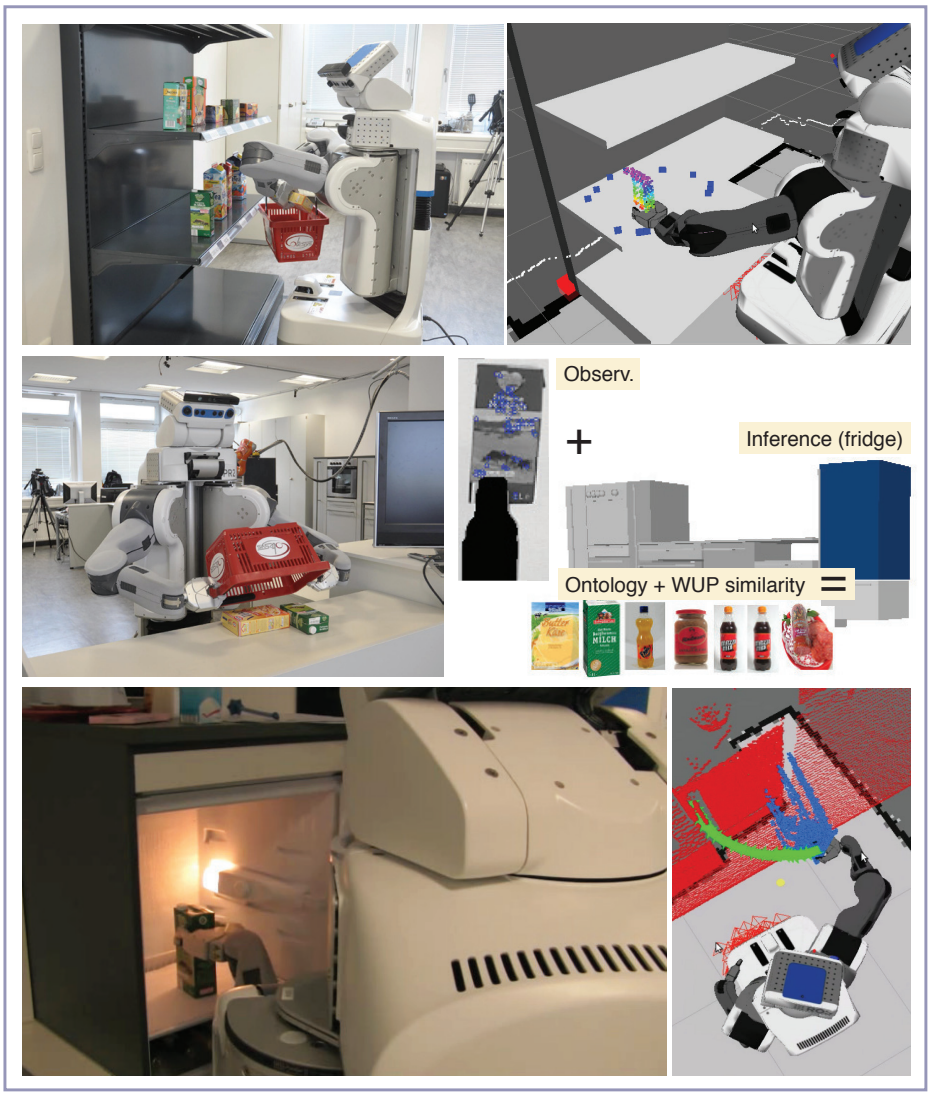


Figure 8.3: *Sequence of screen-shots from the shopping for groceries demonstrations. TUM-James is seen finding objects on the mock-up of the shopping shelf, graping and putting them into the shopping basket and finally bringing them 'home'. In the next step robot uses* ODUfinder *to recognize the object, infers its most probable storage location and stores it away.*

[2012] and requires the robot to first acquire organization principle models given observations of the objects found within a particular environment. In order to determine a suitable storage location for a given object, the algorithm then identifies the class to which the object belongs and performs inference over the model using, in particular, features pertaining to the similarity between the object and the other objects already stored in the environment. More details about this algorithm are available in the paper [Schuster et al., 2012]. For collision-free arm manipulation we rely on planning algorithms available in ROS [Sucan and Chitta, 2013].

