TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik IX

Intelligente Autonome Systeme

Multi-cue Perception for Robotic Object Manipulation

How Spatio-temporal Integration of Multi-modal Information
Aids Task Execution

Zoltán-Csaba Márton

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. N. Navab
Prüfer der Dissertation:	1.	Univ.-Prof. M. Beetz, Ph.D.
		Universität Bremen
	2.	Univ.-Prof. Dr. D. Burschka

Die Dissertation wurde am 08.07.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 05.03.2014 angenommen.

to Kinga and Ilka

All men by nature are actuated with the desire of knowledge, and an indication of this is the love of the senses. For even, irrespective of their utility, are they loved for their own sakes. And preeminently above the rest, the sense of sight. For not only for practical purposes, but also when not intent on doing anything, we choose the power of vision in preference (so to say) to all the rest of the senses. And a cause of this is the following, that this one of the senses particularly enables us to apprehend whatever knowledge it is the inlet of, and that it makes many distinctive qualities manifest.

ARISTOTLE, METAPHYSICS

Abstract

The thesis presents a robotic perception system for manipulation tasks in human living environments, that eases the deployment of autonomous service robots in new environments by enabling it to autonomously learn, model and localize the objects it perceives through the integration of geometric categorization and instance recognition.

To do so, we designed, tested and analyzed novel methods for: (i) learning general object-related principles from the Internet based on their part composition, (ii) computing descriptors that facilitate generalization between instances of the same category, (iii) classification that is robust to large variations between training and testing data, (iv) and surface reconstruction capable of dealing with partial information and cluttered scenes, while respecting physical constraints. These enable robots to take advantage of their capability to explore their surroundings and that they are equipped with sensors with different advantages, by the integration of *prior*, *spatio-temporal* and *multi-cue* information.

The developed system and the underlying modules and principles were shown to outperform alternative solutions in extensive experiments, and were applied in real-world robotic demonstration scenarios, proving the advantage of an embodied, task-adapting approach towards perception.

Kurzfassung

Die vorliegende Dissertation repräsentiert ein Roboter-Wahrnehmungssystem für Häusliche Manipulationsaufgaben, das den Einsatz von autonomen Servicerobotern in unbekannten Umgebungen erleichtert, indem dem Roboter ermöglicht wird, wahrgenommene Objekte autonom durch die Integration von geometrischen Kategorisierungs- und Instanzerkennungs-Algorithmen zu lernen, zu modellieren und zu lokalisieren.

Methoden wurden konzipiert, analysiert und getestet, die: (i) generische, objektrelevante Prinzipien anhand deren Bestandteile über das Internet erlernen, (ii) Gemeinsamkeitsbeschreibungen zwischen Objekten der selben Kategorie errechnen, (iii) mit großer Robustheit gegenüber Varianz zwischen Trainings- und Verifikationsdaten klassifizieren, (iv) sowie Oberflächen unter Berücksichtigung von partiellen Informationen, überladenen Szenen und physikalischen Beschränkungen rekonstruieren. Diese Methoden lassen Roboter den Vorteil ausnutzen, dass sie ihre Umgebung durch die Integration von *vorausgehenden*, *räumlich-zeitlichen* und *multimodalen* Informationen unter Verwendung von Sensoren mit unterschiedlichen Vorteilen erkunden können.

Es wurde in umfangreichen Experimenten gezeigt, dass das entwickelte System sowie die zugrundeliegenden Komponenten und Prinzipien alternative Lösungen an Leistung übertreffen, und hat während der Anwendung in realweltlichen Roboter-Demonstrationsszenarios den Vorteil eines verkörperten, aufgabengerichteten Ansatzes zur Wahrnehmung aufgezeigt.

Acknowledgements

First and foremost, my thanks go to my supervisor, Prof. Michael Beetz, PhD. He always supported and inspired me, and it was a pleasure and honor to work with him. I hope we will continue collaborating and that the physical distance does not become a personal one. The same goes for my mentor, Dr. Radu Bogdan Rusu, who took me under his wing during my university studies and enabled me to write my Masters thesis at the Technische Universität München. I can safely say that his support while we were fellow PhD students was crucial for my work.

Over the years, I was closely working with the perception team of our chair, Nico Blodow, Dejan Pangercic and Mihai Dolha, who were an excellent team for addressing difficult problems together, and also the best office mates one could wish for. I would like to thank them for their help, and I will miss our office full of ideas and fun. Similarly, the whole Intelligent Autonomous Systems group provided an excellent environment, both professionally and personally. Here I would like to especially thank the people I work together most closely: Dominik Jain, Alexis Maldonado and Moritz Tenorth, but my gratitude goes to all of the group members over the years really.

I had the opportunity and pleasure to work with people outside our group as well, and I feel myself lucky to have met Dr. Oscar Martinez Mozos and Dr. Asako Kanezaki, who opened my mind to new ways of looking at problems, for which I am very grateful. Having some great students to supervise was also one of the best things that could happen, as I would learn just as much from them, as they did from me. I really see Dr. Lucian Cosmin Goron, Florian Seidel, Ferenc Balint-Benczedi and Darko Stanimirovic as my colleagues, and I thank them for their unstopping and contagious motivation.

Last but not least, I would also like to thank the people at ISPRS and Leica for giving us access to the Ljubljana street scans, the Stanford Computer Graphics Laboratory for releasing The Stanford 3D Scanning Repository, as well as Radiohead for publishing their urban scan data.

Contents

List of Resources

Figures

Tables

Chapter 1

Introduction

It may not come so far that the avalanche of knowledge transforms science into a specializational porridge, in that state about which an aphorism says that the future expert will know everything about nothing! The saving turn is brought about by a global knowledge system that is generally available – but not for living beings, because they can not cope with this burden.

STANISLAW LEM (OBSERVATION ON THE SPOT)

The ever increasing mechanization and automation of tedious or dangerous tasks increased our standard of living and created entirely new industries. Robotics, for example, gradually takes over hard physical labor and increases productivity, most visibly in factories, but in other domains as well. Some tasks, like the exploration of Mars would be unimaginable today without robots. However, when it comes to performing tasks where the steps can not be described easily in an algorithm that the robot "blindly" follows, we come to the realm of trying to reproduce some aspects of our brain, i.e. adaptability and problem solving in new situations. This is the case especially when a robot has to work in the presence of (and together with) humans, where it has to evaluate and react to large amounts of information. Clearly, a lot of common sense knowledge is required for the robot.

It is said that we live in an information age, with the Internet holding huge amount of knowledge readily available. Often we just use search engines and online encyclopedias to find the information we need efficiently and in a structured manner. However, for autonomous agents these data sources are far less intelligible than for us, as of now. There are efforts to structure the data in a more uniform, and machine read-able way, i.e. the Semantic Web effort[1], but the information is mostly unstructured. Similarly, the world as perceived by computers through different sensors is just raw data, that has to be analyzed by intelligent algorithms before it can be interpreted and interacted with.

Robots thus need to be able to make sense of what they perceive of the environment and of the online knowledge sources, that hold information about the world and object around them, in order to be able to act flexibly, safely and reliably. This is the large-scale scope of the work presented in this thesis, more specifically, the creation of a task adapting visual perception system enabling a household robotic system to understand and interact with its environment.

1.1 Motivation

Visual perception is arguably the most important senses we have for obtaining information about the world (Lynott and Connell, 2009), and it is also the most widely employed environment perception modality in robotics and autonomous systems, used for task as varied as localization, navigation, environment modeling, object recognition, human robot interaction and shared autonomy. While there were some advancements in understanding how human vision works, and some of its limitations can be highlighted with excellent optical illusions, there is no final, unified theory yet, and computer vision is struggling to reach the level of a two year old child (Szeliski, 2010).

Advancements in the field are, however, already being successfully employed for example in human tracking, character recognition, medical imaging and photometry, with a growing need for more advanced solutions. Autonomous systems are expected to make an impact not only on our economy[2] and production[3], but also through the

[1]http://www.w3.org/2001/sw/

[2]Frost & Sullivan's *Global Top 10 Hot Technologies to Invest* include autonomous systems and 3D integration: http://www.frost.com/prod/servlet/press-release.pag?docid=182463128

[3]German Ministry of Economy and Technology funding project *AUTONOMIK4.0* as part of the German Hightech-Strategy 2020: http://www.autonomik40.de/

field of service robotics[4][5][6], accompanied by a growing acceptance of service robotic solutions. According to a German study, the most pressing need in in the elderly care sector (Beske et al., 2007), where the deployment of service robots is supported by three quarter of the technical personnel, half of the healthcare professionals and even 56% of the retired persons (Meyer, 2011). However, there are many skeptics among the elderly especially, therefore equipping the autonomous agents of the future with perception modules, that enable their safe and reliable self-controlled operation, is a central challenge. This is also supported by various studies created for the German Ministry for Economy and Technology (Botthof et al., 2011; Künzel, 2012). These see the autonomous systems capability of acting self-sufficiently being realized through the use of environment models built using 3D and multi-modal (and multi-view) technologies, and exploiting the World Wide Web for obtaining information on understanding their environment. The integration of results from the cognitive sciences is also stressed, as task adaptation semantic reasoning methods, context-based knowledge integration is needed for the creation of flexible products. It is important to note that the integration of robots into an ecosystems of agents might be necessary for reaching such challenging goals as assistance systems for assembly, transportation, medicine, search and rescue, agriculture, and robotic butlers for assisted living. These also require software platforms and general, reusable and extensible modules, easily configurable or self-tuning, that enable scaling to systems of increasing complexity, as current solutions are quasi-handmade.

There are multiple research projects worldwide, at universities, research institutes and companies alike, whose aim is the development of service robots (both for personal and industrial applications). The vacuum cleaner robot Roomba and its competitors are perhaps the most widely known finished products, but besides iRobot and EvolutionRobotics, for example Rethink Robotics, PAL Robotics and Willow Garage are developing complex robots to tap this new market, and companies like BOSCH and KUKA are also moving in this direction. As the large European Robotics Technology Platform (EUROP) and the Coordination Action for Robotics in Europe (CARE) ana-

[4]Europe 2020 flagship initiative *Digital agenda for Europe*: http://ec.europa.eu/digital-agenda/en/science-and-technology/robotics

[5]The DARPA Robotics Challenge: http://www.theroboticschallenge.org/

[6]The AUTOMATICA Internation Trade Fair will have service robotics as new exhibition focal point starting from 2014.

lyze the future research directions[7], these get pushed forward by research institutes, mostly in the public sector, forming networks and collaborations (typically involving industrial partners) like for example in Germany the Cognition Interaction Technology (CITEC) and Cognition for Technical Systems (CoTeSys) clusters of excellence, and the German Service Robotics Initiative (DESIRE).

Such a large effort is needed, as the goal of Artificial Intelligence (AI) turned out to be more challenging as it was expected at the birth of the field, by scientists and the public as well. With all the effort and successes in the last decades, there are plenty of problems remaining for the years to come. Analogously, the task of sensing and interpretation of the environment and the objects in it was also underestimated at first (Boden, 2006). In (Künzel, 2012) the careful inching forward of current robots is contrasted with a person running through a crowd (without injuring anyone), highlighting not only the agility of humans, but also the understanding of the dynamics of the surrounding that is unmatched by robotic perception. Perception, however, is identified as one of the key challenges that need to be solved generally, for a multitude of research endeavors. In contrast to perception in industrial settings, one of the impediments is, as described by Szeliski (2010), that vision (in an unknown and "uncooperating" environment) is an inverse problem, where the perceived image needs to be understood without sufficient information. This is especially true in 2D (see Figure 1.1 for a comic rendering of the problem), and while using 3D technologies helps to recover some of the information more easily, problems abound.

Autonomous robots that are to perform everyday manipulation tasks in human living environments must be able to pick up and place many objects of daily use. To manipulate them, the robots first have to perceive these objects, that is, they are to detect the objects in realistic settings, categorize them, recognize object instances, estimate their pose or even reconstruct their shape from partial views.

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 poses a very hard and long-term research problem. Robot manipulation tasks are usually restricted to detecting

[7]See the *Strategic Research Agenda for robotics* and its product visions at: http://www.robotics-platform.eu/sra

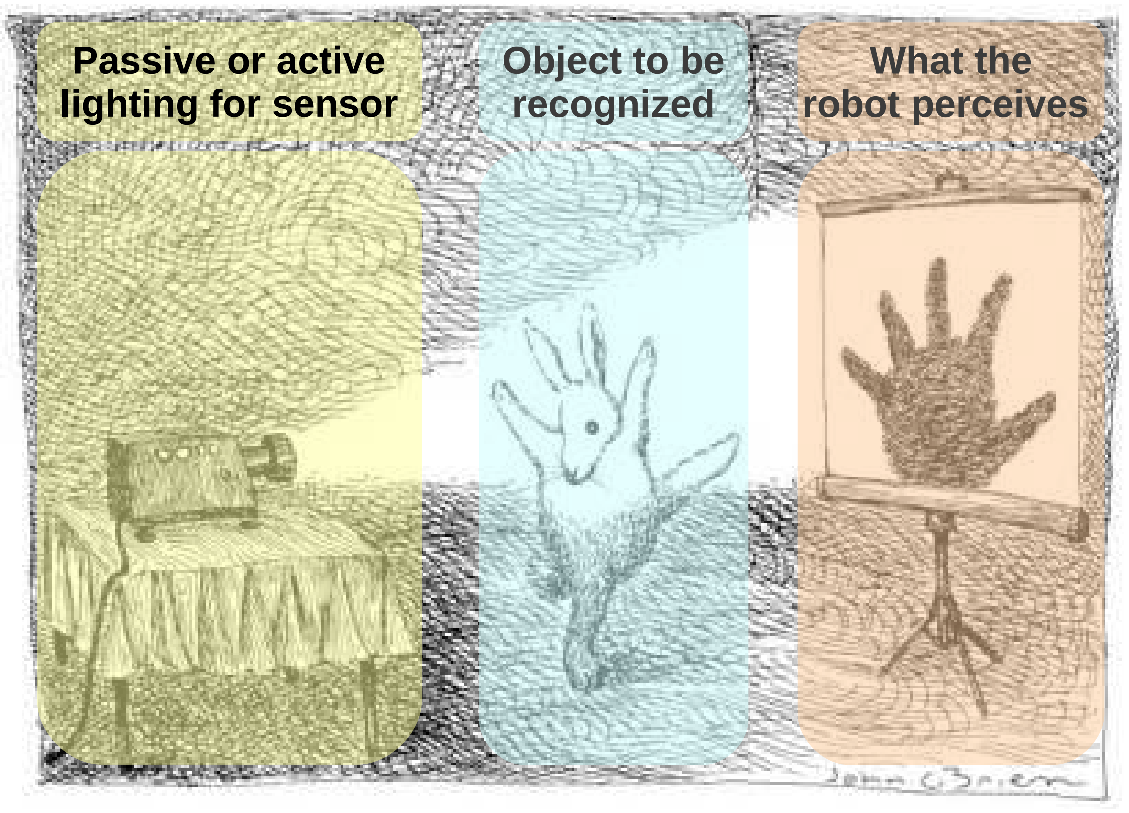


Figure 1.1: *Vision is an inverse problem. We, and robots, perceive only a projection of the world, but still have to create a complete model of it, sometimes leading to erroneous interpretations. (Underlying drawing taken from: © New Yorker Cartoon by John O'brien.)*

distinctively textured objects, objects with distinct colors or specific shapes, based on the full list of possible objects.

The current state of the art algorithms have 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 object poses, in the number of objects they can deal with at once, etc.

The perception tasks performed by most of these robots, however, are both too easy and too hard.

They are too easy because they often cannot perceive the objects to be manipulated with the accuracy and reliability needed for grasping them successfully in real everyday settings. The reasons are manifold. Lighting conditions might cause low contrast and specular reflections. Objects might be partly occluded. The objects themselves might require different perceptual methods. Some of them are best perceived by their visual appearance, such as cereal boxes, while others are better perceived by their shapes. Many objects are quite similar and require detailed-oriented differentiation methods.

They are too hard because they often do not exploit the regularities of their tasks, their environments and the objects they are to manipulate.

Human living environments and objects of daily use in them present present regularities that can be used for creating more robust detection routines. For example, most objects are located at specific locations, placed on horizontal surfaces in a physically stable position, and fall into few geometric categories, as a survey of publicly available household object databases shows (see Table 5.10).

There are methods that have been shown to be able to classify some of the objects that might appear in such settings. There are, however, inherent limitations in these approaches, and as described by Kragic and Vincze (2009), there is no robust and large-scale solution yet. However, object classification can be made significantly more robust if multiple sources of information are used, that capture different aspects of the objects and contextual information.

The focus of this work is on advancing closer to this goal by forming a multi-cue perception system capable of dealing with the above challenges and provide the required functionality both for high-level planning and manipulation. This is achieved by taking advantage of the robots' exploration capabilities and priors about possible object locations and types, the use of multiple sensory modalities, various detectors with different strengths and weaknesses, and the incorporation of multiple observations over time to obtain better results. The efficiently combination of these ideas is required for the realization of cognitive technical systems, as suggested by AI research and cognitive psychology.

1.2 Outline and Contributions

During my stay at the Intelligent Autonomous Systems (IAS) group at TUM, as part of CoTeSys, I had the opportunity to explore a large breadth of the difficult, but interesting field of cognitive perception, and there were important lessons to be learned and transfered between the various related domains. The result was a versatile perception system that is able to deal with different problem domains in a unified way. After giving an overview of the related work, I will present the outline of the system in Chapter 2, and describe the repertoire of general algorithms that will be employed throughout the thesis, along with the specific contributions I made to them to advance the state of the art. Integrating perception with a robot's control and planning systems raises the need for good communication interfaces, but it also offers the opportunity for embodied learning during long-term operation and multi-cue integration, as discussed in Chapter 3. In the remaining of the thesis I will then focus on how to use the system for addressing the challenging problem of object detection in cluttered scenes, first in the case of modeling the semi-static environment (Chapter 4), and then for object detection (Chapter 5). Finally, I will present application scenarios in integrated robotic systems, followed by the concluding remarks and notes on the publishing of the results in Chapter 6.

As the deployment environment of service robots is almost sure to differ from the one used in teaching it to perceive and understand it, the thesis focuses on finding the known category of previously unseen object instances, while incorporating traditional instance level algorithms as well. More specifically, the thesis contributes along the following dimensions:

- The perception system uses the Internet as a resource for training general detection methods, that are then applied in novel settings. Online object databases can be used to learn the general structure of the objects, and to detect these structures in the robot's deployment environment. This was shown to work even if the models used for learning differ significantly from the object instances found in the real environment.

- The incorporation of physical constraints into the detection of objects, in order to validate the object hypotheses, thus reducing false positives. This is achieved by combining bottom-up, data driven hypothesis generation modules with top-down, model driven verification, and knowledge about stability and volume. Additionally, in order to reconstruct the specific objects' 3D model for manipulation, geometric fitting methods were developed that are capable of dealing with partial information.

- The creation of local and global feature descriptors that facilitate generalization, and can therefore be applied in adapting to domain specific perception tasks. The accurate estimation of geometric structure proved to result in a low-dimensional feature descriptor that has a direct physical interpretation, and the object-level descriptor built out of it was shown to outperform abstract, high-dimensional features in the potential for generalization out of the training set.