Using pipeline for the segmentation of textured objects Chapter 5.2.1, we are able to very successfully generate segmentation priors for *ODUfinder* and then recognize objects even when multiple objects are being stacked on top of each other or co-located side by side as shown in Figure 6.2. Unlike using perceptual pop-out as in the pancake demonstration, we in this case thus show and integrate an alternative tool for the generation of constraints for *ODUfinder* which proves the generality of our object recognition system. SOM$^+$map is built and used to determine spatial location of objects of a given category within the grocery store[3]. On the other hand the same SOM$^+$map as in the pancake demonstration is used to infer storing locations for the objects. For the latter we again use Barcoo website, and generate an ontology that extends the KnowRob ontology with knowledge about more than 7,000 manufactured products. In addition to the object category structure, this online shop also provides detailed descriptions of the properties of products, such as the perishability status, price, ingredients, etc. (see Figure 6.10). The latter proves being especially useful in this demonstration when a storage place for an object milk is requested. Since milk is categorized as a perishable objects and that property is linked to the cooling device refrigerator in the extended KnowRob taxonomy, the latter was inferred as a storage location over a kitchen table where milk could also be found during breakfast. As proposed by Schuster et al. [2012], the following features are used to learn the organizational principles: semantic similarity, purpose, meal relevance, size and shape. These features came as a result of a thorough analysis of photographs of kitchens of our colleagues as well as blogs and videos from the Internet. Concerning bi-manual manipulation, we experienced problems in that used arm planning algorithms currently do not allow for the adequate specification of constraints such as when to use which arm, where to grasp an object given a task at hand and how to hold an ob-

[3] Only verified through the use of one shopping rack in this demonstration.

ject during the manipulation (e.g. up-right if it is filled with the liquid). Bridging the gap between manipulation algorithms and semantic understanding of the course of actions for a given task has been part of our ongoing work [Witzig et al., 2013].

8.2.1 Future Challenges

This demonstration as well revealed several challenging research problems. Unlike in the pancake demonstration, we realized that we would need better and more dexterous grippers to be able to grasp the objects from the shelves or place them into the cluttered containers such as refrigerators. Bi-manual manipulation becomes an absolute must when it comes to the tasks such as shopping. An important aspect when operating in public spaces such as grocery stores becomes reliable navigation in dynamic environments that is also cost effective. Alternatives to laser-based navigation solutions will have to be explored in the future by using e.g. camera and/or ultra sound array sensors. Another challenge is to be able to deal with a truly large scale object recognition. Product mining store Barcoo has currently over 7 million objects in their database and the recognition of objects will currently still not be possible with our solution proposed in Chapter 6. Lastly, we believe that to learn organizational principles for a wide variety of household we will have to turn to the shared autonomy approaches more and devise a solution that will render it possible for the remote operators to collaborate with the robots from remote and help them overcome unforeseen situations.

8.3 Robots Serving Drinks

In this demonstration we programmed another PR2 robot (Bosch-Alan) at another PR2 Beta Site at Bosch RTC in Palo Alto to fetch the drinks from the refrigerator and serve them to visitors[4]. The demo begins with a visitor selecting the drink of choice and the robot then proceeds by finding the refrigerator and opening it. Inside the refrigerator it then has to find a selected drink, grasp it, close the refrigerator door and bring the drink to designated table. This demonstration was quickly and effectively built up using the same tools as in the two demonstrations above, that is SOM$^+$maps

[4]http://www.youtube.com/watch?v=z36xkUILtQE

for being able to understand the environment and operate in it, and *ODUfinder* for object detection and recognition. The major difference to previous two demonstration however is in that it took place in two completely different environments. Firstly in Bosch's assistive kitchen and secondly in the exhibition center of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2011. This at least to the extent proves that we developed and built tools that are generic and sustainable. The only real difference that we encountered in this environment were handles that have specular reflection and for which we had to develop a special type of detector as presented in Section 4.4.2.2. The most important lesson learned however is that standardized hardware (PR2 robot in this case) and standardized and quality software (ROS) sped up our development and deployment substantially.

8.4 Discussion

In this chapter we presented three demonstrations using human-like robots in environments that resembled households as closely as possible. The robots were tasked with everyday chores such as serving, cooking and shopping. The algorithms developed in this thesis, in particular SOM$^+$maps, interactive object segmentation and *ODUfinder* were integrated successfully and they performed robustly with respect to robustness, false positive rates and different environments. In general we observed that robots perform better if they are equipped with the domain knowledge and their algorithms for perception, reasoning and action are tightly coupled together - which is exactly where this thesis is placed. The coupling becomes increasingly easier if standardized hardware and software is available. Regarding respective research fields, the most and the best tools are available in the field of perception which is most likely due to the emergence of cheap sensors such as Kinect and availability of excellent open-source tools such as OpenCV and PointCloud library. Recently we are also witnessing a so-called ensemble of experts system being adopted widely. In this systems, like in our perception server (Chapter 6.5), various perception algorithms work together with an aim to maximize the performance. In terms of manipulation there is in general a lack of cheap, precise and compliant hardware. In addition, most grasp and motion planning algorithms do not provide an option to impose semantic constraints and thus do not allow for encoding of task dependent knowledge. Very few work has been so far done on the reasoning and artificial intelligence driven

robotics. For the robots, to be able to live with and serve humans, they will have to understand their environment and their task and have an ability to monitor execution of their tasks, detect and recover from failures and also learn. We believe that this thesis contributes substantially into this direction which has been validated through above presented demonstrations. However, to fully understand the problems in the real world and to make progress quickly two things will need to happen. On the one hand benchmarks will have to be developed and used to be able to test and compare systems developed by different researchers. From that point on a set of best practices will need to be developed that will prevent community to reinvent the wheel over and over again. On the other hand to be able to bring service robots on the market quickly a concept of shared autonomy will need to be adopted widely in order for humans to assist service robots with the high intelligence tasks.

Chapter 9

Conclusion

This chapter concludes the thesis by summarizing the main contributions of our research and providing a line of possible extensions that we either already work on or encourage the research community to continue pursuing.

The thesis has been from the ground on motivated with a real use case as presented in Chapter 1.3. An entire household task has been selected because we on the one hand believe (and work actively towards) that personal robots will penetrate into homes beyond vacuum cleaners and toys, and on the other hand because this kept us focused on developing a solution for the problem rather than finding a problem for an out-of-space solution. After deliberately analyzing the whole "prepare pancakes for breakfast" task we identified two main issues. First, research in robotics has been since Shakey times [Nilsson, 1984] very divided into sub-fields (e.g. computer vision, motion and grasp planning, mapping and localization, etc.) and there has been very few exceptions [Asfour et al., 2000] where researchers tried to integrate tools and algorithms from multiple fields in one realistic and useful robotic application. Secondly, we noticed that almost always the domain knowledge pertaining to a certain task has been either over- or under-simplified. While the former issue leads to incomplete solutions and omittance of helpful clues, the latter one either results in continuous reinventing of the wheel or really sloppily designed solutions. To cover up for these deficiencies we decided to work on the solutions that will make use of knowledge bases, KnowRob in our case, and always strive to have our algorithms in the perception-reason-act loop of the robot.

We first started with the mapping of the environment as presented in Chapter 4. With the SOM$^+$concept that we propose and implement, the robot can autonomously

acquire photo-realistic meshes of the indoor environments and segment the meshes into various fixtures such as planes, knobs and handles. Additionally the robot can also grasp the handles and interactively segment furniture faces and learn articulation models. The results of these computation step are then abstracted away and asserted into a hierarchical ontology of the knowledge base. The knowledge base contains three sources of the knowledge: terminological, assertional and spatial and allows for powerful queries that unify low-level information like the dimensions and poses of objects with semantic information about their properties obtained from encyclopedic sources for instance. In the end we qualitatively and quantitatively evaluated this concept and confirmed that SOM$^+$ maps can be built in various environments and in real time and that robots using maps are significantly faster and robuster when acting in the indoor environments if using a priori stored information.

Next we tackled detection and recognition of furniture objects and objects of daily use. While the former has been published in a separate publication [Mozos et al., 2011], we in Chapter 5 present a novel, interactive perception approach for the segmentation of objects of daily use. In this approach we use robot arm to induce motions into the clutter of objects, observe and track features on the objects and in the end cluster rigidly moving features as belonging to one object or otherwise. This approach shifts the classical paradigm from sense-act to sense-act-sense and by tightly connecting robot's perception and manipulation skills yields superior results to those that are based on the segmentation of static images only. In our approach we tackle textured and textureless objects separately when it comes to feature extraction and tracking and subsequent trajectory clustering but reuse part of the segmentation pipeline that requires computation of the push points, dense model reconstruction and actual pushing. We use such obtained densely reconstructed models to grasp objects or to constrain object recognition task as presented in Chapter 6 to a specific region of interest. Recent robotics challenges[1] have exposed a large need for interactive perception tools and algorithms in order to make robots more autonomous and thus on the market faster.

In Chapter 6 we present an object recognition system that scales well with the large number (couple of thousand) of objects and is also linked with the KnowRob knowledge base. The system uses SIFT features and vocabulary trees to perform search

[1] http://www.theroboticschallenge.org

efficiently. For robustness we introduced an additional measure based on the clustering of SIFT features which significantly improves the performance of the system in the presence of occlusions or minor changes in object's appearance due to tear and wear. The system also uses an external barcode reader library which we connected to the product information site Barcoo which provides a taxonomy of more than seven million object of daily use classes. Having such a taxonomy and object properties such as e.g. perishability, allows us to connect this ontology with the above ontology of the SOM$^+$ environment model and then execute powerful queries such as "Where does this object belong to?" or "Where do I find an object of type milk"? This system is also integrated into the perceptual server which was jointly developed in the IAS group and features various other object detection, categorization and recognition algorithms. With this system we have on the one hand shown that fast, scalable and robust object recognition is possible within a real robotic application. On the other hand, by linking Barcoo and SOM$^+$ we extended a range of possible queries that the robot can pose to its knowledge base by an order of magnitude.