- The introduction of graph-theoretic processing and classification strategy into clutter segmentation, that made efficient approximate subgraph matching possible. Detecting touching and partially visible objects is achieved by employing the developed feature descriptor, and exploiting its additive property for generating subgraph descriptors. The use of graph-based hashing then speeds up the generation of valid segmentation hypotheses.

- The combination of different sensing modalities, classification approaches, and views of the scene into an ensemble of experts, where the various members can complement and correct each other. It was found that by using components that already provide high accuracy multi-class results on their own, and by employing the right learning strategy, results can outperform traditional methods.

I validate the performance of the building blocks and the whole perception system by running it on the robot for solving difficult real world perceptual tasks. The developed recognition method learned object specific part-compositions from online databases and outperformed alternative solutions by over 17% (Table 5.9). It successfully segmented and reconstructed objects in cluttered scenes, with the designed geometric feature significantly surpassing RGB-based ones for the geometric categorization task (Tables 5.3, 5.4 and 5.8).

In the following I will introduce and elaborate on these methods, putting them into context.

In the early twentieth century, Gestalt psychology and its theory of perception came to be formulated, forming a collection of rules collectively called Gestalt laws or principles. These deal with properties that define objects, and how they can be separated from other objects, and are still very much in use in today's computer vision fields. As with many other theories of mind, it does not provide a final list of formulas to follow, rather it is continuously evolving. There have been many advances in theories of perception (Marr, 1982; Palmer, 1999) and cognition, with applications to the large goal of AI (Brooks, 1999). An ever recurring theme seems to be the necessity of bottom-up data driven hypothesis generation, verified by top-down concept driven models, both in human and computer vision (Frisby and Stone, 2010). The cognitivist and emergent philosophies disagree on how high level concepts should be represented/learned, but getting feedback from the outside world and reacting to it seems to be an important aspect of cognition (Vernon, 2005). At least that is how we can intuitively judge intelligence (noting however the issues brought up by Searle's Chinese room argument).

The idea that a cognitive agent needs to be embodied to learn based on gathered experiences has been around since Turing (Turing, 1970), who proposed equipping computers "with the best sense organs that money can buy" and teaching them in order to pass his famous test (Turing, 1950). Similarly, the authors in (Steels and Brooks, 1995) and Vernon (2005) argue for embodiment, and present different paradigms on how to approach the learning and grounding of new information. Practically, the creation of *weak AI* is enough for mastering many of the tasks we would like a robot to perform, but the flexibility provided by adaptability is indeed important, as preconditions, environments and objects in it are often changing. Related ideas are discussed by Horswill (1995), presenting the argument that robot's task and environment adaptation improves its capability to perceive objects.

Autonomous robots would need to detect a large variety of objects, in the range of several thousands, thus finding some common structure and categories is important (Biederman, 1987). Biederman brings up the example of lamps, but chairs would also be a good example, as illustrated by the appropriately named book: 1000 Chairs (Fiell, 2005). Other objects, both large and small, that could be relevant for a robotic household assistant have a large amount of instances, some of it being captured in the 500 Series of books published by Lark Crafts and Lark Books, for example: tables, cabinets, boxes, baskets, bowl, vases, plates and chargers, teapots, pitchers, cups, and of course chairs too. Since typical object recognition methods rely on a closed set of well defined object instances being provided a priori during training, the need for more general concepts for objects becomes apparent. In addition to the classification, pose and other grasp-relevant information is needed by the robot, thus exploiting every possible source of information will be necessary.

The use of multiple detectors is important, as real-world objects, especially products of the same company have very similar appearance (coffee, cans, chocolate, etc.) as illustrated in Figure 1.2 and discussed in (Marton et al., 2011). Using only geometric descriptors is not generally applicable either, as, for example, mugs, drinks and cereal boxes typically have similar shapes but different appearance. In both cases, the intended use of the object can be significantly different and can be relevant to the task at hand. Additionally, some objects might be hard to detect with one sensor, but more easily detectable using others (for example semi-transparent or shiny objects).

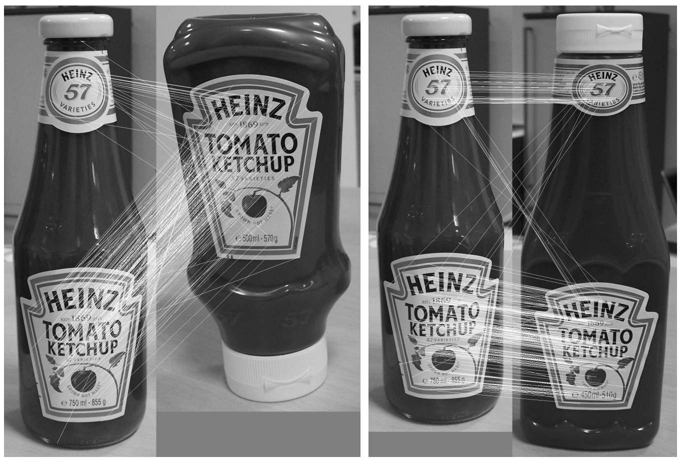


Figure 1.2: *Two examples of good model matches using local image features. The images however depict three ketchup bottles with three different shapes which makes classification of the objects based on visual appearance challenging.*

While recognition of specific (especially textured) objects is very advanced, as in the early days of computer vision, perceptual grouping category recognition approaches are becoming interesting again (Dickinson, 2009; Szeliski, 2010). This is because the visual interest-point based feature descriptor matching approaches for object instance recognition are less applicable for category recognition. Another beneficial development has been the availability of large amounts of richly annotated data on the Internet, facilitating advanced learning methods. These can be used for learning general principles on the part structure of object categories for example. Merging "partial" information (different sensors, features, viewpoints, part compositions) improves robustness, as does the task-adaptation of the different modules (specialized class hierarchies, priors on possible object types/locations coming from task specification or environment observation).

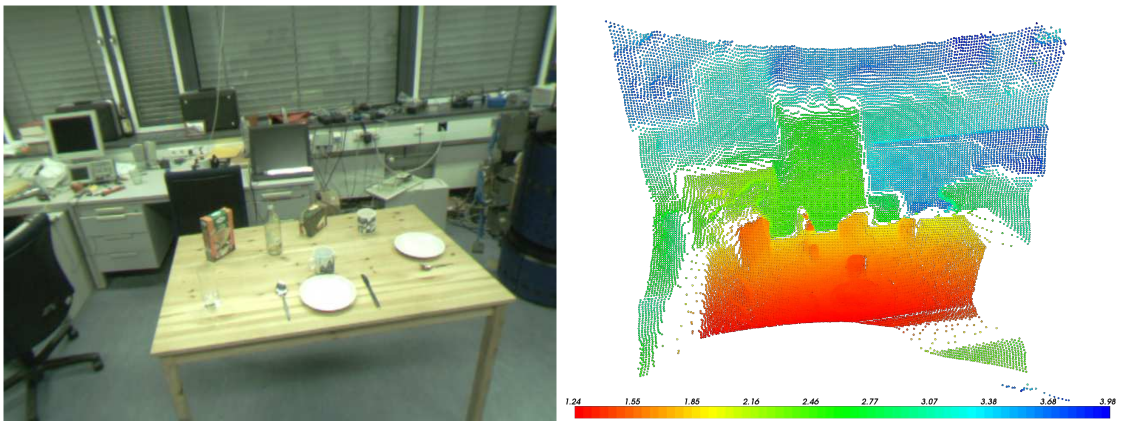
Such a system was developed in this thesis, as exemplified in Figure 1.3, where different part segmentations and various detectors running on complementary sensory data are combined to produce high quality recognition. In the figure the detections which are correct are marked with blue, while the ones that are incorrect with red (with the ground truth in parentheses). The detection of problematic objects like glasses and silverware is facilitated by the merging of information sources and capturing inter-part relations. Even though segmentation is not always correct (see the "ambiguous" ground truth label), when part combinations are taken into account by the method, there is a chance for recognizing one of the contained objects from the segment. In this example low quality 3D data was used, in which small boxes and mugs are hard to distinguish, but, as presented later, these principles were applied to high quality laser and structured light data as well.

1.3 Related Work

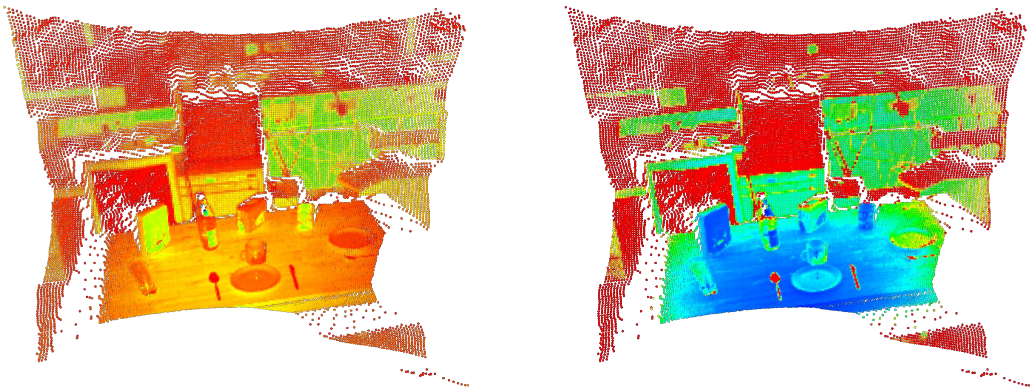
As we are always building on existing work, this section is an attempt to present a cross-section of the field, placing the presented work into context. In the following subsections, related approaches are discussed, grouped together according to the the main areas of research this work falls into.

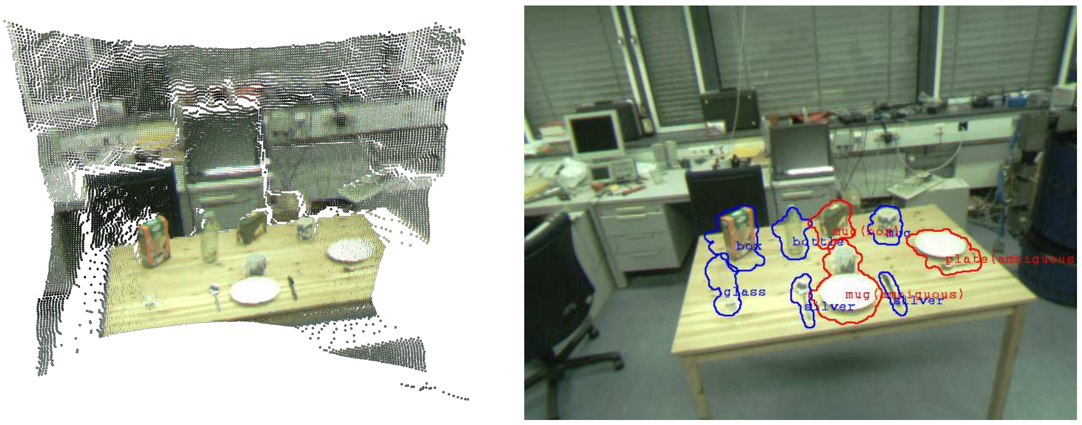
1.3.1 Surface Reconstruction

Most of the robotic navigation systems today rely on 2D discrete representations of the world, such as large scale 2D occupancy maps. for example (Vincent et al., 2008; Thrun et al., 2006; Konolige et al., 2006). This is in contrast with applications in computer graphics – like the reconstruction of architectural heritage (Levoy et al.,



(a) RGB camera image and corresponding depth measurements for a tabletop scene


(b) Additional measurements for the 3D points (infrared signal amplitude and confidence)



(c) Merged RGB and 3D data, and its use for part-based segmentation and categorization

Figure 1.3: *Multi-modal object detection using a Videre stereo-on-a-chip camera and a SwissRanger 4000 time-of-flight camera. Please note that due to the low quality 3D data, small boxes and mugs are hard to distinguish.*

2000) or 3D watertight objects (Gopi et al., 2000) – which build extremely accurate models whose appearance has to be as close as possible to the original surface, but at a much smaller scale. In contrast to well known surface reconstruction methods from computer graphics (Gopi and Krishnan, 2002; Scheidegger et al., 2005; Amenta et al., 2001; Dey and Goswami, 2003), robotics applications require methods that work reliably on noisy scans acquired by mobile robots navigating in indoor and outdoor environments.

Since a single scan usually can not cover the whole region of interest, multiple scans need to be registered and merged, typically with a variant of the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992; Zhang, 1994). In order to autonomously acquire the different views of scenes and objects, Next Best View (NBV) algorithms are required. This has been a long-explored, and still active field of research, due its computational complexity (Bajcsy, 1988; Pito, 1999; Reed and Allen, 2000; Callieri et al., 2004; Suppa, 2008; Potthast and Sukhatme, 2011; Kriegel et al., 2012). Another approach to compute the ego-motion and combine registered views into maps is to use a combination of multiple sensor, like a laser, a camera, and a stereo pair, aided by an inertial measurement unit (IMU) as by Mirisola et al. (2007) for example.

Hahne and Alexa (2008); Zhu et al. (2008) present two methods for improving the depth image of time-of-flight cameras based on a stereo camera setup and graph-cut and probabilistic algorithms. The cameras used are a 19k type PDM and a SwissRanger 3000. While these methods give promising results, the increased acquisition time for a slight improvement in accuracy is a downside for applications such as ours. A similar approach is presented in (Diebel and Thrun, 2005) but no information is given by the authors on the computational performance of the algorithm.

Once a complete point cloud model is obtained, points from the different scans can be used to reconstruct the underlying surface. This step can reduce the amount of noise, and even out registration mistakes in order to aid further processing of the data. The Moving Least Squares (MLS) algorithm is a widely used technique for approximating scattered data using smooth functions. Alexa et al. (2003) describe a method to fit polynomials to a point cloud based only on coordinate data and estimated surface normals. To counteract the effects of noise, our approach uses Sample Consensus like methods and other robust estimators in order to calculate the best reference plane before fitting a high order polynomial (Rusu et al., 2007). Another Robust Moving

Least Squares algorithm is described in (Fleishman et al., 2005), which differs from ours in that it uses LMedS instead of our sample consensus (SAC) based model fitting approach. To obtain a polygonal model of the world, further surface reconstruction methods can be applied such as (Hoppe et al., 1992; Gopi et al., 2000).

1.3.2 Object Model Fitting for Grasping

In case a surface model is already available for the object, and it just has to be attached to the current sensor readings, the process is analogous to robot localization. There as well a model exists, i.e. the map, and the relative transformation between the robot and the map is computed based on the sensor readings of the environment. In case there is an initial guess of this transformation, the model and sensor data just has to be locally registered by minimizing a distance-based cost function. Such methods have been widely explored in the Simultaneous Localization And Mapping (SLAM) field, a short overview of which is given by Sturm et al. (2012). However, for fitting object models, there is typically no such approximate solution, and global registration has to be performed, bearing close resemblance to the kidnapped robot problem.

Multiple sensors were used for solving such tasks, like cameras (Ulrich et al., 2009a; Coates et al., 2009), stereo cameras (Fritz et al., 2009; Hillenbrand, 2008), 3D sensors (Steder et al., 2009), and also their combinations to speed up or improve results (Klank et al., 2009a).

A vision-based grasping system which segments objects on a table and constructs triangular meshes for them is presented in (Richtsfeld and Vincze, 2008). While the presented method is general and works for many objects, it creates complicated models for certain objects, which could be simplified through the usage of geometric primitives. A simplification of the modeling problem is used in Grasp-It (Miller and Allen, 2004), where combinations of geometric shape primitives such as spheres, cylinders, cones and boxes are used to model each object. Bley et al. (2006) introduced "geons" (geometric icons), that can be used to develop generic category descriptions using geometric primitives, but also additional category knowledge, for the purpose of building libraries of grasp models for a variety of objects.

A computer vision and machine learning based method is used in (Saxena et al., 2008) to train classifiers that can predict the grasping points in an image. This is then applied to images of unseen objects. To obtain 3D positions of grasping points, the authors use stereo cameras, but their approach works reliably only to the extent provided by the training data. Another issue is the segmentation of objects, since grasp points are provided with no information about what objects are in the scene and to which of them do the identified points correspond. In (Bone et al., 2008) an accurate line laser and a camera is used to build models and identify grasping points for novel objects with very encouraging results. However the system was tested only on two objects, thus its scalability is not clear.

Recently, Richtsfeld et al. (2012) presented a multi-level approach to fit planar or curved surfaces to over-segment parts, and then define inter-segment relations to decide if they should be merged or not. Unlike our approach, they consider relations between non-touching parts as well, but the method performs best for merging touching segments and for convex shapes. Other approaches also focus on creating surface models by fitting shape primitives or superquadrics and considering the spacial relations between them (Georg Biegelbauer, 2007; Li et al., 2011; Nieuwenhuisen et al., 2013), but in slightly simpler scenarios.

Available models of complex objects are decomposed into superquadric parts by Biegelbauer and Vincze (2007) and Zhang et al. (2004), and these models are matched to a point cloud. This needs a database of models, and moreover, their decomposition into superquadric components, which is often difficult to obtain. A random sample consensus (RANSAC) based approach for model decomposition is presented by Schnabel et al. (2007), where a set of 3D geometric primitives (planes, spheres, cylinders, cones and tori) are fit to noisy point clouds. Since the point clouds presented there are complete, the authors do not need to reconstruct the missing parts.

Thrun and Wegbreit (2005) describe a method for detecting and verifying symmetries in point clouds obtained from a single viewpoint, and they project the existing points according to the detected symmetry to obtain the back side. However, using our methods we were able to reconstruct surfaces by approximating them with shape equations and thus generate a complete model, which can be meshed with the required density or sparseness according to the speed and accuracy requirements of our applications.

In the work of Xue et al. (2009) the grasping of objects modeled in the 3D object modeling center (KIT Object Models Web Database) was presented. The center employs a digitizer, a turntable and a pair of RGB cameras mounted to a rotating bracket which allows for views from above the scene. At the time being, there are around 110 highly detailed, high-precision objects publicly available. While working with such a system and data would yield high quality results, its downside lies in the fact that the modeling center is rather expensive and cannot be used online, i.e. mounted on a mobile robot for autonomous mapping. In another initiative, a database (Columbia Grasp Database) has been built (Goldfeder et al., 2009). The major difference between this work and ours lies in how the models are obtained. The authors created artificial 3D models whereas we acquired our models by scanning real world objects and surfaces, and are thus facing the problem of noisy and cluttered data.

The method by Romea et al. (2009) estimates 6 degrees of freedom (DOF) object poses in cluttered scenes by matching local descriptors to stored models. Since the objects present in household environments are often texture-less, our approach constitutes an important advantage over the above proposed research initiatives, which fail to work in the absence of well textured objects.

A geometric template (i.e. CAD-like model) matching method is presented by Drost et al. (2010), where candidate poses are estimated based on local 3D features. A similar method that uses camera images as input is described by Ulrich et al. (2009b) and Chen et al. (2010) for example. However, handling very large number of objects with such methods is very time-consuming.

1.3.3 Object Classification

There are two principal mainstream lines in the area of the object recognition related research: one aiming at recognition of objects in camera images, and one using 3D depth data acquired through range scanning devices. While there has been much valuable research in image-based techniques, e.g. by Savarese and Fei-Fei (2010), combining these with depth or 3D cues improves their performance.

Depending on the type of perception data, various different 2D distinctive local features have been developed. The SIFT descriptor (Lowe, 2004), which is one of the best-known keypoint descriptors and detectors, makes use of detected keypoints in images and compares them with referenced object pictures in order to identify the objects currently being observed. In the 3D domain, the VFH (Rusu et al., 2010) descriptor was recently developed as the latest, viewpoint dependent, extension of of PFH (Rusu et al., 2008a). The feature's discriminative power is increased by the inclusion of the viewpoint, which, however, also represents a deficiency in that the feature becomes orientation variant.