In our final contribution in Chapter 7 we effectively combine results of Chapter 4, Chapter 5 and Chapter 6 and add statistical relational learning in the mix. Such resulting system termed K-CoPMan can now test and satisfy knowledge preconditions for everyday manipulation. As an example, the robot using this system can now perceive the tabletop scene and then infer which objects are missing and then go and fetch them. In a nutshell the system employs two main perceptual mechanisms: task-directed and passive perception. *Task-directed perception* provides information necessary for accomplishing manipulation tasks – information about the object to be acted on and the scene context. The *passive perception* is to make the robot environment-aware by also memorizing objects that are not task-relevant at the time of perception. Using knowledge processing mechanisms and the belief state (memory) of the robot, the robot can for instance point the camera at places where it believes objects to be or exploit the fact that objects inside a cupboard are invisible unless the door is open. Finally, to cope with non-deterministic domains, we integrated statistical models, in particular statistical relational models (BLN), into our knowledge processing system.

9.1 Future Work

Though above systems have been thoroughly validated and have shown great promise en route of introducing robots into everyday environments, there is still plenty of research issues to be addressed.

With SOM$^+$ maps we propose an inclusion of the recent surface reconstruction approaches that are based on the work by Izadi et al. [2011], most notably a work by Whelan et al. [2012] who overcome memory limitations of the former by introducing factor graphs in the pipeline and thus enabled surface reconstruction of really large environments. Further, solutions for dealing with the dynamically changing environments will have to be explored and solved. Though the cases when large objects such as cupboards, tables, chairs, etc. are moved around are rare, we still need to update our maps accordingly. We propose to either use our passive perception model from K-CoPMan and continuously and lazy monitor the environment or introduce the human in the loop as suggested by Pitzer et al. [2011]. Further remaining issues involve the level of detail of the interpretation of reconstructed surfaces, modeling of the inside of cupboards and interactive acquisition of the data (e.g. interactive segmentation and articulation model learning) in constrained environments. On the knowledge side our objective is to be able to automatize creation of the ontologies for the environment modeling by e.g. sourcing information from the furniture stores such as IKEA. This will help us to skip the manual alignment step as reported earlier. Further we plan to introduce more and even more powerful queries and test and validate them rigorously in a larger set of real household environments. For that we believe that, we as a community, have to come up with a benchmark that will consist of datasets from numerous environments and enable us to simulate robot's actions in those environments. We propose a use of simulators such as Gazebo [Koenig and Howard, 2004], where photo-realistic meshes and their interpretations can be inserted and the robot's behavior simulated physically correctly.

In the interactive object segmentation work we plan to fuse both approaches for the segmentation of textured and textureless objects. Currently both pipelines diverge in two places, in the feature estimation and tracking step and in the trajectory segmentation step. For the former, a measure to estimate the texturedness in a certain scene will need to be developed. For the latter we plan to either make the choice

of the segmentation algorithm dependent on the selected feature or we will try to make graph-based segmentation faster to deal with the larger number of features and thus textured objects as well. In addition we plan to integrate an arm motion planning component [Sucan and Chitta, 2013] which will allow us to execute safe and sound push action. One missing component in the presented setup are transparent and translucent objects. We have recently seen detectors [Klank et al., 2011] and recognizers [Lysenkov et al., 2012] that are for the most of the time exploiting a deficiency of the time-of-flight to perceive transparent objects and take the lack of the sensor data as an indication of the presence of such objects. While this is a valid assumption and we plan to adopt it, the lack of data can also be caused by other phenomena in the environment such as the presence of glass or a strong sun light. To overcome this we plan to integrate ultra sound array sensors on robot's head and tactile sensors [Romano, 2011] on robot's hands to be able to test and validate if transparent objects have been detected and recognized correctly. Finally, we plan to introduce the interactive perception approach in other tasks such as navigation in dynamic environments, where obstacle would need to be removed, as well. Further interesting tasks to consider are validation of the pose estimation on the recognized objects or relative localization with respect to the object of interest, e.g. a refrigerator.

Emergence of cheap sensors such as Kinect [WillowGarage, 2010b] or SoftKinetic [SoftKinetic, 2012], have triggered and already substantially improved research in the field of object of daily use detection and recognition [Lai et al., 2011b; Hinterstoisser et al., 2012; Collet Romea et al., 2011]. The remaining challenges include fusing of all available algorithms in an even larger ensemble of experts system that will learn on the fly which algorithms to use given a certain situation and also learn the necessary parameters. Using UIMA [Ferrucci and Lally, 2004] architecture which was used in the famous Jeopardy quiz by Watson computer is something that we are already exploring and that with very promising results. UIMA architecture in the Watson setup is used to analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user. Another remaining problem is on how to capture and incorporate all possible object models in the world. While we currently do not have a direct answer to this question, we definitely believe that working together with large retailer stores and companies such as Google that have computational resources as well access to online data, is the way to go. Expanding Google's

freebase [Bollacker et al., 2008] with the necessary structs to hold terminological, assertional and spatial information is one of the possible solutions.