Taken individually however, these two separate approaches are still insufficient to solve the full object recognition problem as both are prone to failure in situation where texture-less objects are present or depth data is too noisy or ambiguous. Therefore, different research initiatives have decided to combine sets of local features and cluster them together using different metrics (kernels) in order to be able to infer the global identifiers for objects. This was especially aided by the development of structured light sensors that provide depth images combined with traditional RGB ones (RGBD), like the PrimeSense Carmine, Microsoft Kinect and Asus Xtion.

Lai et al. (2011b) validate the use of different visual modalities, using spin images (Johnson, 1997) for describing shape, and SIFT and texton histogram features to capture the visual appearance. They showed that color-based cues are more important for instance recognition, while geometric ones are better suited for categorization, and that their combination improves on both. This finding is also supported by Sun et al. (2010).

Inspired by earlier work based on developmental psychology (Griffith et al., 2009), object categorization using multiple modalities is explored by Sinapov and Stoytchev (2011). While the authors base their approach on psychological findings that suggest that a single sensory modality is often not enough, they leave out the most descriptive modality, vision (Lynott and Connell, 2009), and focus on proprioceptive and auditory feedback.

The combination of depth information with camera images is addressed by Quigley et al. (2009). The authors calculate depth information for each pixel in the scene by applying laser-line triangulation with a rotating vertical laser and a camera. To obtain high resolution 3D images, each scan requires 6 seconds with an additional 4 seconds spent on post-processing and triangulation. Thus a waiting period of 10 seconds has to be expected before object detection and robot manipulation could be performed.

Sun et al. (2010) presented a method inspired by Hough voting, and take advantage of both image and depth information from a single view. They evaluated detection and pose estimation separately, on a dataset containing 30 objects, identifying 16 distinct poses for each.

There are several other features provided in publicly available libraries like the Point Cloud Library (PCL) (Rusu and Cousins, 2011), for example the global feature GFPFH (Rusu et al., 2009c), intensity spin and RIFT (Lazebnik et al., 2005) which combine 3D and texture, PFH-RGB that extends PFH with color, and NARF (Steder et al., 2010) which was developed for range images. Various other global (e.g. BOB by Payet and Todorovic (2011)) or local (e.g. 3D SURF by Knopp et al. (2010)) features have been described, and there are also approaches which combine separately developed geometry and color descriptors into a single feature. However, properly balancing these two different properties is difficult, as discussed in (Kanezaki et al., 2011c). The rapid growth of the perception field means that comparing, evaluating, and combining the available approaches becomes increasingly relevant.

1.3.4 Ensemble Methods and Incremental Learning

While single classifiers, or sequential applications of them are what was typically used for object recognition so far, a combination of different classifiers and features using ensemble learning seems like a promising approach as it has been successfully applied to challenging machine learning problems in other domains.

Jain et al. (2000) provides a taxonomy for ensemble methods. They categorize methods based on:

- the architecture type (which can be parallel, cascading, gated, hierarchical or a combination of these)
- whether or not the ensemble is trainable

- according to the level of information the members of the ensemble produce about their classification decisions
- whether it is adaptive, which means that the ensemble weights the contribution of a member to the decision based on the input pattern

Sewell (2007) gives an overview of the literature on ensemble methods and also some practical advice. He concludes that there is no benefit from combining different classifiers over the same feature set and there is substantial benefit from combining the results of one classifier across different features sets, suggesting the applicability of ensemble learning to multi-modal perception using a single classifier per feature.

Lam and Suen (1995) investigated how to optimally combine classifiers for character recognition and found that simple voting is often the most robust choice. They found, that the combination methods which are trainable exhibited better performance than voting on the partition of the dataset on which they were trained, but performed worse on a second partition of the dataset.

An approach which has been particularly successful in the Netflix contest (Sill et al., 2009) is stacking, introduced by Wolpert (1992). In stacking the outputs of so called level-0 classifiers and also meta-data is combined to form the input for classifiers on level-1 which are trained to improve generalization accuracy on a validation set.

Madry et al. (2011) evaluated several linear and non-linear methods for combining classifiers trained on shape and appearance features. They used SIFT (on grayscale and the opponent color channels) and histograms of oriented gradients (HOG) for 2D appearance and 2D shape, together with the Fast PFH version for 3D shape. As base classifiers they used (multi-class) support vector machines (SVM) with χ^2 kernel. The ensemble methods evaluated were the max confidence rule, the product rule using confidences and the confidence weighted voting rule. They also tested stacking using SVM with the RBF kernel, the histogram intersection kernel and the χ^2 kernel trained on the confidences of the base classifiers. On a dataset of 11 object categories (10 objects each, 16 views per object), they found that the voting rule outperforms the other rules as well as the SVM.

Incremental learning and recognition of objects is done in an unsupervised manner by Triebel et al. (2010), but the authors focus mainly on furniture pieces, and it is not clear how well multiple objects could be reliably detected without any prior information. Our approach is a semi-supervised one, where the robot generates object hypotheses on its own, but the categorization happens into user-defined classes.

Hager and Wegbreit (2011) present a system that recognizes simple shapes based on geometry in clutter, on the assumption that the scene changes incrementally over time. Here we take a slightly different approach in that we are learning new views of more varied object autonomously and build up a database of the actual object instances in our environment.

A similar approach to ours is taken by (Hinterstoisser et al., 2010), where a new view of an object is added to the model if it is so different from the stored ones as to cause the matching score to drop below the detection level. As we are combining the image data with 3D information, we effectively eliminate the problem of scale and have a better segmentation of the objects, and therefore require fewer views to be learned.

1.3.5 Segmentation and Part-Based Recognition

Many of the methods described above rely on a correct segmentation of object hypotheses, from which they can compute features to be used for recognizing them. This is especially the case for global 3D descriptors like VFH. However, correctly segmenting objects in not always possible, especially when they are in cluttered scenes. A solution to this problem is part-based detection, which relies only on the identification of simple parts, that are typically easier to segment than complete objects.

Additionally, in order to match the perception capabilities of humans, Dickinson (2009) advocates that searching for predefined templates is not enough, because recognition of new exemplars of known categories have to be facilitated. Therefore the recognition and grouping of basic shapes is advocated, an aggregation process called *perceptual grouping*. As argued for example by Huber et al. (2004), part-based detection has the advantage of generalizing to unknown instances of object types. While in (Huber et al., 2004), (Marton et al., 2009c) and for the part-based VFH version called CVFH (Aldoma et al., 2011), a single object is required in the query, approaches like (Lai and Fox, 2010; Mozos et al., 2011) can efficiently detect objects in clutter.

This typically requires over-segmenting the the scene, possibly multiple times, such that object boundaries are respected, but an object can be split into multiple parts. Therefore, a correct and fully reproducible segmentation is not needed, thus simpler segmentation methods can be employed, usually based on detecting properties like concavity, that are known to delimit objects (Jacobs, 2001; Singh and Hoffman, 2001). This approach was employed for example in (Kanezaki et al., 2011b) in a multiple instance learning framework, using the part grouping method presented in Chapter 5.

While there are many promising advances in correctly segmenting objects, with and without some human aid, like (Bergström et al., 2011; Mishra and Aloimonos, 2011; Fowlkes et al., 2007; Comaniciu et al., 2002), as pointed out by Malisiewicz and Efros (2007), considering multiple over-segmented views is preferable to relying on a single, possibly erroneous, segmentation. There are several other approaches to clutter segmentation, but with limited exploration of the object recognition problem, like (Somanath et al., 2009; Schuster et al., 2010; Goron et al., 2010).

Recent approaches to image segmentation and annotation are presented by Socher et al. (2011) and shown to be less effective than the part-based approach proposed by the authors. After over-segmenting the image they create a binary tree representation by recursive merging nodes and assigning semantic features to them. The authors apply the same method to parsing natural language as well. While the approach looks extremely promising, the merges are not guaranteed to be performed first inside a single object in the case of multiple objects of the same type being next to each other. Instead of a fixed grouping, we enumerate all the possible subgraphs a node can be part of, and classify each of them separately. A second voting step to obtain object assignment and model fitting can then be easily applied, as detailed in (Mozos et al., 2011).

Fergus et al. (2003) model objects as flexible constellations of parts in an unsupervised scale-invariant learning approach for detecting objects in a wide range of images. While our method is somewhat different, it can be viewed as the 3D application of the presented principles, with the notable difference that we use additive features and consider part combinations, aided by hashing of graph features that capture spacial topology.

Shotton et al. (2007) incorporate poses and viewpoints, texture, layout, and context information for image segmentation based object recognition. In a complementing publication (Shotton et al., 2008) they address the problem of categorical objects recognition and localization in space and scale using a sliding window classifier. The method is still image based, but in its formulation and its use of geometry related image features is similar to 3D approaches.

In the 3D domain, a related idea was explored by Mian et al. (2006), but with a hash table built on local 3D features for CAD fitting, while we have feature independent hashing. Another approach by Ruiz-Correa et al. (2003) uses abstract shape class representations that capture the spatial relationships between the object parts. While the method uses point signatures, we compute descriptors for complete parts, thus can take advantage of more descriptive features.

1.3.6 Perception Systems for Object Recognition

Existing classification systems like the Shark Machine Learning Library (shark-project.sourceforge.new) or Shogun (www.shogun-toolbox.org) were not designed to be used by robots directly, and they would need to be wrapped by some middle-ware in order to use them during a robot's operation. Such a wrapping can be provided through the communication interface of the Robot Operating System (ROS, http://www.ros.org), since it is portable onto different systems, is distributed and supports multiple programming languages. Much of the required functionality is already available in image and 3D data processing libraries like OpenCV (Bradski and Pisarevsky, 2000), PCL, and the STAIR Vision Library (Gould et al., 2010).

Nakayama et al. (2009) present the AI Goggles system , which is a wearable system capable of describing generic objects in the environment and retrieving the memories of them 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, however the training and testing phases are separate. 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 to learn the latent variables. In this work, we present a method for combining multiple features, and discuss how unknown objects can be detected and learned automatically.

Another system optimized for speed is MOPED (Torres et al., 2010). It builds on POSESEQ, a state of the art object recognition algorithm and demonstrates a massive improvement in scalability and latency without sacrificing robustness. The authors achieve this with both algorithmic (different variations of nearest neighbor search) and architectural (SIMD instructions, etc) improvements, with a novel feature matching algorithm, a hybrid GPU/CPU architecture that exploits parallelism at all levels, and an optimized resource scheduler. Since the system considers visual only features, it has certain limitations that can be tackled by considering 3D data as well. Our system uses a different visual classifier, but in theory any method can be used. The main advantage of our approach is in the combination of image and 3D features, and the building of models for grasping and re-detection for new objects.

A system for 3D perception and modeling is presented in (Arbeiter et al., 2010), that 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 then serve as input for a modified fastSLAM algorithm for rendering environment maps or object models. While the 3D aspects are not given that much importance, dealing with repeating visual landmarks on an object is handled nicely by the system. As the focus is on determining correspondences between scans, recognizing when an object is novel and has to be learned is not addressed.

A self-referenced 3D modeler is presented in (Strobl et al., 2009), where the authors demonstrate that an ego-motion algorithm tracking natural, distinctive features and a concurrent 3D modeling of the scene is indeed possible. The use of stereo vision, an inertial measurement unit, and robust cost functions for pose estimation in the system further increase performance. While the authors build accurate models using a hand-held device, we are using the robot's own sensors and movements to not only model, but also recognize its environment and the objects it encounters.

Chapter 2

Analysis of Perception Modules

Essentially, all models are wrong, but some are useful.

George E. P. Box (Empirical Model-Building and Response Surfaces)

Every kind of ignorance in the world all results from not realizing that our perceptions are gambles. We believe what we see and then we believe our interpretation of it, we don't even know we are making an interpretation most of the time. We think this is reality.

Robert Anton Wilson

This chapter presents the necessary steps and methods needed for robotic perception, and how to best group them together to form an integrated and robust processing pipeline. For the comprehensive perception system described in this thesis, it is very important to define common processing steps and their possible interconnections, such that interfaces between the various modules may be fixed. As we will see in Chapter 5, the right interplay between these modules will enable a more powerful system than just the combination of the parts.

Apart from innovations in the individual steps, what sets the work in this chapter apart is the unique integration of different knowledge sources. This was possible at the Intelligent Autonomous Systems group, thanks to the collaboration of people with complementing competences. All in all, these methods are attempting to interpret the world and objects in it based on some assumptions, which might not always hold.

So they will be extended and combined in the remaining of the thesis to produce "gambles" that have higher chance of being close to reality.

It is generally true that many of the models, optimization criteria, reconstructions and features are not capturing the world perfectly. However, as George E. P. Box put it *"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."* Given knowledge about the specific application domain, improvements are always possible, but these always have to be interpreted in the light of the scenarios they are used in (Wagstaff, 2012). This chapter will focus on such an analysis and evaluation of the building blocks used in the later chapters.

2.1 The Perception Pipeline

A typical interconnection of various perception modules is depicted in Figure 2.1. The individual processing steps are depicted as rectangles with sharp corners while the data structures are shown as rectangles with round edges. In order to guide the reader, the specific modules from the application scenario from Figure 1.3 are used as examples for the processing steps, as used in (Marton et al., 2009c). The input shown at the top of the figure are the image streams provided by the used sensor, which can encompass left-right streams of stereo cameras, the distance or RGBD images produced by 3D cameras, or even infrared images. The output, depicted at the bottom comprises a set of reconstructions of the objects in the scene together with their categorization.

A preliminary offline step for an accurate data fusion is the calibration of the different sensing devices with respect to each other. The calibration enables the data interpretation algorithms to solve the sensor data correspondence problem and to bring the images in the same coordinate system. Typically the cameras are rigidly mounted, so we can assume that the extrinsic calibration parameters do not change.

The first step in during processing is the *registration of sensor modalities*, where the data of the individual sensors is combined to form a 3D point cloud representation of the scene. In case of RGBD cameras, each point in the cloud already carries information from both the RGB and depth sensors.

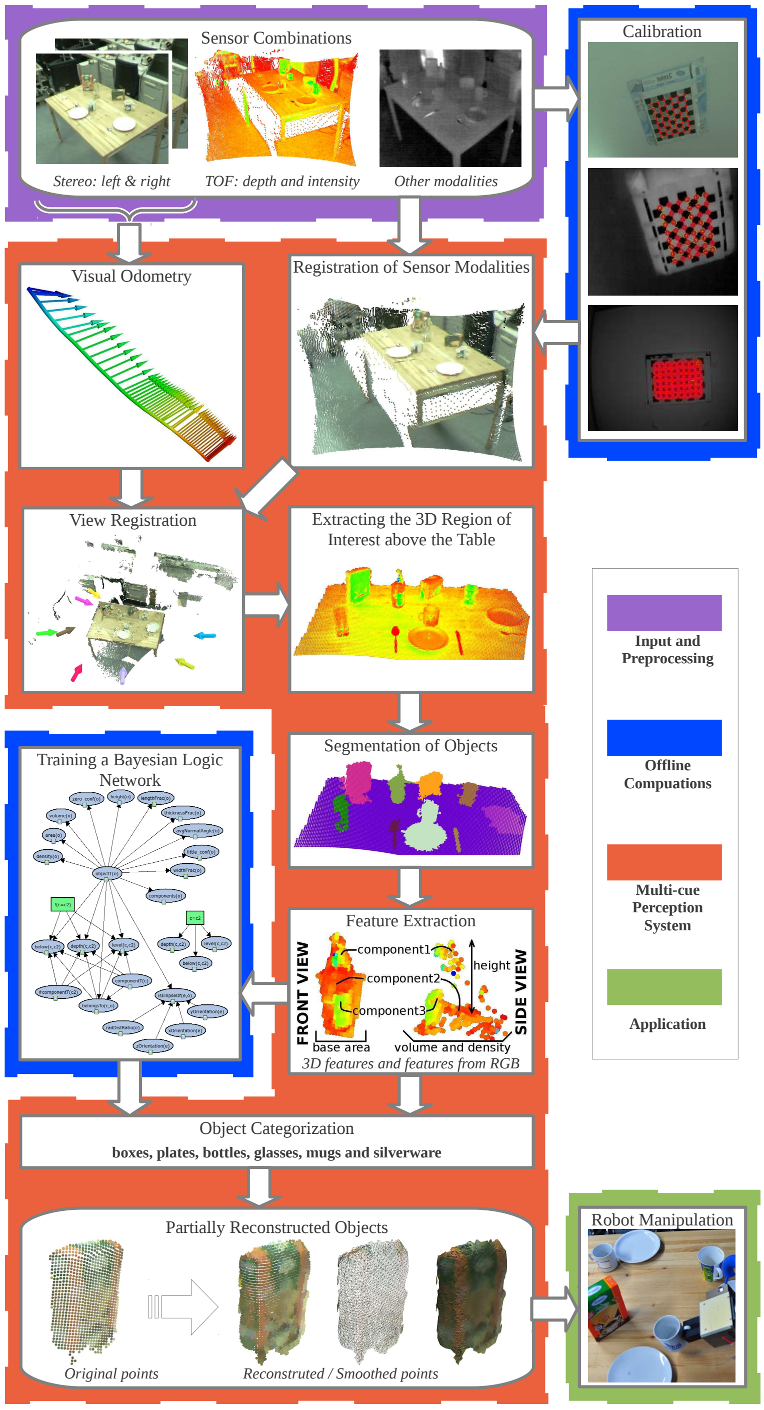


Figure 2.1: *An architectural overview of perceptual tasks, with steps exemplified based on the application from Figure 1.3.*

The *view registration* step registers the individual point cloud views into a global point cloud. This requires the estimation of the poses of the camera relative to the scene. This step is represented here by *visual odometry*, which is based on the stereo image stream obtained so far.

Next, an extraction of the 3D regions of interest is performed. In the example depicted in Figure 2.1, the robot has to interpret a table-setting scene. Therefore, the table needs to be segmented and the objects on top of it must be clustered and extracted as separate regions. Additionally, a table prior could be extracted from a knowledge database, where available (Rusu et al., 2008c). The *segmentation of objects* step will eliminate points that are generated by the table itself in order to find the sets of points that correspond to objects lying on the table. These sets are then segmented to form hypotheses about objects on the table. For each of them, the object feature inference mechanism infers the respective feature values.

The *object categorization* is then performed using machine learning techniques. The categorization step returns, for each perceived object, the most probable category and, depending on the classifier, a measure that signals how confident the system is that the categorization is right, or a confidence value for all the possible categories. In the presented example this is achieved by phrasing it as a probabilistic inference task based on learned statistical relational models about objects, their composition and their perceptual features. To this end, we have trained a Bayesian logic network that represents the relationships between features and object categories as a graph and then learns the probabilistic correlation between objects, categories, and their perceptual features from a set of training examples.

Finally, using the information provided by the categorization system, the *object reconstruction step* partially reconstructs object surfaces into more compact models by the use of triangular meshes.

The following sections describe the aforementioned processing steps in more detail.

2.2 Next Best View and Registration

In order to leverage the advantages offered by the embodiment of a mobile robot, special algorithms are needed for an efficient exploration of the environment and the fusion of the data obtained from various viewpoints. These tasks of next best view planning, visual odometry and scan registration are closely related to the SLAM research field, but constitute separate research directions with a long history already, as reviewed in subsection 1.3.1.

NBV algorithms and ICP-variants are used both for exploring and fusing scans of large environments as well as of small objects, and both are important functions for a service robot. In this section I will shortly present the various methods that can be used by our robot to explore its surrounding and build models of it, discussing their benefits and drawbacks given the sensor, the number of degrees of freedom (DOF) and application domain.

In the simplest case of environment exploration, the robotic platform has a fixed sensor mounted on its body, and it moves around on the floor, changing its 2D position and orientation. Thus, exploring such an environment and registering scans obtained by the 3D sensor is a 3DOF problem. This was the case in (Blodow et al., 2011a), where we described suitable NBV and ICP algorithms. Here, I will present their extension to 6DOF, while considering both color and geometric cues for aligning the views.

In order to maximize exploration it is advantageous to “look behind” occluding surfaces. This can be achieved by keeping a voxelized or octree representation of the workspace’s volume (Kriegel et al., 2012), and store in it the information if it was explored (and found to be free or occupied by an object’s surface) or not. The accessible boundaries between the two, i.e. between free and occluded voxel cells, can then be scanned in order to explore as much of the occluded space as possible. Possible sensor placement poses can be evaluated based on a suitable metric, as was done by (Blodow et al., 2011a).

However, expanding the method described there to 6DOF poses is problematic, as the space of possible poses is extremely large. Therefore, a reduced list of them needs to be generated by the following heuristics. First, exploration boundary regions can

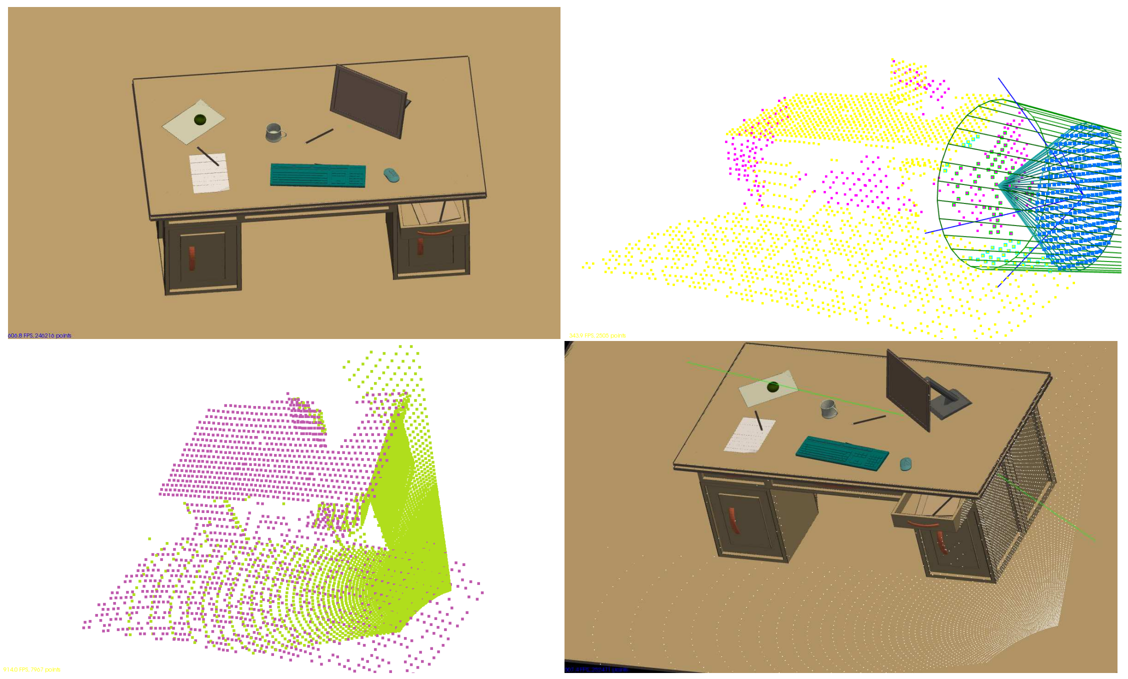


Figure 2.2: *Major steps of the 6DOF next best view algorithm. After a scan is obtained from the initial view (top left), the boundary between explored and occluded space can be sampled (purple points in the top right). These regions must be scanned such that to maximize the explored volume. 6DOF view frustums (marked by the four blue lines) can be evaluated for occluding surfaces by simulating the view through raytracing. After a suitable view is selected a second scan is added (bottom left) and the knowledge about free, occupied and occluded space updated. The last image shows the two view-rays (green lines), and the process is repeated until there are no large boundaries left to explore.*

be grouped into connected, nearly flat patches, treating them together. Second, for each laser and depth camera there is an optimal distance to scan from, which can be fixed, discarding too distant or too close points relative to the boundary to be scanned. Third, the orientation of the sensor should align with the largest extent of the boundary segment being analyzed.

These reductions in the size of the searchspace allow for a near real-time selection of poses using the procedure described in Figure 2.8. The main difference between the scan pose selection procedure from (Kriegel et al., 2012) and the one used here is the fact that we explore the occluded space directly instead of incrementally exploring along the edges of the scanned surfaces.

Even if successive scans are made from defined or computed poses, localization errors and mistakes introduced by calibration errors in the kinematic chains will result in misalignments between the captured scans. To correct these, the off the shelf solution is ICP, or one of its variants accessible in the PCL Library. ICP is by design based on point correspondences, even in the case when the error to be minimized takes into account surface normals in the "point-to-plane distance" method.

However, if only few good quality point correspondences can be computed, taking into account the planar patches found in the two scans and aligning those, proved to be a considerable benefit, as we showed in (Stanimirovic et al., 2013). The described method takes into account point-to-point correspondences (computed using both 3D and RGB-based features), but extends them with plane-to-plane correspondences as well, in a common optimization framework, showing superior results to alternative point-based approaches. Using the identified plane correspondences it is even possible to align scans without any estimate of their relative transformation (i.e. global registration or initial alignment).

For aligning partial point cloud views the example application from Figure 1.3 employs a Visual Odometer (VO) on the stereo camera images. This is because the low quality of the Time-of-Flight sensor does not allow an accurate registration. However, visual odometry is capable of accurate localization, given enough visual features in the scene.

For each series of monocular and disparity images from the stereo camera, the VO computes the camera motion between the views and estimates a rigid transformation (3D rotation and translation). Recent work (May et al., 2008) has addressed the problem of estimating the camera motion using a SwissRanger SR-3k TOF camera. However, this did not give satisfactory results for our setup, where the amplitude of the image is varying substantially depending on whether the camera is closer or further away from the object scene. Therefore, we used the Visual Odometry system from (Morisset et al., 2009) that seeks to estimate the camera motion by tracking CenSurE (Center Surround Extrema) features from one stereo image pair to the other. Then, the motion is scored using the pixel reprojection errors in both cameras, and its inliers are evaluated using the disparity space homography. The hypothesis with the best score (maximum number of inliers) is used as the starting point for a nonlinear optimization routine. Figure 2.3 presents the fusion of two different point cloud views using the camera motion estimates given by the VO framework.

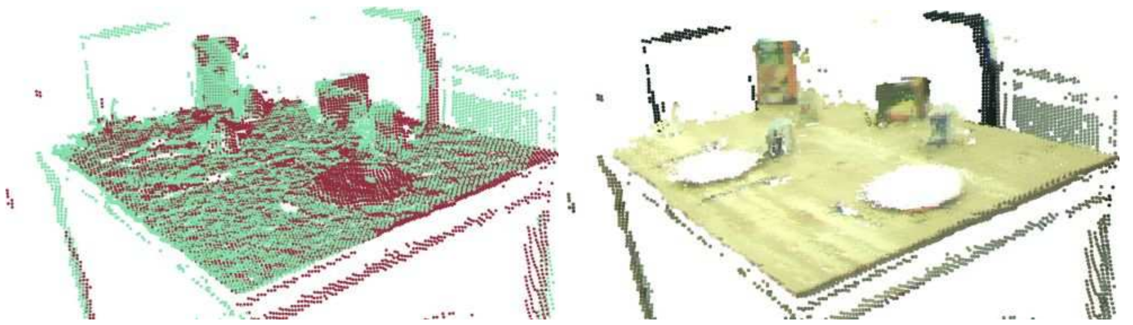


Figure 2.3: *Left: the fusion of two different point clouds acquired from separate views using the information provided by the Visual Odometry system. Right: the mapping of RGB information to the resultant global point cloud model.*

2.3 Surface Reconstruction

Having obtained and registered the measurements from multiple viewpoints, remaining problems in the data have to be corrected, and the real underlying surface approximated, in order to perform surface reconstruction. This is especially useful for creating accurate models of objects, but offers substantial benefits for environment modeling as well, as presented in (Rusu et al., 2008c; Marton et al., 2009b).

2.3.1 Robust Moving Least Squares

Single-scan, or registered point cloud data (PCD) typically contains holes due to scanning non-Lambertian surfaces (i.e. reflective or absorbing), and drastically changing point densities combined with doubled regions because of imprecise alignment. The surface reconstruction module transforms this PCD model into a homogeneous point distribution suitable for geometrical reasoning, containing interpolated points (which approximate the original input data) that account for the above mentioned problems. To do this we use our Robust Moving Least Squares (RMLS) algorithm (Rusu et al., 2007, 2008c). We summarize the algorithm here, extending the short overview provided in (Rusu, 2009) based on (Marton, 2007).

The first step of the algorithm is to compute a set of points which are to be fitted to the surface. If only surface reconstruction is required, the initial guess for the surface can be the original PCD itself. To up- or down-sample the underlying surfaces, an initial guess is computed by approximating the point cloud with a set of points Q that are in the vicinity of the original points and their coordinates are integer multiples of a specified resampling step size s:

$$Q = \{q_k \in \mathbb{R}^3 \mid q_k = s \cdot \left|\frac{p_j}{s}\right|,\ p_j \in P\} \tag{2.1}$$

Note, that s can be selected also adaptively, by making it depend on the estimated local surface curvature. The uniqueness of the obtained points must be ensured and extra points must be added to areas where a hole was detected. The algorithm is similar to the one for boundary points detection, with the difference that here we check if a point is inside of an object, so the computed maximal angle must be sufficiently small to ensure correct filling.

In the second step, for each point $q_k \in Q$ a local neighborhood P_k is selected, by using either a fixed number of nearest neighbors, or all neighbors in a fixed radius. The first method can result in neighbors that are relatively far away form the currently fitted point, and their influence must be minimized when fitting the point. The second method has the disadvantage that it may yield too few neighbors if the search radius is too small, but it ensures that the neighbors will surround the point to be fitted. In practice this second method is preferred, as the relevant parts of the point cloud are typically nearby, thus possessing a high enough density.

Weights are assigned to every neighbor $p_i \in P_k$ of q_k based on the distance to the currently fitted point.

$$W_{(p_i)}^{q_k} = \exp - \frac{dist(p_j, q_k)^2}{h^2},\ q_k \in Q,\ p_j \in P_k \tag{2.2}$$

Each point q from the initial guess will be projected onto the plane fitted through its nearest neighbors. To fit a plane to the local neighborhood P_k, the minimization of the weighted squares of the error (defined as the sum of weighted distances of the points from the plane) was suggested by an iterative projection approach (Alexa et al., 2004). This method, although weighted, takes into account the whole neighborhood of the projected point, meaning that it will be influenced by a relatively small patch of outliers, which we found to be common in point clouds obtained by laser scanners. These groups of outliers cannot be eliminated statistically (e.g. using the method from Rusu et al. (2008c)) because they have approximately the same density as the points that are actually on the surface, so a method with a larger breakdown point is needed. We developed a weighted distance based version of the Random Sample Consensus (RANSAC) algorithm (Rusu et al., 2007). This algorithm will yield good results as long as the plane having the highest inlier weights is not one formed by noisy measurements (which can be the case even above 50% noise in the neighborhood).

To identify the inliers $p_l \in P_{in} \subset P_k$, a random sample is selected from the neighborhood and a plane D is fitted to them. To measure the probability that the obtained plane is correct, the weighted distances of the points near the plane are summed up and mapped on the $(0,1)$ interval. Iterating through a statistically defined number of steps, the inliers will be obtained as the points close to the fitted plane. These inliers will be used as the reduced neighborhood of the projected point. It must be noted that the weighting ensures that the plane will be fitted to points that are closest to q_k. A comparison between the Weighted Least Squares and our Weighted RANSAC can be seen in Figure 2.4.

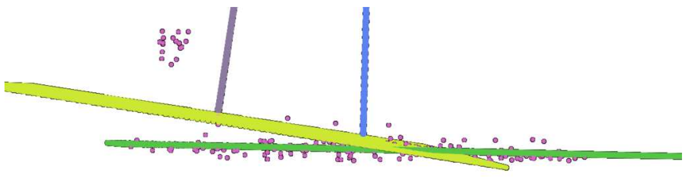


Figure 2.4: *WLS and WRANSAC (dark green) for a cloud with 10% outliers.*

The fitted plane's equation and the curvature can be obtained by performing eigen-analysis on the covariance matrix of the selected inliers (while the color of the obtained point can be approximated with the weighted average of the inliers' colors). We assemble a weighted covariance matrix from the points p_l of the inlier set (where $l = 1...m$):

$$C = \sum_{l=1}^{m} W_l \cdot (p_l - \overline{p})^T \cdot (p_l - \overline{p}),\ \overline{p} = \frac{1}{m} \cdot \sum_{l=1}^{m} p_l$$

followed by the eigenvector V and eigenvalue λ computation $C \cdot V = \lambda \cdot V$.

In the third fitting step the polynomial approximation is performed. For this, the xyz coordinate system is transformed into the uvn local coordinate system and a high-order bivariable polynomial is fitted to the heights of the points above the plane and the height of the fitted point is recalculated. For example, a polynomial of order 3 is of the form:

$$h_{(u,v)} = c_0 + c_1 u + c_2 v + c_3 uv + c_4 u^2 + c_5 v^2 + c_6 u^2 v + c_7 uv^2 + c_8 u^3 + c_9 v^3 \qquad (2.3)$$

where u and v are coordinates in the local coordinate system lying on the tangent plane, having the projection of the query point as the origin.

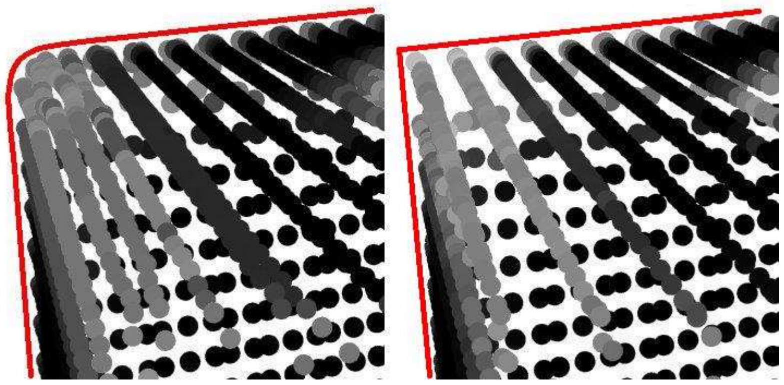


Figure 2.5: *Comparison of edge and corner preservation between standard MLS (left) and our Robust MLS (right) for a cube. Both pictures show curvature values in grayscale: low curvatures are darker.*

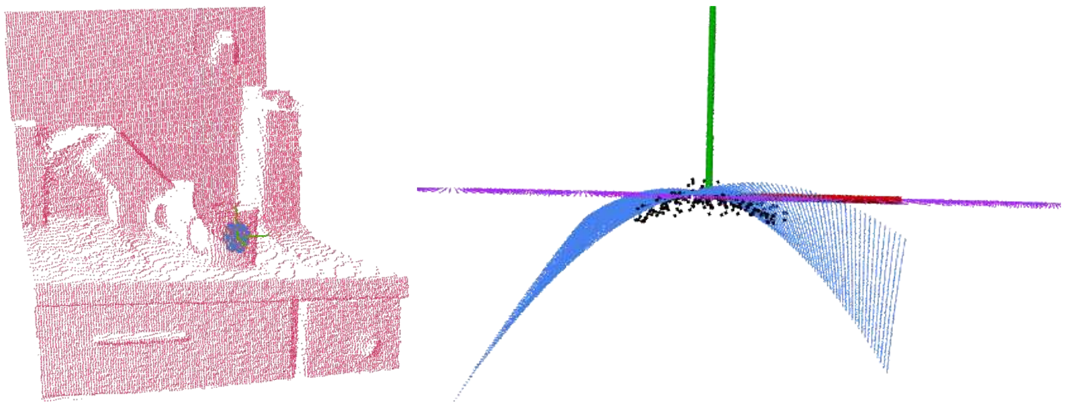


Figure 2.6: *Fitted polynomial to a local neighborhood. Left: local coordinate frame (green) and points forming the neighborhood (blue). Right: supporting plane (purple) and polynomial approximation (blue).*

To compute the vector of unknown coefficients c_i, we minimize the error function as in (Alexa et al., 2003; Wang and Oliveira, 2003) (see Figure 2.6). Since the reference plane was computed robustly, the height function defined on it preserves the edges sharper than in regular MLS (see Figure 2.5). After resampling, surface normals have to be re-computed as the normal of the fitted polynomial in point q, through the partial derivatives. We can easily compute the normal $\vec{n}$ of the estimated surface by computing the two partial derivatives at $(u, v) = (0, 0)$ in the local coordinate system and their cross product:

$$\vec{n} = (\vec{U} + c_1\vec{N}) \times (\vec{V} + c_2\vec{N}) \tag{2.4}$$

where c_1 and c_2 are the first order coefficients for the u and v coordinates, respectively, $\vec{U}$ and $\vec{N}$ being the corresponding unit vectors.

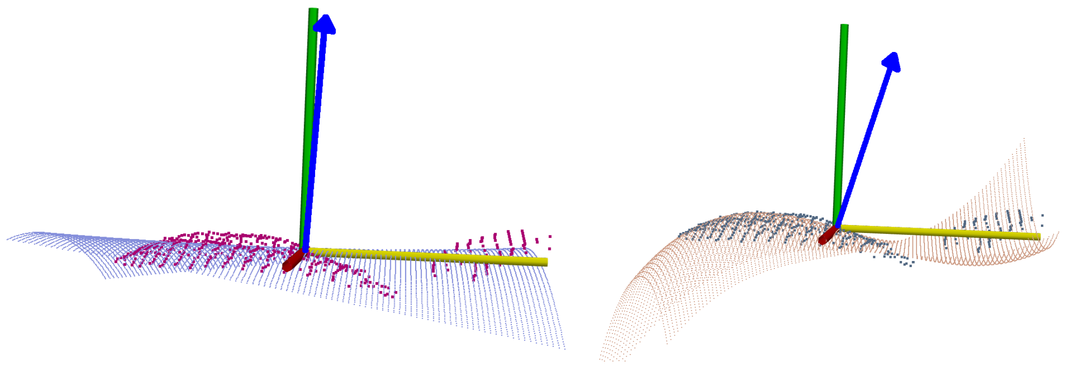


Figure 2.7: *Effect of the fitted polynomial's order to a local neighborhood. Local coordinate frame and points forming the neighborhood overlayed with the fitted polynomials of order 2 (left) and 3 (right), along with the estimated surface normal (blue arrow).*

While it is computationally more expensive to fit 3^{rd} order polynomials, in this case the results are better especially if one considers the correctness of the recalculated surface normals (see Figure 2.7). Fitting higher order polynomials is more and more costly as the number of members increases, and the additional precision gained is insignificant.