Bibliography

Robert Adelmann. C.: Toolkit for Bar Code Recognition and Resolving on Camera. In *Phones - Jump Starting the Internet of Things. In: Informatik 2006 workshop on Mobile and Embedded Interactive Systems*, 2006.

Georg Arbeiter, Jan Fischer, and Alexander Verl. 3D Perception and Modeling for Manipulation on Care-O-bot 3. In *In Proceedings of the ICRA 2010 Workshop: Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

Tamim Asfour, Karsten Berns, and Rüdiger Dillmann. The humanoid robot ARMAR: Design and control. In *IN IEEE/APS INTL CONFERENCE ON HUMANOID ROBOTS*, pages 7–8, 2000.

Pedram Azad, David Munch, Tamim Asfour, and Rüdiger Dillmann. 6-DoF model-based tracking of arbitrarily shaped 3D objects. In *ICRA*, 2011.

Michael Beetz, Dominik Jain, Lorenz Mösenlechner, Moritz Tenorth, Lars Kunze, Nico Blodow, and Dejan Pangercic. Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence. *Proceedings of the IEEE, Special Issue on Quality of Life Technology*, 100(8):2454–2471, 2012.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic Roommates Making Pancakes. In *11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017, 2010.

Michael Beetz, Freek Stulp, Bernd Radig, Jan Bandouch, Nico Blodow, Mihai Dolha, Andreas Fedrizzi, Dominik Jain, Uli Klank, Ingo Kresse, Alexis Maldonado, Zoltan Marton, Lorenz Mösenlechner, Federico Ruiz, Radu Bogdan Rusu, and Moritz Tenorth. The Assistive

Kitchen – A Demonstration Scenario for Cognitive Technical Systems. In *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, pages 1–8, 2008. Invited paper.

Niklas Bergström, Carl Henrik Ek, Mårten Björkman, and Danica Kragic. Scene Understanding through Interactive Perception. In *In 8th International Conference on Computer Vision Systems (ICVS)*, 2011.

Christian Bersch, Dejan Pangercic, Sarah Osentoski, Karol Hausman, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Segmentation of Textured and Textureless Objects through Interactive Perception. In *RSS Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012.

Paul J. Besl and Neil D. McKay. A Method for Registration of 3D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

Nico Blodow, Lucian Cosmin Goron, Zoltan-Csaba Marton, Dejan Pangercic, Thomas Rühr, Moritz Tenorth, and Michael Beetz. Autonomous Semantic Mapping for Robots Performing Everyday Manipulation Tasks in Kitchen Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

Nico Blodow, Dominik Jain, Zoltan-Csaba Marton, and Michael Beetz. Perception and Probabilistic Anchoring for Dynamic World State Logging. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 160–166, 2010.

Nico Blodow, Radu Bogdan Rusu, Zoltan Csaba Marton, and Michael Beetz. Partial View Modeling and Validation in 3D Laser Scans for Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2009.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

Jean Y. Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm. Jean-YvesBouguet, 2002.

Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26: 1124–1137, 2004.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Jeff Brown et al. ZBar Bar Code Reader, 2011.

Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 282–295. Springer-Verlag, 2010.

J. Bruce, Tucker Balch, and Maria Manuela Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061 – 2066, 2000.

Lillian Chang, Joshua R. Smith, and Dieter Fox. Interactive singulation of objects from a pile. *International Conference on Robotics and Automation (ICRA)*, 2012.

Matei Ciocarlie, Kaijen Hsiao, E. Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan A. Sucan. Towards Reliable Grasping and Manipulation in Household Environments. In *Proceedings of RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

Code128. Information technology – Automatic identification and data capture techniques – Bar code symbology specification – Code 128, 2000.

Leslie B. Cohen and Cara H. Cashon. Infant Perception and Cognition. In *Comprehensive Handbook of Psychology, Volume 6: Developmental Psychology*, chapter II. Infancy, pages 65–89. Wiley and Sons, 2003.

Alvaro Collet Romea, Manuel Martinez Torres, and Siddhartha Srinivasa. The MOPED framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research*, 30(10):1284 – 1306, 2011.

DataBar. Information technology – Automatic identification and data capture techniques – GS1 DataBar bar code symbology specification, 2006.

Raphael Deimel and Oliver Brock. A Compliant Hand Based on a Novel Pneumatic Actuator. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–07, 2013.

Birgid Eberhardt. Service Robotics – an Opportunity for Demographic Change. 2012.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

David Ferrucci and Adam Lally. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10 (3-4):327–348, 2004.

Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

P. Fitzpatrick. First contact: an active vision approach to segmentation. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2003.

D. Fox. KLD-Sampling: Adaptive Particle Filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2007.

Inc Google. *Google Voice Search*.

G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3472–3478, 2007.

GS1. GS1 General Specifications, 2010.

Megha Gupta and Gaurav S. Sukhatme. Using Manipulation Primitives for Brick Sorting in Clutter. *International Conference on Robotics and Automation (ICRA)*, 2012.