To process 3D points obtained by the noisier or sparser sensors (like the Hokuyo laser or the Kinect camera), the presented RMLS method can be sped up and simplified. For such data it proved sufficient to use the Principle Component Analysis (PCA) approximation of the tangent plane's normal, and re-fit the points and normals using the MLS optimization as shown in Figure 2.8. Moreover, for 2.5D datasets the neighborhood searches can be sped up by considering the organized grid-like structure of the points.

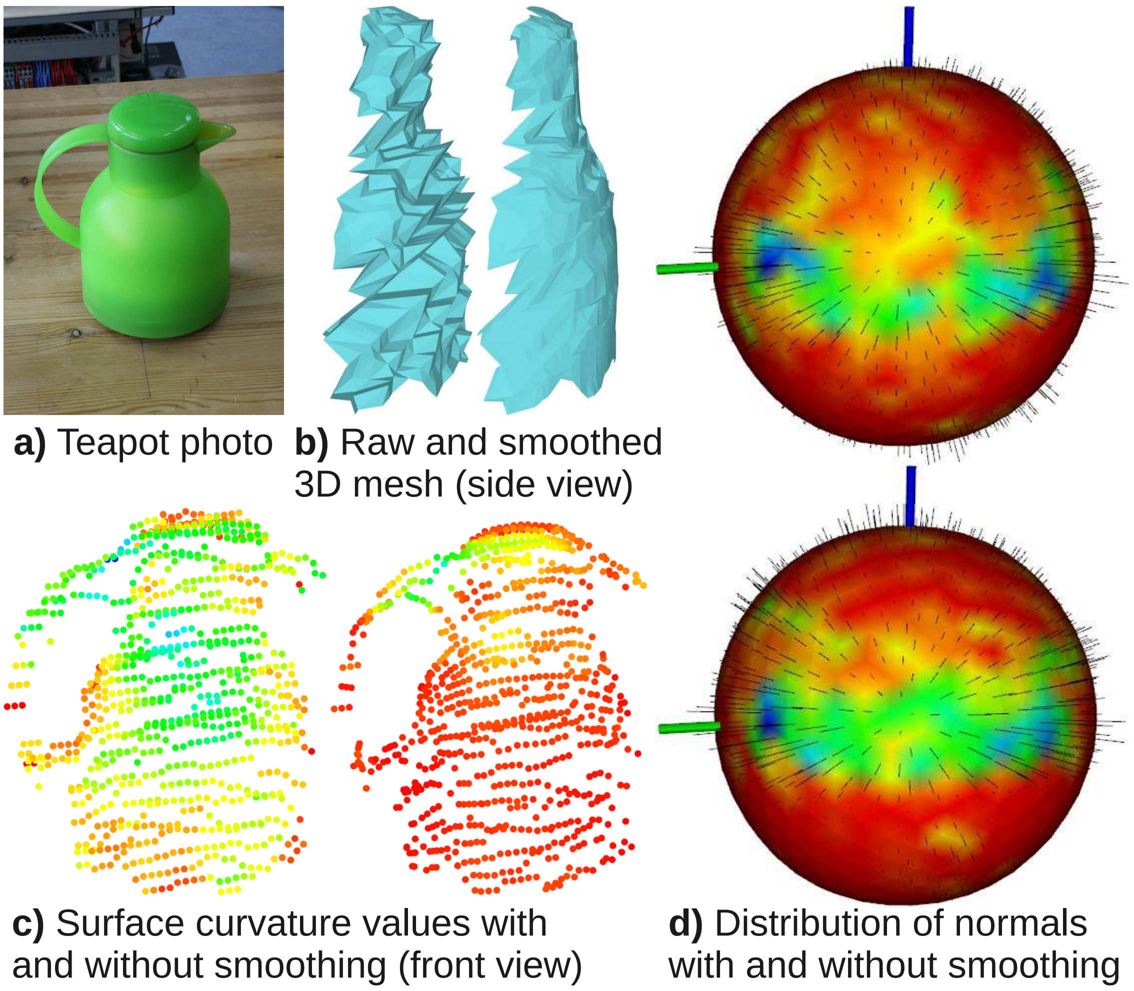

a) Teapot photo
b) Raw and smoothed 3D mesh (side view)
c) Surface curvature values with and without smoothing (front view)
d) Distribution of normals with and without smoothing

Figure 2.8: *Effects of smoothing on surface normals and curvature estimation.*

2.3.2 Triangulation

These datasets represent discrete, sampled representations of the world, so a transformation into continuous surfaces is required for advanced grasp planning systems. To this end, we investigate the following computational problem: given a noisy 3D point cloud model of a real-world scene as depicted in Figure 2.9, create a continuous triangular mesh surface representation which reliably represents the scene, as fast as possible.

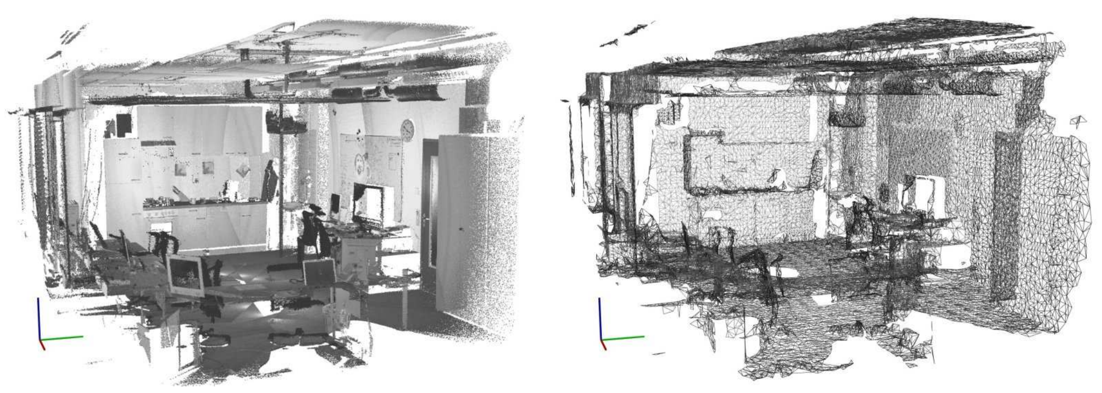


Figure 2.9: *Left: point cloud dataset of an indoor kitchen environment (15 millions of points) shown in grayscale intensity. Right: its mesh representation (down-sampled for visualization).*

Our approach for creating the triangular mesh is based on the *incremental surface growing* principle (as devised by (Mencl and Müller, 1998)), because it can deal with all types of surfaces where points on two different layers can be distinguished by a distance criterion. Our algorithm selects a starting triangle's vertices and connects new triangles to it until either all points are considered or no more valid triangles can be connected. In the second case a new seed triangle is placed in the unconnected part and the triangulation is restarted.

In contrast to well known surface reconstruction methods from computer graphics (Gopi and Krishnan, 2002; Scheidegger et al., 2005; Amenta et al., 2001; Dey and Goswami, 2003), our method works reliably on noisy 2.5D data scans acquired by mobile robots navigating in indoor and outdoor environments.

The proposed surface reconstruction algorithm called Greedy Projection Triangulation (GP3), is partially based on the triangulation algorithm presented in (Gopi and Krishnan, 2002). The method has the advantage that it is a greedy type approach, where edges are written directly and are never deleted, making it fast and memory efficient, yielding the first correct triangulation (Marton et al., 2009b).

Triangulation is done incrementally: i) for each point p, a neighborhood is selected by searching for the point's nearest k neighbors in a sphere with radius $r = \mu \cdot d_0$

that adapts to the local point density (d_0 is the distance of p to its closest neighbor and μ is a user-specified constant); ii) the neighborhood is projected on a plane that is approximately tangential to the surface formed by the neighborhood and ordered around p; iii) then the points are pruned by visibility and connected to p and to consecutive points by edges, forming triangles that have a maximum angle criterion and an optional minimum angle criterion.

The projection along the estimated surface normal occurs in the last two steps previously mentioned. In (Gopi and Krishnan, 2002), this normal is estimated as the average of the normals of the triangles that have p as one of their vertices. This simplification makes sense only if the underlying surface is assumed to be smooth, i.e. the deviation between the normals of any two incident triangles on a vertex is less than 90°. The projection of the neighborhood along the computed normal is then ordered around p, and the points that have to be connected to it are identified based on their visibility with respect to previous triangle edges (here only the current boundaries of the mesh have to be considered). The remaining points are connected to p in consecutive pairs, forming triangles.

The advantage of this method is that it preserves all points and makes no interpolation, over which we want to have greater control in our algorithm. However, for noisy datasets the smoothness and locally uniform sampling constraints presented in (Gopi and Krishnan, 2002) do not hold. In our implementation, this problem is solved, and we adapt the resulting triangle sizes to the surface's properties. A similar method is presented in (Scheidegger et al., 2005), though it suffers from the limitation of not being able to deal with sharp features, and it only works with densely sampled point sets.

Our hole filling approach is similar to the one presented in (Wang and Oliveira, 2003), but we identify holes automatically and grow the triangle mesh into it by the use of robustly fitted vertices.

The main contributions of our surface reconstruction method include the following ones:

- *adaptability to variable densities:* Our method does not assume smooth surfaces or locally uniform neighborhoods, and can adapt to the point density variations present in 2.5D datasets.

- *near realtime performance:* We employ fast *k*d-tree searches for nearest neighbor computations (Arya and Mount, 1993) and optimized lookups of previously computed triangle edges, to achieve extremely fast visibility checks (i.e. tests for the intersections of triangle edges).

- *noise robustness:* Estimated surface normals at a point are computed using a robust Weighted Least Squares method on the full neighborhood, rather than averaging the normals of the current adjacent triangles as done in (Gopi and Krishnan, 2002).

- *memory efficiency:* For each point we propagate a front wave, updated continuously, containing the point's associated advancing triangular edges. This has the advantage that we do not have to remember all the triangles in the entire dataset, and the lookup of edges that have to be checked for visibility is performed locally, as opposed to Gopi and Krishnan (2002)).

- *fast updates for incremental scans:* By determining the overlapping area between each new registered data scan and the existing surface mesh, our method is able to grow the existing model without recreating the entire triangular mesh.

- *supporting point labels:* The proposed surface reconstruction framework is flexible with respect to a local triangulation stopping criterion, in the sense that previously determined point labels can be used to decouple triangular meshes. An example is presented in Figure 4.5, where a distance connectivity criterion was used to segment and label objects lying on a horizontal planar surface, and thus resulting in separate triangular meshes for each object.

The algorithm has no constraint on the data being watertight or complete, and the data can contain occlusions and may be a combination of several registered partial views. Therefore, the data may represent both large environmental models such as complete rooms, but also smaller segmented objects lying on horizontal planar surfaces (e.g. tables, cupboard shelves, counters) which may be manipulated by the robot. These particularities of our application scenario, require a set of constraints on our surface reconstruction system's behavior:

- since the robot acquires data in stages (i.e. partial scans of the world), whenever new data scans are available, the current existing mesh needs to be updated efficiently;
- to support dynamic scenes, such as objects being moved around from one place to the other, a mechanism for decoupling and reconstructing the mesh as fast as possible needs to exist, and separate objects should be triangulated separately;
- a re-sampling step with variable vertex densities needs to be available, as described in subection 2.3.1, for performing triangulation when the input data contains high levels of noise, measurement errors, and holes, i.e. missing measurements.

The last step is especially useful for large datasets when a compact representation is needed for high variations in surface curvature, requiring higher vertex densities on sharp edges and lower on flat surfaces.

In order to deal with the special requirements posed by our application and to achieve near real-time performance, we employ several techniques and optimizations. An important difference between our approach and (Gopi and Krishnan, 2002) is that our method works directly on the 3D point cloud data as we facilitate fast nearest neighbor searches with *k*d-trees, and avoid creating an additional dexel structure, which requires a two step search through the entire dataset.

Additionally, since we deal with large datasets, we optimize the lookup of edges that have to be checked for pruning points by the visibility criterion in a neighborhood.

Whenever a new triangle is generated, the two neighbors that are connected to each fringe point (i.e. a point on the boundary of the current mesh) through the edges of the advancing front are saved. This creates a redundancy that is required for cases when only one of the two vertices is in a neighborhood, but speeds up the selection of the edges since we find them locally, by looking at the fringe points.

This has the advantage of automatically providing a closed loop of boundary points (i.e. points lying on the boundary of the underlying surface) on the sides of the surface and around its internal holes. We use these points to fill small holes in the data, as explained in 2.3.1.

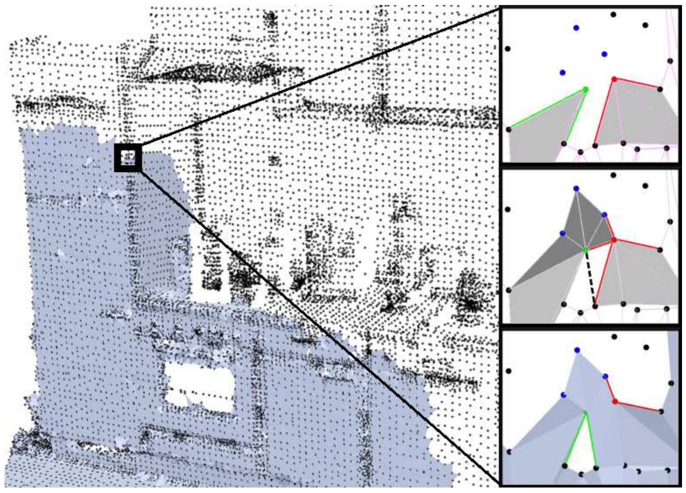


Figure 2.10: *Two parts of the advancing fronts touched in a single point (red) resulting in multiple incident front edges for that point, which was solved by adding an extra triangle (black dashed line) – see pictures on the right. The green point is the current source for triangulation, the blue and red points are the points in its neighborhood, the green lines are the front edges of the green point, while the red ones are for the red point. The light gray triangles are the ones originally connected to the green and red points, while the dark ones would be the only new triangles if no correction would occur, producing four front edges connected to the red point – see picture in right-middle.*

The disadvantage of this optimization is that it creates special cases which have to be solved locally when the advancing fronts touch in a single point (this happens at $\approx$ 6% of the points in a dataset). We solved this problem by connecting additional points until each point has only two front edges, thus avoiding to run into the same situation again (see Figure 2.10 right, from top to bottom).

Due to the amount of noise present in our datasets, the modeled surface is not smooth, and averaging normals of incident triangles (as done by Gopi and Krishnan (2002)) in a point $p \in P$ to obtain an approximation of the surface normal gives unsatisfactory results. To account for noise, we compute a weighted least squares plane in the p's neighborhood, and use the normal of the plane as an estimate of the true surface normal (as detailed in 2.3.1). Optionally, we refine the normal by fitting a bivariate polynomial to the neighborhood, as by Alexa et al. (2004). This extra step is computationally more expensive than averaging triangle normals, but this way triangles don't have to be stored and looked up, rendering our approach more memory efficient.

Note that the pruning of small triangles is mandatory for our application because noisy real world scenes are not smooth, and thus vertices of a small triangle could appear as “flipped” in different neighborhoods, creating small inconsistencies in the triangulation.

Finally, we are using our RMLS algorithm to achieve locally uniform sampling where needed by resampling the underlying surface while preserving its edges. The level of smoothing can be adjusted and it produces adaptive vertex densities for triangulation, as presented below.

The triangulation method is capable of correctly adapting to locally uniform sampling. If the point density is variable, but there are smooth transitions between the different densities, the local neighborhood search radius adapts to it and makes sure that no point is left out. When distances between the scan lines are substantially different than the distances between the points in a scan line, the data does not satisfy this sampling criterion (see Figure 2.11).

To ensure a correct triangulation, extra points need to be interpolated between the existing ones. We are using our RMLS algorithm to resample the point data, as described previously. Figure 2.12 presents the results obtained after resampling an outdoor urban scene to achieve a local uniform density using RMLS.

Resampling has another advantage that we can exploit: by influencing the density of the generated vertices based on the estimated curvature of the underlying surface (Pauly et al., 2002), more points can be fitted to regions where more detail is needed. The density for the average estimated surface curvature in the dataset is specified by the user and for other values it is linearly interpolated with a given slope (Marton et al., 2009b). This ensures that the reconstruction will be finer in these areas (see Figure 2.13), and thus *adaptive* with respect to the underlying surface curvature.

An additional hole filling strategy is to use the known holes from the internal boundary loops that result from triangulation. We automatically fit extra vertices to the covered

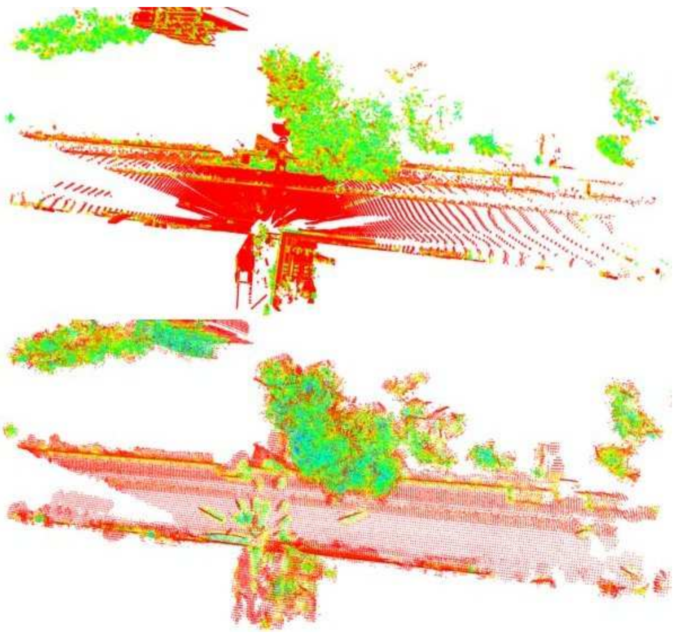


Figure 2.11: *Top: 3D scan of an urban scene with 497220 points (top view) and non-uniform local point density; bottom: resampled scene with 289657 points and even sampling distribution and filled holes. The colors represent the estimated surface curvatures (red meaning low, yellow colors medium, and green to blue meaning high curvature).*

area if its size is smaller than a user-defined threshold, and advance the boundary loop further to close the hole.

To evaluate our system's efficiency, we performed several tests using multiple datasets. Since the input datasets are comprised of millions of points with dense neighborhoods at small scales, it would be extremely inefficient to reconstruct the surface using the complete data. To obtain sensible data resolutions, the datasets are resampled with RMLS using variable vertex densities. The results are presented in Table 2.1.

The timings presented in Table 2.1 represent the actual time our methods took to compute the surface mesh for each dataset, excluding any I/O operations. The hardware used was a standard notebook at 2Ghz with 2GB of RAM. Depending on the desired mesh accuracy, the input dataset could potentially be downsampled even more, resulting in faster surface mesh computation times.