Rakesh Gupta and Mykel J. Kochenderfer. Common Sense Data Acquisition for Indoor Mobile Robots. In *AAAI*, pages 605–610, 2004.

Martin Haegele. World Robotics - Service Robots 2011. Technical report, International Federation of Robotics, 2011.

Karol Hausman, Ferenc Balint-Benczedi, Dejan Pangercic, Zoltan-Csaba Marton, Ryohei Ueda, Kei Okada, and Michael Beetz. Tracking-based Interactive Segmentation of Textureless Objects. In *In IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany*, 2013.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *ISER 2010*, 2010.

Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph. Exploring Expression Data: Identification and Analysis of Coexpressed Genes. *Genome Research*, 9(11):1106–1115, 1999.

S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Texture-Less Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, 2011.

David W. Jacobs. Perceptual Organization As Generic Object Recognition. In *From Fragments to Objects - Segmentation and Grouping in Vision*, chapter IV. Models Of Segmentation And Grouping, pages 295–329. 2001.

A. Jain and C.C. Kemp. Pulling Open Doors and Drawers: Coordinating an Omni-directional Base and a Compliant Arm with Equilibrium Point Control. In *ICRA*, 2010.

Dominik Jain, Stefan Waldherr, and Michael Beetz. Bayesian Logic Networks. Technical report, IAS Group, Fakultät für Informatik, Technische Universität München, 2009.

Dominik Joho, Martin Senk, and Wolfram Burgard. Learning Search Heuristics for Finding Objects in Structured Environments. *Robotics and Autonomous Systems*, 59(5):319–328, 2011.

Asako Kanezaki, Zoltan-Csaba Marton, Dejan Pangercic, Tatsuya Harada, Yasuo Kuniyoshi, and Michael Beetz. Voxelized Shape and Color Histograms for RGB-D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, 2011.

Dov Katz and Oliver Brock. Interactive Segmentation of Articulated Objects in 3D. In *Workshop on Mobile Manipulation at ICRA*, 2011.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *4th Eurographics symposium on Geometry processing*, pages 61–70, 2006.

Jacqueline Kenney, Thomas Buckley, and Oliver Brock. Interactive segmentation for manipulation in unstructured environments. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, 2009.

KIT. Kit Object Models Web Database, 2010.

Ulrich Klank, Daniel Carton, and Michael Beetz. Transparent Object Detection and Reconstruction on a Mobile Platform. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. Real-time CAD Model Matching for Mobile Manipulation and Grasping. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 290–296, 2009.

Jens Klappstein, Tobi Vaudrey, Clemens Rabe, Andreas Wedel, and Reinhard Klette. Moving Object Segmentation Using Optical Flow and Depth Information. In *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, PSIVT '09, pages 611–623. Springer-Verlag, 2008.

E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib. Grasping with Application to an Autonomous Checkout Robot. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2837–2844, 2011a.

Ellen Klingbeil, Deepak Drao, Blake Carpenter, Varun Ganapathi, Oussama Khatib, and Andrew Y. Ng. Grasping with Application to an Autonomous Checkout Robot. In *In International Conference on Robotics and Automation (ICRA)*, 2011b.

N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 3:2149–2154 vol.3, 2004.

David Kortenkamp and Terry Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, AAAI'94, pages 979–984, 1994.

Lars Kunze, Moritz Tenorth, and Michael Beetz. Putting People's Common Sense into Knowledge Bases of Household Robots. In *33rd Annual German Conference on Artificial Intelligence (KI 2010)*, pages 151–159. Springer, 2010.

K Lai and D Fox. Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation. *The International Journal of Robotics Research*, 29(8):1019–1037, 2010.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *International Conference on Robotics and Automation (ICRA)*, 2011a.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse Distance Learning for Object Recognition Combining RGB and Depth Information. In *IEEE International Conference on on Robotics and Automation*, 2011b.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, 2002.

libfastsift. Fast SIFT Image Features Library, 2009.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987.

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. Journal of Comp. Vision*, 60(2):91–110, 2004.

Ilya Lysenkov, Victor Eruhimov, and Gary Bradski. Recognition and Pose Estimation of Rigid Transparent Objects with a Kinect Sensor. In *Proc. of Robotics: Science and Systems*, 2012.

Tomasz Malisiewicz and Alexei A. Efros. Improving Spatial Support for Objects via Multiple Segmentations. In *Proceedings of the British Machine Vision Conference*, 2007.

D. Marr and E. Hildreth. Theory of Edge Detection. In *Proceedings of the Royal Society of London. Series B, Biological Sciences*, volume 207, pages 187–217, 1980.

Zoltan-Csaba Marton, Ferenc Balint-Benczedi, Nico Blodow, Lucian Cosmin Goron, and Michael Beetz. Object Categorization in Clutter using Additive Features and Hashing of Part-graph Descriptors. In *Proceedings of Spatial Cognition (SC)*, 2012.