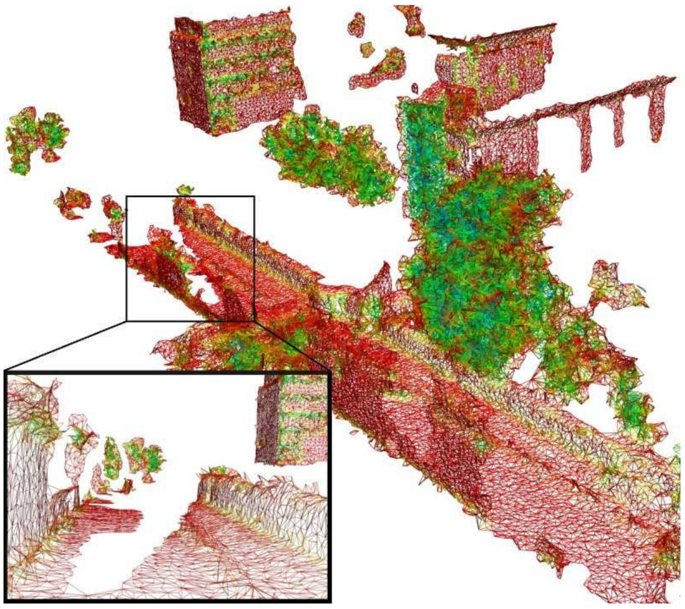


Figure 2.12: *Triangulation of the urban scene from Figure 2.11 (downsampled for visualization) with a closeup on the far side of the street, where the triangulation fails in the original dataset. Colors represent estimated surface curvature (red meaning low, and blue high curvature).*

Dataset	**points**	**triangles**	**time**
Mug with adaptive resampling (Fig. 2.14)	4754	9330	0.079 sec
Kitchen front (Fig. 2.10, final result in Fig. 2.13)	24378	45828	0.816 sec
Chicken (from the Stanford dataset)	29518	57476	1.262 sec
Oil pump (from the Stanford dataset)	30932	61082	1.610 sec
Kitchen 360°, downsampled (Fig. 2.9)	40242	71436	2.347 sec
Dragon (from the Stanford dataset)	41841	80101	2.107 sec
Parasaurolophus (from the Stanford dataset)	47758	92898	2.093 sec
Ljubljana urban scene (Fig. 2.12)	65646	114452	8.983 sec
Chef (from the Stanford dataset)	69007	134633	3.104 sec
Rhino (from the Stanford dataset)	79934	155872	3.593 sec
Radiohead street scan (Fig. 2.16)	329741	546516	17.857 sec

Table 2.1: *Performance evaluation of our surface reconstruction method for multiple datasets. Several standard datasets were used from the well known Stanford 3D Scanning Repository*[1].

The results from Table 2.1 are visualized in Figure 2.15. Aside from small variations that depend on the local topology of the datasets, the timings depend linearly on the

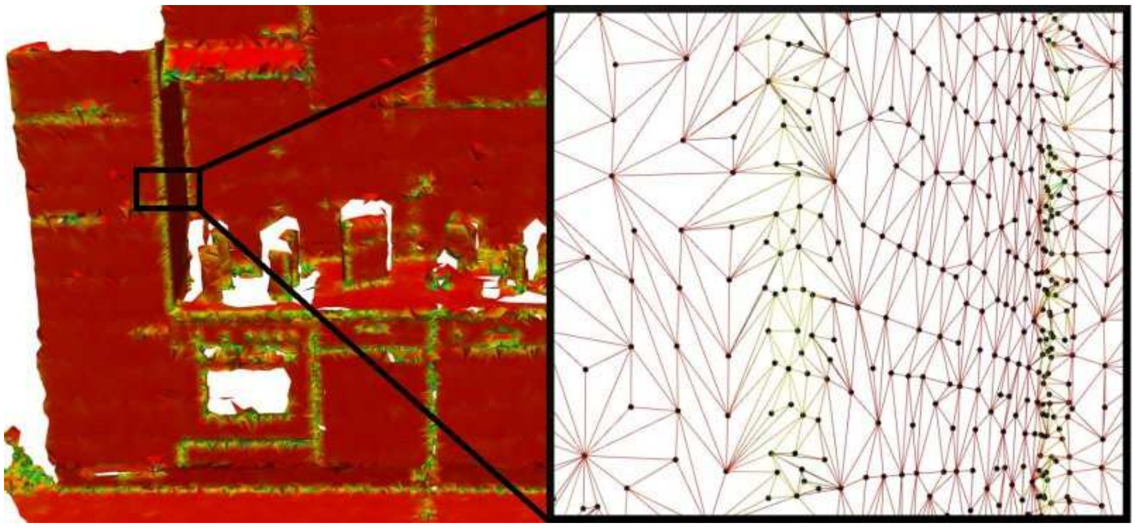


Figure 2.13: *Triangulation of a surface with adaptive vertex density. Left: triangulated surface colored based on estimated surface curvature (red meaning low, and green high curvature). Right: close-up on an edge where the vertices generated by RMLS are denser than on planar areas.*

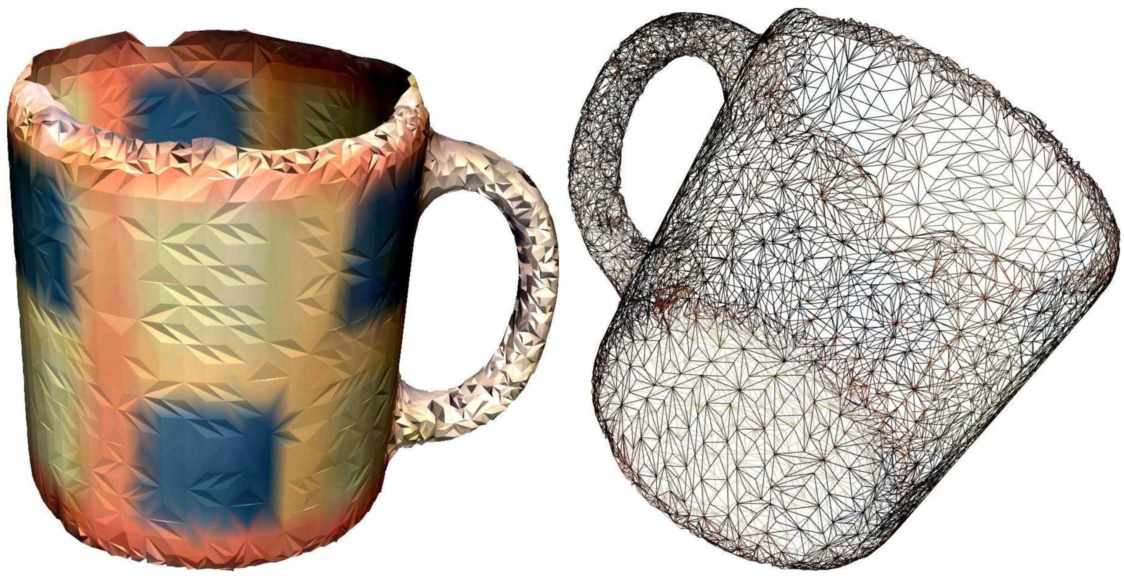


Figure 2.14: *3D scan of a mug triangulated, shown as surface on the left, and as triangles on the right.*

number of points used, as expected from a local triangulation (the logarithmic nearest neighbor search is much faster than the remaining operations). The exception is the Ljubljana street scan with the problematic point density that has to be accounted for, as discussed earlier.

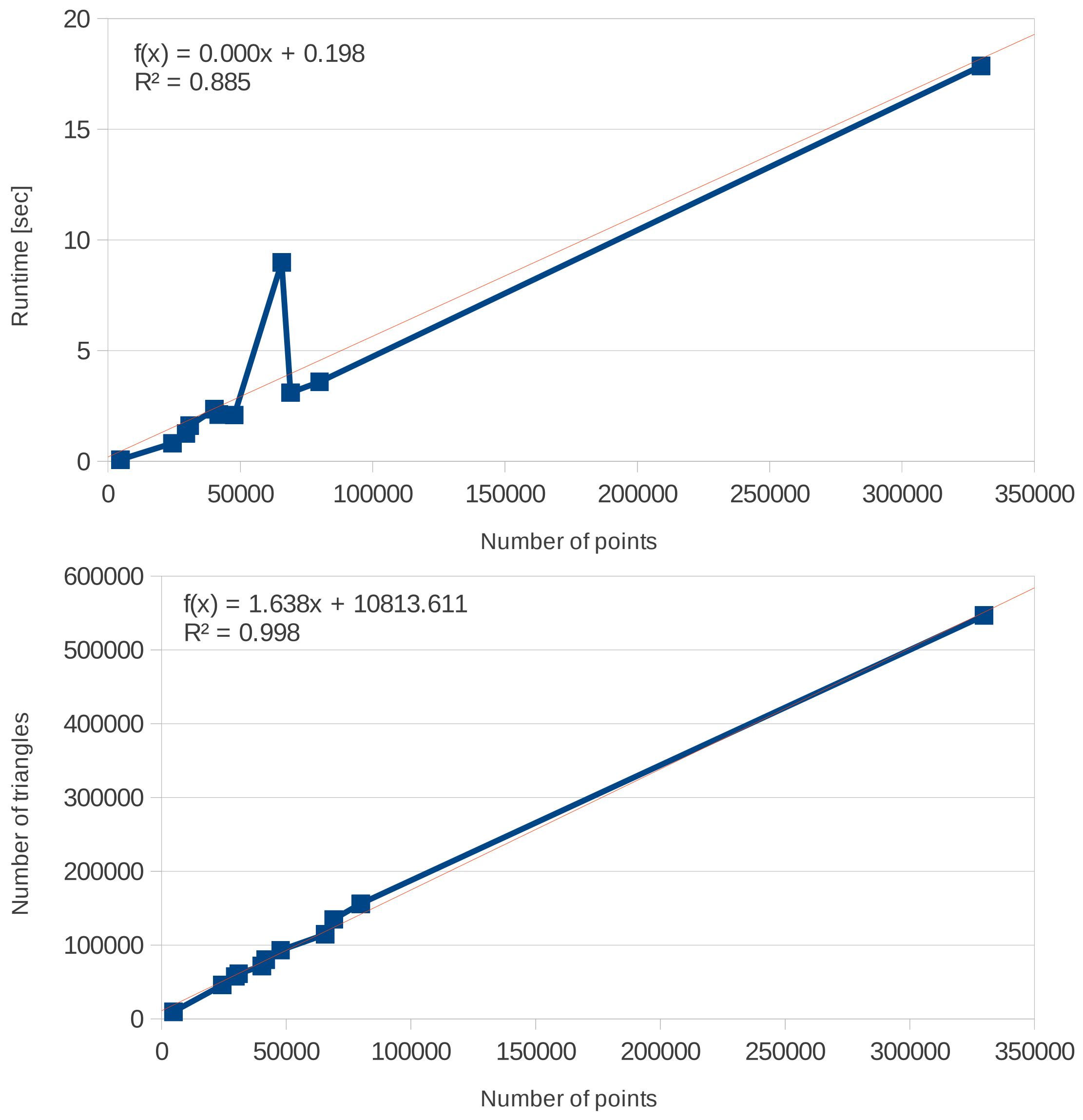


Figure 2.15: *Linear time complexity (top) and number of triangles (bottom) as a function of the number of points. The linear approximation is visualized using the orange lines, with the equation and correlation coefficient values shown in the respective plots. Data from Table 2.1.*

The datasets presented in Table 2.1 are all formed of one part and were all reconstructed using the entire data as a whole, thus leaving little space for further optimizations, like for example a parallelization of the triangles computation. In contrast, the dataset presented in Figure 4.5 is comprised of 22 parts with 252719 points total, where a part is defined by a subgroup of points which share the same attribute or label. A triangulation of the entire dataset without using the point labels results in a total computation time of 14.842 seconds. However, by using the extra information present in the data, we can create each triangle mesh individually, and thus parallelize the results. By performing the reconstruction on 4 different parts at once, the average computation time was brought down to 3.905 seconds.

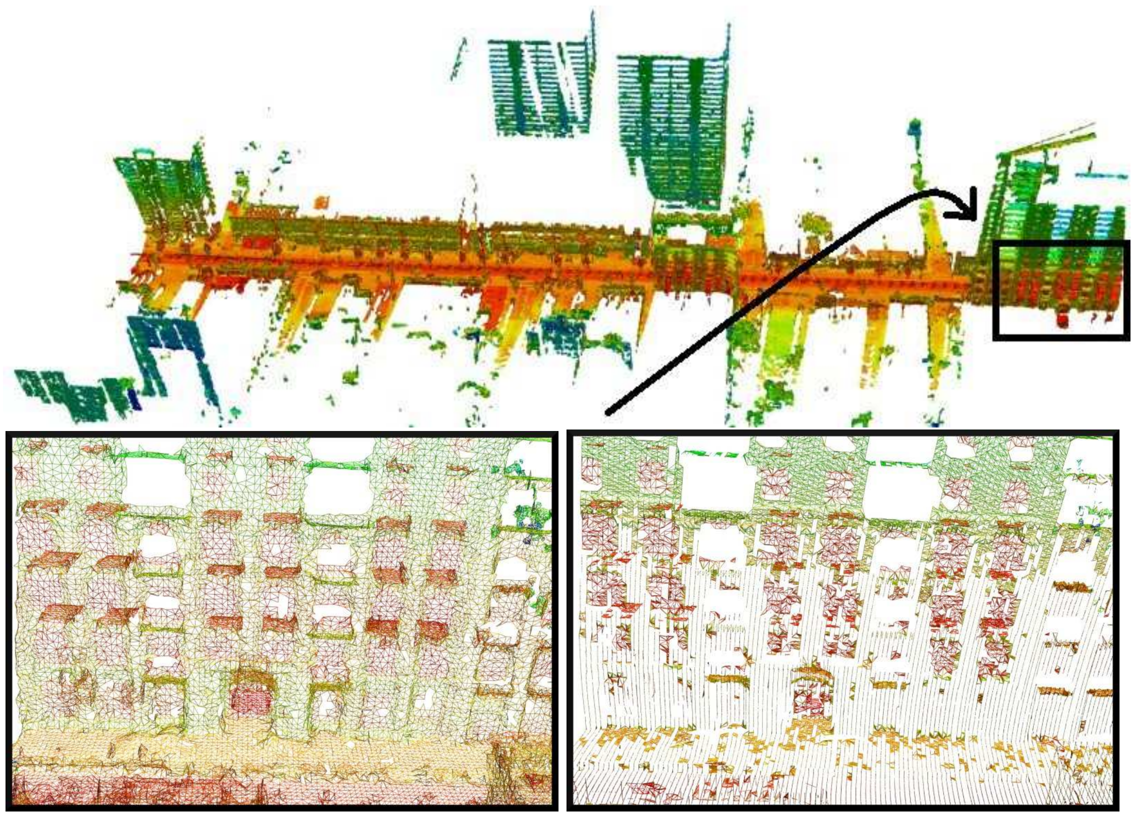


Figure 2.16: *Scan of an approximately 500m long stretch of an urban street resampled and triangulated (top) and closeup (bottom left). The triangulation presented in (Gopi and Krishnan, 2002) failed because the dataset does not respect the local uniform sampling criterion (bottom right).*

In order to compare our results with the method by Gopi and Krishnan (2002), we obtained the 3D scan of a stretch of street taken for a Radiohead video clip. The bottom left part of Figure 2.16 presents the correct triangulation by our algorithm for the scan, where the density of points in a scan line is much higher than the distance between the scan lines. The triangulation method presented in (Gopi and Krishnan, 2002) failed to correctly triangulate the dataset since the locally uniform sampling criterion was not met. Even with $\mu = 5$ and a running time of 1134.36 seconds the result is not satisfactory as seen in the bottom right.

2.4 Segmenting Object Hypotheses

As discussed in (Blodow et al., 2010; Pangercic et al., 2010; Marton et al., 2011), approaches to segment camera or point cloud percepts into distinct objects can be roughly split into two paradigms: i) locating an instance of a known object in the scene, using e.g. appearance models or feature descriptors, and ii) segmenting the sensory data without prior knowledge about the objects or their views in order to acquire information about them.

In the first case, a direct search for each appearance can be performed, or a set of feature words can be extracted from the sensor data and matched to a model database as in (Torres et al., 2010; Hinterstoisser et al., 2010). These approaches are usually limited by the fact that for locating one or more of a very large number of possible objects, performance can decrease significantly.

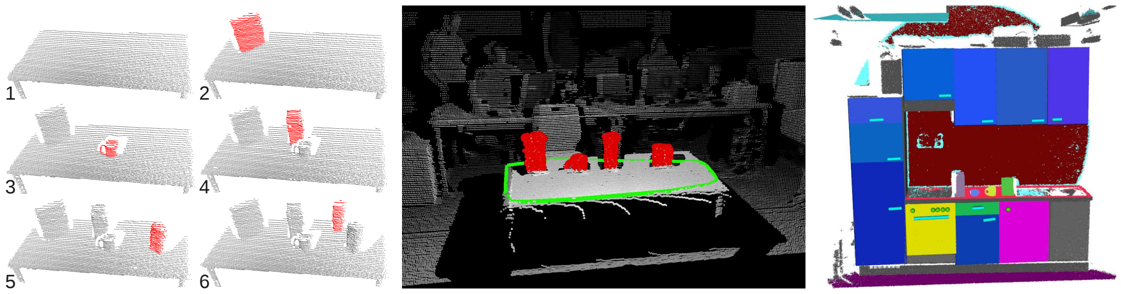


Figure 2.17: *Different object detection methods, from left to right: background subtraction, planar support, and environment map based.*

In the second case, as the objects to be detected are unknown, it is possible to segment scenes under certain assumptions, as presented in Figure 2.17:

- *Background Subtraction:* Supposing successive addition of objects to the scene, one can easily detect successive changes even in cluttered scenes[2].
- *Planar Support:* Supposing a detectable support that assures physical stability allows for segmentation of objects in an euclidean sense, as long as the objects are apart from one another (Rusu et al., 2009a). However, too much clutter is not acceptable in this case.
- *Environment Maps:* Supposing a known 3D environment, such as stored in a Semantic Object Map, makes it possible to define regions of interest in which objects are expected, which can then be processed as in the *Planar Support* method. This approach is a natural extension to the 3D semantic mapping efforts.

Apart from these, part based segmentation and recognition is getting more and more popular, due to its potential to deal with cluttered scenes, based only on a single frame (Marton et al., 2014). We will give a short overview here on obtaining parts of various types through segmentation techniques, but will analyze and evaluate this idea in greater detail in the following.

In our example application from Figure 1.3, the biggest errors in the depth images are produced while scanning shiny or transparent objects, and these are hard to account for using other sensor sources. Therefore, we make use of a machine learning approach for categorization, and use the results to decide how to partially reconstruct the missing 3D data. For a partial view or a given set of partial views registered together, our framework proceeds to extract horizontal planes which might support objects on them.

The supporting planes are computed by making use of standard robust estimators such as RANSAC (Fischler and Bolles, 1981) and fitting restricted planar models to it. In our implementation, we assume that the camera is close enough to the actual table and looking at it, so the largest planar model found closest the viewpoint is assumed to be the table. Once a model has been found, we compute the table inliers $\mathscr{T}$ and boundaries, approximate them with a convex polygon, and extract all point clusters

[2]More advanced methods include the learning of a foreground/background model for example.

located on it by checking if their projection on the plane intersects the polygon. This results in a rough geometrical segmentation of the scene.

An important aspect of the sensed 3D data is that because of highly reflective or non-refractive surfaces, points which physically describe a given object will be estimated and sampled somewhere else in space, thus returning erroneous measurements. This mostly occurs with glasses, bottles, and silverware, but can seldom be encountered for certain cups as well. Although in most cases these erroneous points are clustered together above the table plane, there are situations where they are distributed along the table surface, thus making it impossible for any geometric method to segment them accurately (see Figure 2.18).