Zoltan Csaba Marton, Dejan Pangercic, Nico Blodow, and Michael Beetz. Combined 2D-3D Categorization and Classification for Multimodal Perception Systems. *The International Journal of Robotics Research*, 30(11):1378–1402, 2011.

Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3D Modelling of Novel Objects from a Single View. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

Robert Mateescu and Rina Dechter. Mixed Deterministic and Probabilistic Networks. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):3–51, 2008.

Ajay K. Mishra and Yiannis Aloimonos. Active Segmentation With Fixation. In *ICCV*, 2009.

Tracy L. Mitzner, Cory-Ann Smarr, Jenay M. Beer, Tiffany L. Chen, Jennifer Megan Springman, Akanksha Prakash, Charles C. Kemp, and Wendy A. Rogers. Older Adults' Acceptance of Assistive Robots for the Home. Technical Report HFA-TR-1105, Georgia Institute of Technology: Human Factors and Aging Laboratory, 2011.

J. Montiel, J. Civera, and A. Davison. Unified Inverse Depth Parametrization for Monocular SLAM. In *Proceedings of Robotics: Science and Systems*, 2006.

Oscar Martinez Mozos, Zoltan Csaba Marton, and Michael Beetz. Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans. *Robotics & Automation Magazine*, 18(2):22–32, 2011.

Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. AI Goggles: Real-time Description and Retrieval in the Real World with Online Learning. In *Proceedings of the 2009 Canadian Conference on Computer and Robot Vision*, 2009a.

Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. AI Goggles: Real-time Description and Retrieval in the Real World with Online Learning. *Computer and Robot Vision, Canadian Conference*, 0:184–191, 2009b.

N. J. Nilsson. Shakey the Robot. Technical Report 323, AI Center, SRI International, Menlo Park, CA, USA, 1984.

David Nister and Henrik Stewenius. Scalable Recognition with a Vocabulary Tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168. IEEE Computer Society, 2006.

Dejan Pangercic, Vladimir Haltakov, and Michael Beetz. Fast and Robust Object Detection in Household Environments Using Vocabulary Trees with SIFT Descriptors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, 2011a.

Dejan Pangercic, Koppany Mathe, Zoltan-Csaba Marton, Lucian Cosmin Goron, Monica-Simona Opris, Martin Schuster, Moritz Tenorth, Dominik Jain, Thomas Ruehr, and Michael Beetz. A Robot that Shops for and Stores Groceries. AAAI Video Competition (AIVC 2011), 2011b.

Dejan Pangercic, Rok Tavcar, Moritz Tenorth, and Michael Beetz. Visual Scene Detection and Interpretation using Encyclopedic Knowledge and Formal Description Logic. In *Proceedings of the International Conference on Advanced Robotics (ICAR).*, 2009.

Dejan Pangercic, Moritz Tenorth, Dominik Jain, and Michael Beetz. Combining Perception and Knowledge Processing for Everyday Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1065–1071, 2010.

Dejan Pangercic, Moritz Tenorth, Benjamin Pitzer, and Michael Beetz. Semantic Object Maps for Robotic Housework - Representation, Acquisition and Use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Patrick Pfaff, Wolfram Burgard, and Dieter Fox. Robust Monte-Carlo Localization Using Adaptive Likelihood Models. In *EUROS*, pages 181–194, 2006.

B. Pitzer, S. Kammel, C. DuHadway, and J. Becker. Automatic reconstruction of textured 3D models. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3486–3493, 2010.

Benjamin Pitzer, Michael Styer, Christian Bersch, Charles DuHadway, and Jan Becker. Towards perceptual shared autonomy for robotic mobile manipulation. In *ICRA*, pages 6245–6251, 2011.

PR2. Personal Robot 2, 2010.

Erwin Prassler, Herman Bruyninckx, Klas Nilsson, and Azamat Shakhimardanov. The Use of Reuse for Designing and Manufacturing Robots. Technical report, White Paper, 2009.

Mario Prats. *Robot Physical Interaction through the combination of Vision, Tactile and Force Feedback bibtex*. Phd thesis, Jaume-I University, Munich, Germany, 2009.

Andrzej Pronobis. *Semantic Mapping with Mobile Robots*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2011.

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA*

Workshop on Open Source Software, 2009.

Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation*, 60:2004, 2004.

J.C. Rocholl, S. Klenk, and G. Heidemann. Robust 1D Barcode Recognition on Mobile Devices. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2712 –2715, 2010.

Joe Romano. PR2 Tactile Sensor, 2011.

Thomas Rühr, Jürgen Sturm, Dejan Pangercic, Michael Beetz, and Daniel Cremers. A Generalized Framework for Opening Doors and Drawers in Kitchen Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael 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, 2008.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Andreas Holzbach, and Michael Beetz. Model-based and Learned Semantic Object Labeling in 3D Point Cloud Maps of Kitchen Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009a.

Radu Bogdan Rusu, Wim Meeussen, Sachin Chitta, and Michael Beetz. Laser-based Perception for Door and Handle Identification. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, 2009b. Best Paper Award.

Radu Bogdan Rusu, Aravind Sundaresan, Benoit Morisset, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe, and Michael Beetz. Leaving Flatland: Efficient Real-Time 3D Navigation. *Journal of Field Robotics (JFR)*, 2009c.

Manabu Saito, Haseru Chen, Kei Okada, Masayuki Inaba, Lars Kunze, and Michael Beetz. Semantic Object Search in Large-scale Indoor Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, 2011.

Martin Schuster, Dominik Jain, Moritz Tenorth, and Michael Beetz. Learning Organizational Principles in Human Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3867–3874, 2012.

Shaojie Shen, Nathan Michael, and Vijay Kumar. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In *ICRA*, pages 20–25, 2011.

Jianbo Shi and Carlo Tomasi. Good Features to Track. In *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, 2003.

C. M. Smith, J. J. Leonard, A. A. Bennett, and C. Shaw. Feature-based concurrent mapping and localization for autonomous underwater vehicles. In *Proceedings of IEEE Oceans*, 1997.

SoftKinetic. Depthsense Cameras, 2012.

K. H. Strobl, E. Mair, T. Bodenmüller, S. Kielhöfer, W. Sepp, M. Suppa, D. Burschka, and G. Hirzinger. The Self-Referenced DLR 3D-Modeler. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 21–28, 2009. best paper finalist.

J. Sturm, K. Konolige, C. Stachniss, and W. Burgard. 3D Pose Estimation, Tracking and Model Learning of Articulated Objects from Dense Depth Video using Projected Texture Stereo. In *Advanced Reasoning with Depth Cameras at the Robotics: (RSS10)*, 2010.

J. Sturm, C. Stachniss, and W. Burgard. A Probabilistic Framework for Learning Kinematic Models of Articulated Objects. *Journal on Artificial Intelligence Research (JAIR)*, 41:477–626, 2011.

Ioan A. Sucan and Sachin Chitta. MoveIt, 2013.

Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

Moritz Tenorth. *Knowledge Processing for Autonomous Robots*. PhD thesis, Technische Universität München, 2011.

Moritz Tenorth and Michael Beetz. KnowRob – Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, 2009.

Moritz Tenorth, Ulrich Klank, Dejan Pangercic, and Michael Beetz. Web-enabled Robots – Robots that Use the Web as an Information Resource. *Robotics & Automation Magazine*, 18 (2):58–68, 2011.

Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz. KNOWROB-MAP – Knowledge-Linked Semantic Object Maps. In *10th IEEE-RAS International Conference on Humanoid Robots*, pages 430–435, 2010.

S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map Learning and High-Speed Navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

Sebastian Thrun. Robotic Mapping: A Survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

Rudolph Triebel, Jiwon Shin, and Roland Siegwart. Segmentation and Unsupervised Part-based Discovery of Repetitive Objects. In *Proceedings of Robotics: Science and Systems*, 2010.

Markus Ulrich, Christian Wiedemann, and Carsten Steger. CAD-Based Recognition of 3D Objects in Monocular Images. In *IEEE International Conference on Robotics and Automation*, pages 1191–1198, 2009.

UnboundedRobotics. UBR-1 Robot, 2013.

René Vidal, Yi Ma, and Stefano Soatto. Two-view multibody structure from motion. *International Journal of Computer Vision*, 68:2006, 2004.

S. Wachenfeld, S. Terlunen, and Xiaoyi Jiang. Robust recognition of 1-D barcodes using camera phones. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, 2008.

Alastair J. Walker. An Efficient Method for Generating Discrete Random Variables with General Distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, 1977.

T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially Extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, pages 67–96, 2012.

WillowGarage. Precision of the Kinect sensor, 2010a.

WillowGarage. *Technical description of Kinect calibration*, 2010b.

Thomas Witzig, J. Marius Zöllner, Dejan Pangercic, Sarah Osentoski, Philip Roan, Rainer Jäkel, and Rüdiger Dillmann. Context Aware Shared Autonomy for Robotic Manipulation Tasks. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo Big Sight, Japan*, 2013.

K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010. Software available at http://octomap.sf.net/.

Xuehan Xiong, Daniel Munoz, J. Andrew (Drew) Bagnell, and Martial Hebert. 3-D Scene Analysis via Sequenced Predictions over Points and Regions. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

Heran Yang, Tiffany Low, Matthew Cong, and Ashutosh Saxena. Inferring 3D Articulated Models for Box Packaging Robot. *CoRR*, abs/1106.4632, 2011.

Qingming Zhan, Yubin Liang, and Yinghui Xiao. Color-Based Segmentation of Point Clouds. 2009.