A solution to this problem is to perform a clustering of the point cloud in the Swiss-Ranger amplitude space, which can be used together with the geometric information to extract different object components. Our clustering method is based on a region growing approach, where points with the same amplitude characteristics are added to a list of seed points, and a region is grown until no neighboring points with the same characteristics are left (Marton et al., 2009c; Rusu et al., 2008c). Since it is impossible to set the amplitude clustering thresholds to some fixed values that would work for any table setting, we automatically estimate them as follows.

We approximate the distribution of the amplitude values of the points in $\mathscr{T}$ with a Gaussian distribution ($\approx$ 65% being in the $\mu \pm \sigma$ interval) and we compute the mean μ and standard deviation σ. Then, we estimate the mean $\overline{\mu}$ of the amplitude differences between neighboring points in $\mathscr{T}$ as

$$\overline{\mu} = \frac{1}{nr_T} \sum_{t_i \in T} \left(\frac{1}{nr_N} \sum_{n_j \in \mathscr{N}_i} \left| t_i^{int} - n_j^{int} \right| \right) \tag{2.5}$$

where $\mathscr{N}_i \subset \mathscr{T}$ is the set of neighboring pixels of t_i in the amplitude image that are also on the table ($n_j \in \mathscr{T}$), t_i^{int} and n_j^{int} are the amplitude values at the respective points, and the number of points in $\mathscr{T}$ and $\mathscr{N}_i$ is nr_T and nr_N respectively.

Since the goal is to prevent the under-segmentation of the table and the objects on it, we can tighten the statistical threshold and use $\overline{\mu}/2$ as a connectivity criterion in amplitude to limit the 3D region growing in the table points. Additionally we impose a maximum divergence of two standard deviations to avoid growing too far even if there is a small gradient. This way we obtain multiple region patches on and in the vicinity of the table.

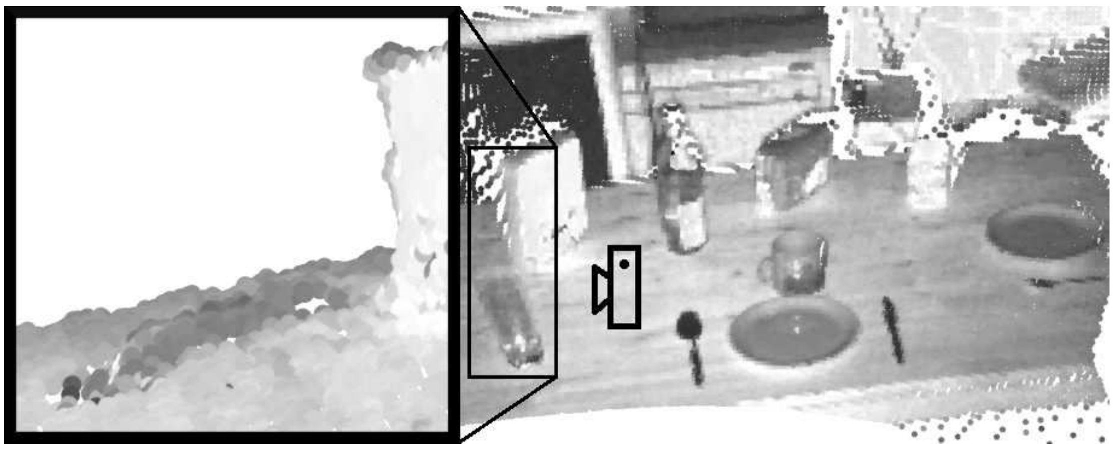


Figure 2.18: *A table setting view in amplitude space (grayscale). A close-up of a glass object is shown in the left part of the figure, showing no geometry information at all. For a side view including the bottle see Figure 2.3.*

Due to the strict thresholds, the different parts of the table will be over-segmented, but they can be easily merged by verifying their estimated plane normal to the one estimated for all the points in $\mathcal{T}$. The rest of the points are marked as either belonging to an object on the table (e.g. a plate) or to a *shadow*. The parts of the scene where the amplitude values change drastically will produce very small regions. These *shadow* points are typically produced either by shiny objects like silverware or transparent objects that create patterns on the part of the table which they occlude. Glasses and bottles in particular reflect the waves of the sensor such that some of the resulting points will be below or inside the table (see Figure 2.18, for example).

By using the relative viewpoint information of every scene, we can locate the points that are above or slightly below the table and group them into connected components – labeled as belonging to objects outside the table's plane.

Thus, our region growing method will yield three types of point labels: *outside* – belonging to a region of an object that is outside the table, *near* – belonging to a region of an object that is near the table, and *shadow* – belonging to a small region with distinctive amplitude value near the table. The resulting labeling can be seen in the left part of Figure 2.19, where each connected group of points with the same

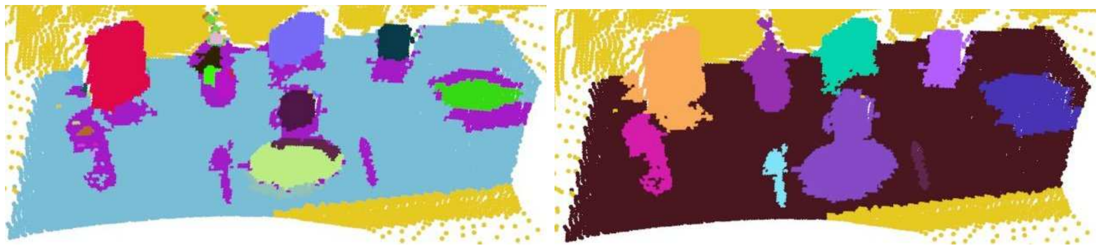


Figure 2.19: *Left: different object regions in different colors together with points on the table (light blue) and discarded points (yellow). Note that shadow points are in very small regions and are marked with the same label (purple). Right: object candidate clusters in random colors.*

label is considered a component of an object. These labeled points are then clustered together in 3D to form object candidates (see Figure 2.19, right).

As a summary, over-segmentation is typically easier than a correct segmentation of a difficult scene. Such an approach can provide informative object, or object part hypotheses for machine learning approaches, as we will see in the following. Other over-segmentation approaches also exist, out of which we found the surface normal angle based one to be the most relevant for geometry-based approaches (Mozos et al., 2011). Such a segmentation is presented in Figure 2.20.

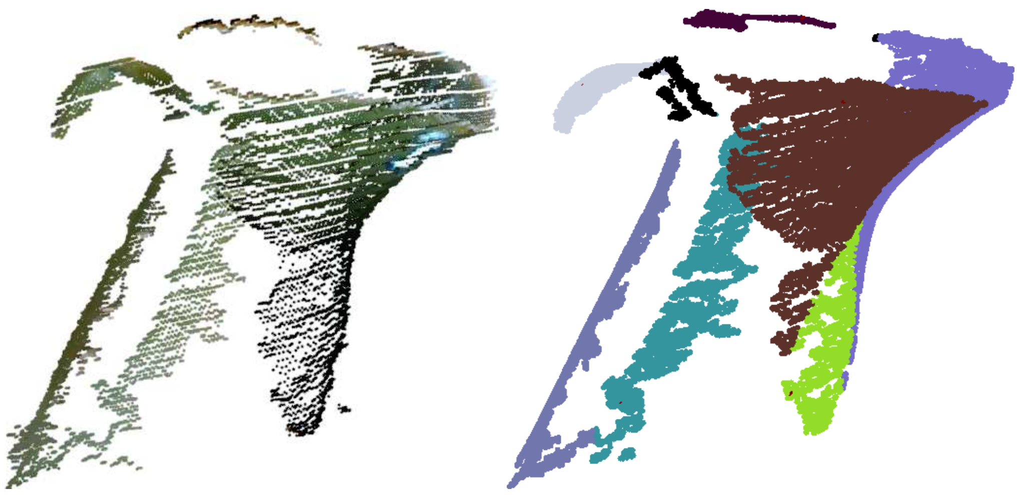


Figure 2.20: *Cut out object from a Kinect frame with visualized color (left) and parts identified by region growing using a normal angle criteria (right).*

2.5 Estimating Descriptive 3D Features

Pinz (2005) defines categorization as being a *generic object recognition* (generic OR), this being the opposite of a *specific OR*, which deals with the recognition of a specific object (an instance of those categories). Example general categories are cups, plates, boxes, etc., while in specific OR one aims to recognize a specific instance of box, cup etc. Since most household objects from the same category share similar shape, but distinctive appearance, 3D features are typically used for generic ORs, while visual features for specific ones.

Here we will focus on geometric ones, which will be shown in Chapter 5 to be more suitable to generalize to novel object instances of trained categories. More specifically, we will describe the design, analysis and evaluation of the local (i.e. point-based) Radius-based Surface Descriptor (RSD) and the Global Radius-based Surface Descriptor (GRSD), introduced in (Marton et al., 2010a) and (Marton et al., 2010b) respectively.

2.5.1 Local Surface Radius Estimation

The most descriptive 3D features are normal based as we reviewed in (Aldoma et al., 2012), so accurate normal estimation is one of the first preprocessing steps in 3D point cloud processing. Thus the RMLS smoothing and normal estimation presented in 2.3.1 is a necessary precondition for computing descriptive local features.

While estimated surface normals are important, they alone tell little about the surface's type. The estimation of the curvature using the ratio of eigenvalues of a neighborhood's covariance matrix on the other hand neglects surface normals and we found it to be too inaccurate. As argued in (Marton et al., 2010a), the approximated radii of the smallest and biggest fitting curves to a local neighborhood are values with physical meaning, which can be tied directly to the underlying surface without the need for classification. These values form the RSD feature which is computed starting similarly as in the case of spin images with a local support (Johnson, 1997). However, we have the added advantage that we consider the normals of the neighboring points directly, and that we can extract values that intuitively describe the local surface (as shown in Figure 2.21).

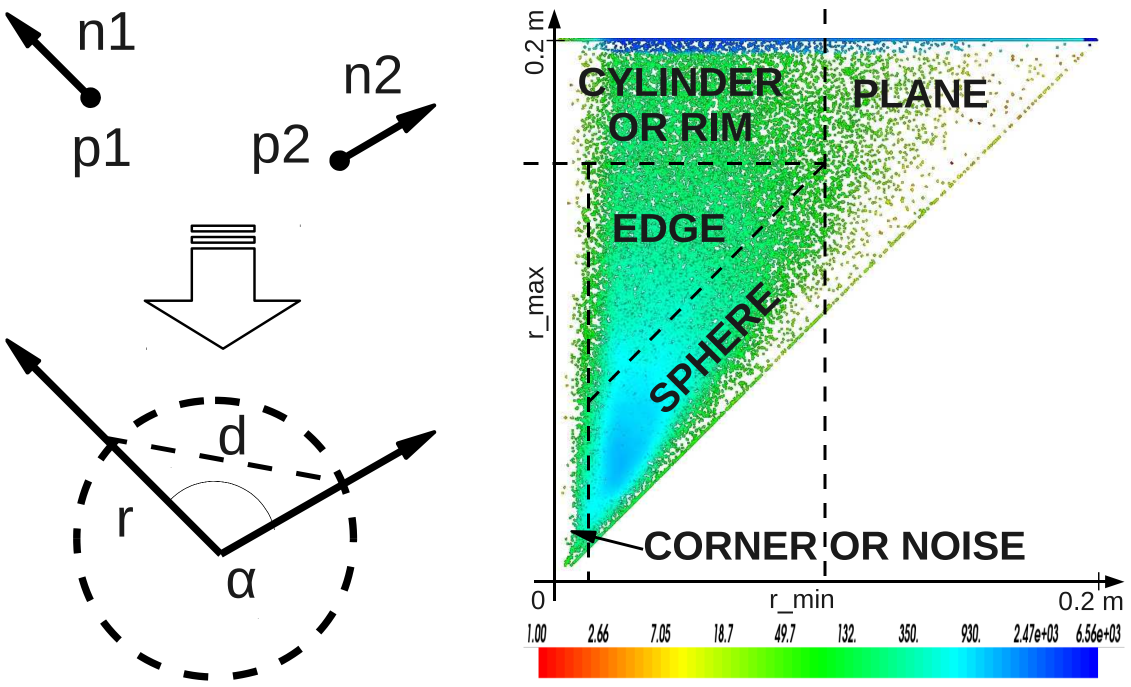


Figure 2.21: *Left: Illustration of how a distance and an angle between normals define a radius as per Equation* (2.6). *Right: Identifying surface classes without the need of classification. The plot shows the log density/number of points in a* 1 *cm range in the feature space of* r_{min} *and* r_{max} *for a variety of objects, with the rough ranges corresponding to certain shapes marked.*

If we look at the case of the sphere, for each point all the circles that fit to its neighborhood have the same radius, namely the radius r of the sphere itself. For each of these circles we can write the following relation between the distance d of a point on the sphere from the original point and the angle α between these two points' (undirected) normals:

$$d_{(\alpha)} = \sqrt{2}r\sqrt{1-cos(\alpha)} \tag{2.6}$$

From this we can see that given the distance and the angle between two point normals one can approximate the radius of the circle the point is on.

In the case of an ideal plane, this estimated radius will always be infinite with all neighbors, since they have parallel normals. A point on a cylinder is on multiple

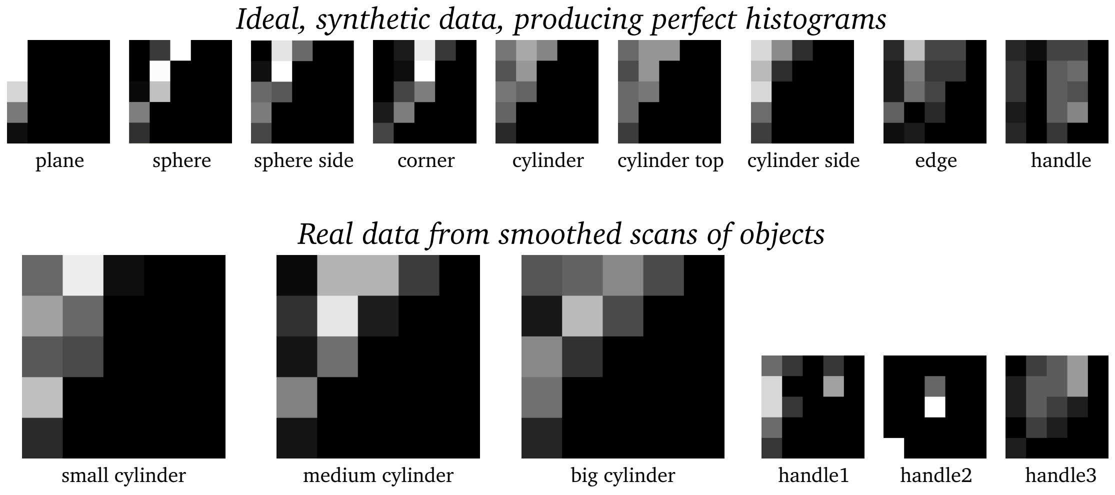


Figure 2.22: *Computation of the Radius-based Surface Descriptor (RSD). First, a 2D histogram of normal angle difference and distance from a reference point to its neighbors is computed. Then, from these histograms the physical surface radii can be read as the slopes of the different lines going through the lower left corner.*

circles (ellipses actually), and the radius estimated with different neighbors will vary between the minimum radius (the radius of the cylinder) and the maximum radius (infinity). For corners and edges the estimated radius changes similarly as for spheres and cylinders (see Figure 2.22).

Given a point on a surface along with its neighbors, this minimum and maximum radius can be estimated using the model in Equation (2.6) by solving the equation system for r given the maximum and minimum angles for different distance intervals. To make the estimation easier, we can exploit the fact that $\alpha \in [0, \pi/2]$[3] and the Taylor decomposition of Equation (2.6) is simple:

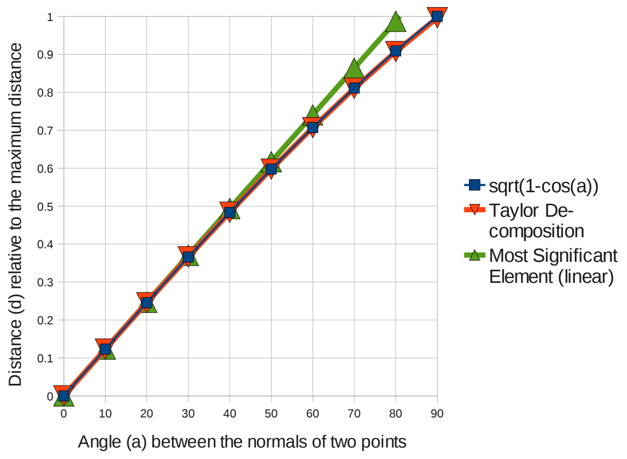
$$d_{(\alpha)} = r\alpha + r\alpha^3/24 + O(\alpha^5) \tag{2.7}$$

where $O(\alpha^5)$ indicates the existence of elements with order 5 and upwards. Thus we can assume $d \approx r\alpha$ which renders the problem of finding r to a simple regression – as seen in Figure 2.23.

[3]Since we don't assume that normals are oriented consistently we compute α as the angle between the lines defined by the two points and their normals.

To find the minimum and maximum radius of a point based on its nearest neighbors, we compute the minimum and maximum angle in each of 5 distance bins, and compute the corresponding radius value. This introduces a small error (r_{min} is typically under- and r_{max} over-estimated), which we found to be less than 1 cm for all points of a set of 10 synthetic spheres with different surface radii R (for which $r_{min} = r_{min} = R$). For the real scan of the teapot in the right part of Figure 2.23 we found similar errors (its top cylinder having a radius of 4 cm and the bottom 7 cm).

In the cases where the two normals do not form an equal angle with the distance vector, the algorithm would assume they do and compute the radius like that. However, as the process is repeated for every point pair, and we are only interested in the minimum and maximum radius, these small errors are either evened out, or produce the typical result for curvature computation – i.e. estimated radii on a plane start dropping from very large to very small values the closer the point is to an edge. In the case of RSD this is also aided by the fact that the previous surface and normal estimation step also relies on a neighborhood of points. Please see Figure 5.20 for an example involving sharp edges (tea box).



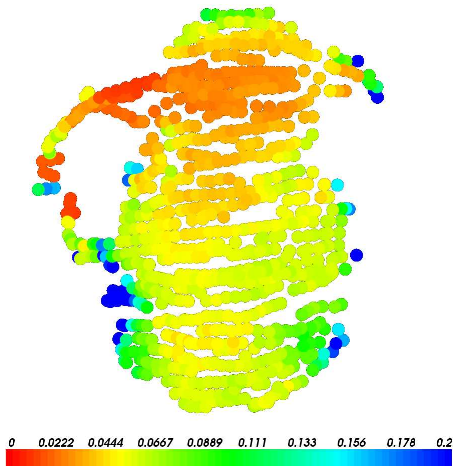


Figure 2.23: *Left: variation of point-to-point distance by angle, measured from the center point of a half-sphere. Right: radius estimation results for a scan of a teapot – values are color-coded from red (radius of 0) to blue (radius of* 0.2 *m or higher). Note that for visualization purposes the values were capped at* 0.2 *m, as planar regions have an infinite or very large radius.*

This feature is easy to compute, while still being very descriptive (see Figure 2.24) and does not require consistently oriented normals. Because it is a continuous value that estimates the real minimal metric radius of the curve each point lies on, it can be used for example as a prior when sampling points to fit different surfaces.

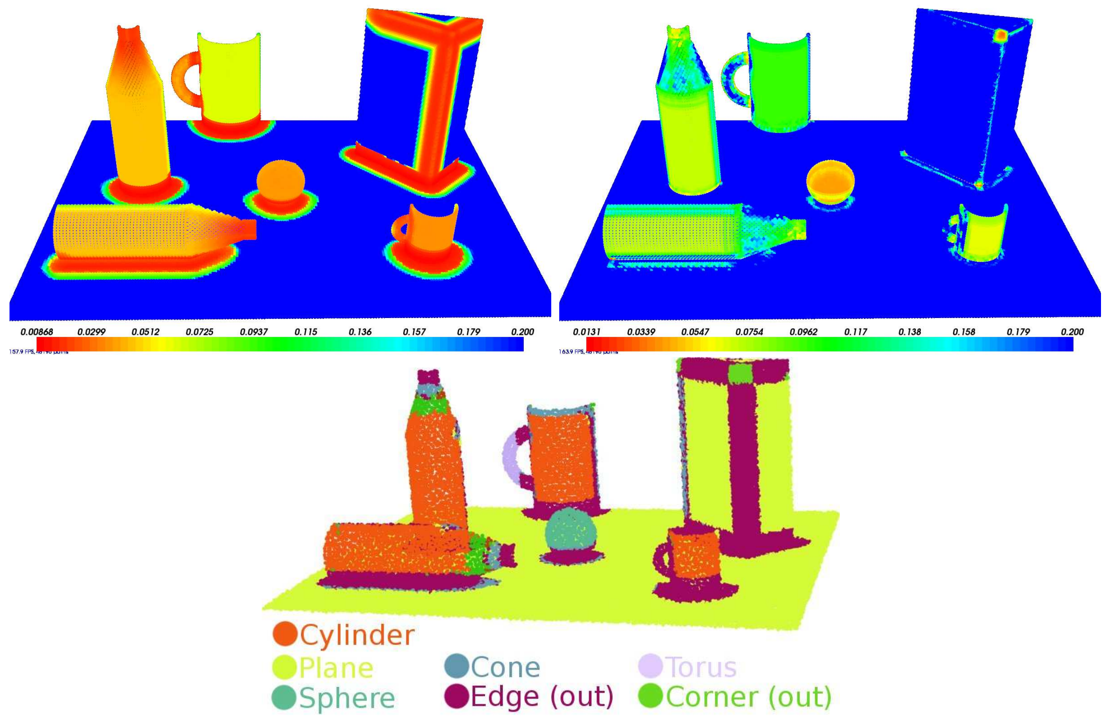


Figure 2.24: *Comparison of RSD on the top (r_{min} and r_{max}) to PFH (Rusu et al., 2008a) on the bottom (the slower, but more descriptive version of the popular FPFH (Rusu et al., 2009b) descriptor) for surface type recognition. RSD is faster to compute and more descriptive as it assigns continuous values instead of discrete labels. It only has two dimensions as opposed to 81 or 125 used in PFH (33 in FPFH), and no machine learning is needed for surface type labeling due to physical meaning (see next subsection). As we discussed in (Aldoma et al., 2012), for using RSD as a local feature in (global) registration tasks the whole underlying 2D histogram should be used.*

For model reconstruction scenarios, points with very high estimated minimal radius are preferred when fitting boxes, while they are avoided along with points having a very small radius when searching for rotational models. We incorporated this preference in the random sampling step for each model fitting algorithm (Marton et al., 2011). Since the surfaces scanned with these noisy sensors appear to be “bumpy” even after MLS, the estimated minimal radius is in some cases smaller than it ideally should be. Still, they provide our RANSAC methods with a fast and accurate weighting for the points, depending on the model to be fit. The inliers are also weighted by how well their radius fits the model.

Based on the minimum and maximum radius it is possible to devise simple heuristic rules that categorize the surface types and enable the computation of a global descriptor (GRSD and GRSD-) as detailed in the next subsection.

2.5.2 Global Feature from Local Surface Types

We are using a two layered classification scheme for geometric categorization (Marton et al., 2010b), where local surface labels are combined in a global descriptor. To speed up the local surface classification, we label voxels directly instead of individual points, as those are needed for the global classification. This reduces the complexity proportionally to the average number of points in a voxel.

In each voxel with a width of 2.5 cm, we compute the minimum and maximum radius and label the surface by successive checks of the radii: planar ($r_{min} > 0.1$), cylindrical (if not planar and $r_{max} > 0.175$), sharp edge/corner or noisy (if not cylindrical and $r_{min} < 0.015$), spherical (if not edge and $r_{max} - r_{min} < 0.05$) and rim (in the remaining cases). This is a simple and fast way to categorize surfaces (see Figure 2.21), and it divides the feature space formed by the radii well enough for the computation of a global feature. We picked the local surface labels to have an intuitive significance, but the actual correctness of these labels is not relevant for the global classification.

Once all voxels are annotated locally using a geometric class, our processing pipeline constructs a global feature space that can produce a unique signature for each object cluster. This space is based on the idea that, for a set of labeled voxels, a global feature can be constructed by observing the relationships between all these local labels (and the encompassed free space). Since the labels represent geometric classes obtained from the classification of RSD descriptors, we call this new feature the Global Radius-based Surface Descriptor (GRSD).

Figure 2.25 shows two sets of feature vectors for different objects generated by the GRSD estimation. We were using the Octomap library (Wurm et al., 2010) to facilitate ray intersection tests in the voxels encompassing the objects.

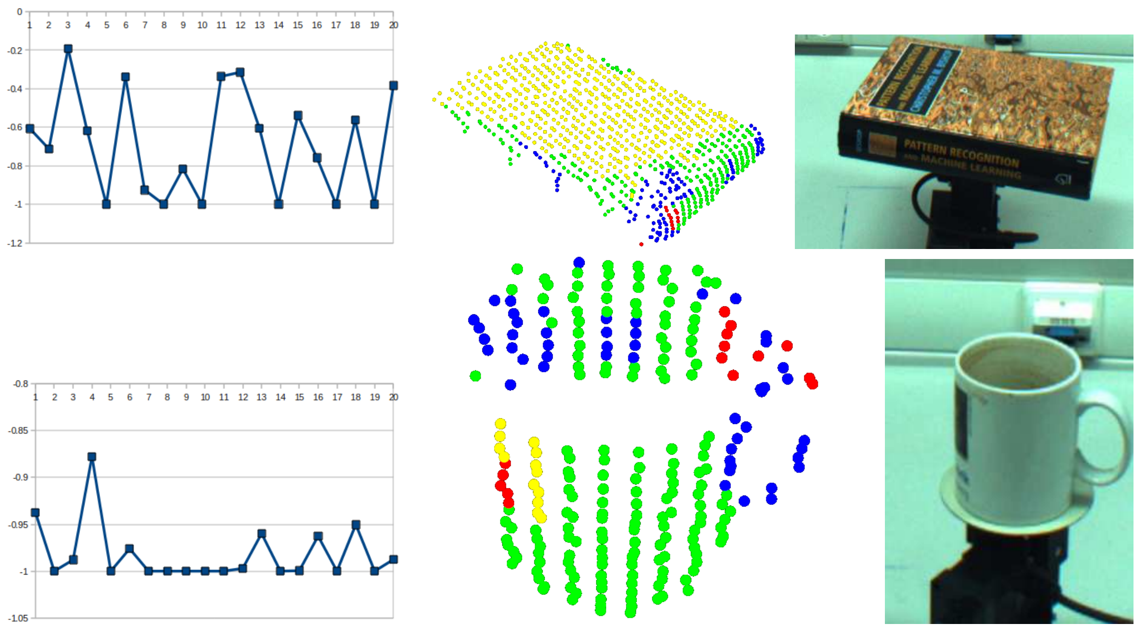


Figure 2.25: *Plots of GRSD histograms (left) and identified surface types based on RSD (middle) for a flat box (i.e. book, upper row) and a cylinder (i.e. mug, bottom row). Left: the GRSD histogram bin values are scaled between -1 and 1 (based on the value ranges found in the training data). Middle: the colors represent the following local surfaces: red - sharp edge/corner (or noise), yellow - plane, green - cylinder, light blue - sphere (not present), and dark blue - rim (i.e. boundary, transition between surfaces).*

The computation of GRSD is similar to GFPFH (Rusu et al., 2009c), but we use voxel-based labeling based on RSD, and we sum up the individual histograms instead of computing their distribution, to further reduce computational complexity. This way, the complete processing of a cluster (correcting, estimating normals, computing the voxelized RSD values, labeling voxels and constructing the GRSD) takes between 0.2 and 0.5 seconds (depending on object size) on a single core working at 2 GHz.

As described in (Kanezaki et al., 2011c), we adapted the GRSD feature to be additive[4], by simplifying it to a histogram of transitions between surfaces of different type (and free space), thus neglecting the ray-tracing step. This simplified feature (which we

[4]The additive property means that adding the feature computed for two point sets separately results (approximately) to the feature obtained for the joined point sets. This will be an important aspect for object recognition in clutter by efficient matching of part-graphs to scene-subgraphs in the application scenario from Chapter 5.

called "GRSD-") is very efficient to compute, and we compared its descriptiveness to the original implementation for geometric categorization tasks. To generate the results from Figure 2.26, we used over 7500 scans from the object model database presented by Lai et al. (2011a), with a 2 : 1 split between training and testing data.

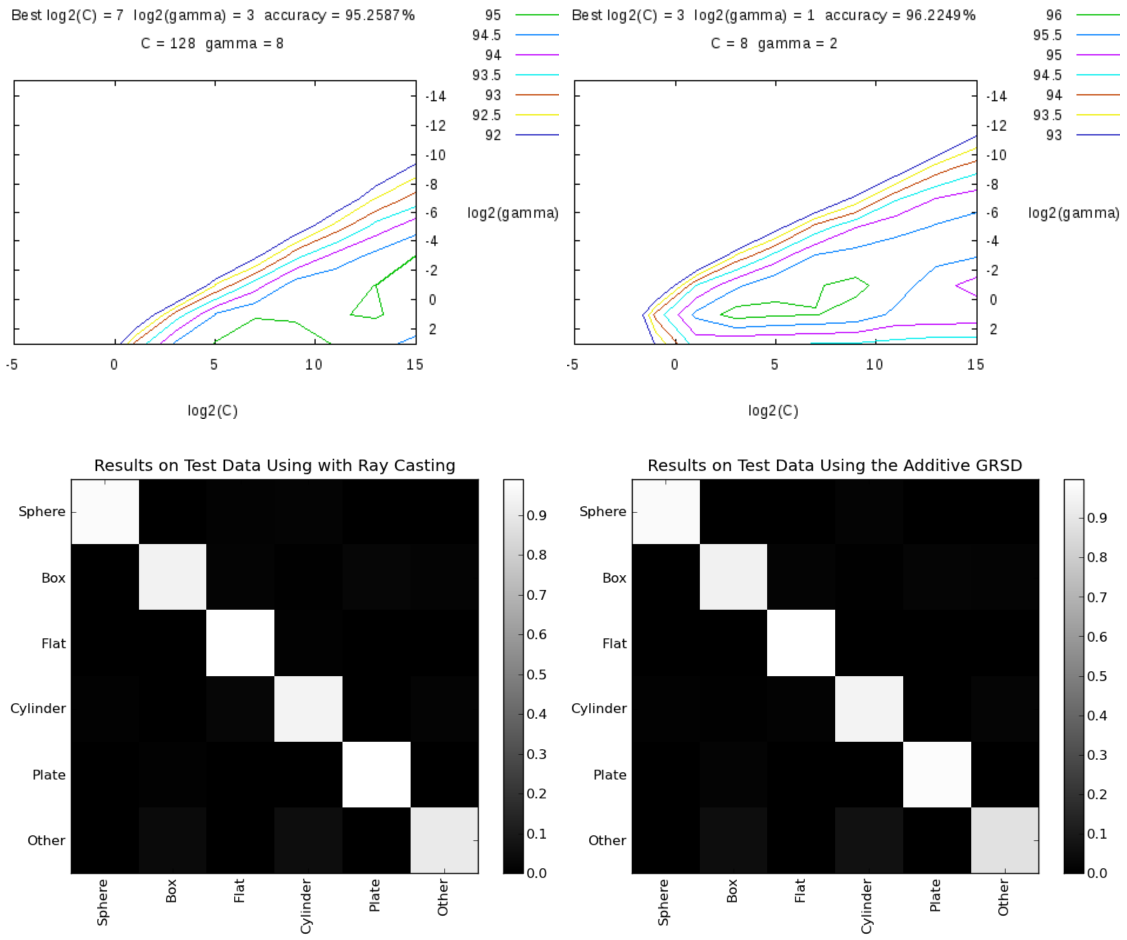


Figure 2.26: *Comparison of the raycasting GRSD (left) and the additive GRSD- (right) versions using a Support Vector Machines (SVM) classifier (Chang and Lin, 2001). Cross-validation results for different SVM parameters on the top show a slight increase in performance of GRSD- over the original GRSD, and the much lower C parameter value suggests less overfitting. The confusion matrices in the bottom reveal that the mistakes in categorization are very similar using the two features. However, GRSD- is more efficient to compute and has the additive property.*

Another additive feature that works similarly as GRSD- but captures color correlations is Circular Color Cubic Higher-order Local Auto-Correlation (C^3-HLAC). As presented in (Kanezaki et al., 2011a), it is rotation variant and artificial rotations need to be included during training. Additionally, the Linear Subspace Method exploits the additive property of the feature for classification based on partial views, but not the relations between the different parts of the objects. In (Kanezaki et al., 2011c) we introduced a rotation-invariant C^3-HLAC and its combination with GRSD-, named Voxelized Shape and Color Histogram (VOSCH). VOSCH outperformed, or performed similarly as C^3-HLAC and GRSD- for instance recognition tasks, but we will see that this does not hold for general category recognition.

2.5.3 Comparison of Features

In order to evaluate the GRSD- feature, unlike in the tests presented in (Kanezaki et al., 2011c), we are interested in categorization, not instance level recognition, a task at which geometric features excel more (Marton et al., 2014).

First, we compared GRSD- to standard features whose implementation is available in PCL. For this large scale test we used the complete RGBD Object Dataset (Lai et al., 2011a) and the methodology presented by the authors. We performed 10 categorization tasks using their 51 object categories, and reported the mean and standard deviation of the success rate in Table 2.2. The high-dimensional features we tested, i.e. VFH (Rusu et al., 2010) and the earlier Point Feature Histogram (PFH) (Rusu et al., 2008a), performed in the same range as reported by Lai *et al.* for geometric categorization, however using a simpler classifier.

This test is an extension of the experiments we performed in (Aldoma et al., 2012). We also evaluated a version of PFH that takes colors into account, called PFHRGB, to highlight the limitation of color-based features for categorization tasks.

Another way of capturing global information about a part, would be using a bag-of-features approach (Csurka et al., 2004). To evaluate this, we used K-Means to create a codebook of 50 RSD features. We sampled ten points of each object, again using K-Means, for classification, but could not match the results obtained by GRSD-.

[5] Building the search tree took approximately 12 hours for this metric, so it was evaluated only for a single partitioning of the dataset.

k-nn metric	VFH [308d]	GRSD- [20d]	PFH [125d]	PFHRGB [250d]	RSD [50d]
L1	56.44 ± 2.66	43.93 ± 2.20	53.00 ± 1.64	49.11 ± 1.73	37.98 ± 2.11
L2	52.75 ± 2.33	43.10 ± 2.20	51.31 ± 1.86	51.4 ± 2.41	35.19 ± 1.82
Hellinger	57.20 [5]	45.91 ± 2.00	55.95 ± 1.48	51.41 ± 2.41	38.25 ± 2.15
χ^2	56.3 ± 2.86	45.95 ± 1.99	55.63 ± 1.49	51.17 ± 2.33	38.16 ± 2.02

Table 2.2: *Cross-validation accuracy on the RGBD Object Dataset (51 classes).*

While the performance of GRSD- is somewhat less than that of VFH and PFH, thanks its low dimensionality, classification has very low time complexity as shown in Table 2.3.

k-nn metric	VFH [308d]	GRSD- [20d]	PFH [125d]	PFHRGB [250d]	RSD [50d]
L1	0.427	0.006	0.075	0.103	0.041
L2	0.078	0.001	0.011	0.015	0.007
Hellinger	0.671	0.004	0.061	0.073	0.048
χ^2	0.225	0.002	0.031	0.054	0.028

Table 2.3: *Per instance classification time in seconds by feature length.*

A second comparison, now involving the raytracing GRSD feature as well was done by training a multi-class Support Vector Machines (SVM) (Chang and Lin, 2001) classifier with an Radius Basis Function (RBF) kernel, with the best parameters found using grid search. Due to excessively long training and test times, tests were performed on the dataset presented in Table 5.2 (a subset of the (Lai et al., 2011a) dataset), and the evaluation procedure is also simplified. The data was split up for training and testing by using every third scan of an object for testing purposes and training the classifier on the rest of the scans. Nonetheless, it is sufficient to draw some conclusions from the results shown in Table 2.4.

Feature	Number of Dimensions	Accuracy [%]
GRSD	21d	95.21
GRSD-	20d	95.29
VFH	308d	97.35

Table 2.4: *Feature comparison using SVM.*

As we can see, the GRSD versions have less dimensions but they do not fall prohibitively behind the high dimensional VFH's performance. Additionally, they are not view dependent, and GRSD- offers the required additive property to easily compute the descriptors of grouped parts. View invariance is especially useful because datasets may contain different/unknown orientation (depending on sensor) and the acquisition angles can vary greatly in household robotic applications.

2.6 Learning Object Models from the Internet

To obtain the necessary "common sense" knowledge needed for performing complex tasks, using rich information from online sources is a promising approach (Waibel et al., 2011). Therefore, we expect the future World Wide Web to include a shared web for robots, in which they can retrieve data and information needed for accomplishing their tasks. Among many other information, this web will contain models of robots' environments and the objects therein. It is to be expected that a World Wide Web for robots will include thousands of object models encountered by the agents operating in our working and living environments, so enabling robots to use them, will bring them one step closer to being fully autonomous. Recognizing and handling objects and furniture pieces is a basic step for realistic manipulation activities, thus detecting the different object categories and localizing specific object instances is a fundamental issue.

There are already huge databases online that hold robotic perception relevant information, like the Trimble (Google) 3D Warehouse[6], product catalogs of online stores[7], or object model databases (like the aforementioned (Lai et al., 2011a) and the ones we reviewed in (Marton et al., 2011)). These can be used as training data for detecting similar objects in real sensor data acquired by the robot (Klank et al., 2009b; Zia et al., 2009), and can even be enhanced by real, environment specific data using *domain adaptation* (Lai and Fox, 2010). Such links to online structured object descriptions, knowledge bases and web stores allows the creation of more complete object descriptions and to reason on more than what is visible (backsides, storage temperature, weight, etc.) as in (Tenorth et al., 2011). By combining perception and knowledge reasoning to improve information gathered by the robotic agent, it is possible to treat perception as an integrated system of the autonomous agent, capa-

[6]http://sketchup.google.com/3dwarehouse/
[7]http://GermanDeli.com

ble of answering complex queries about its perceived environment (Pangercic et al., 2010).

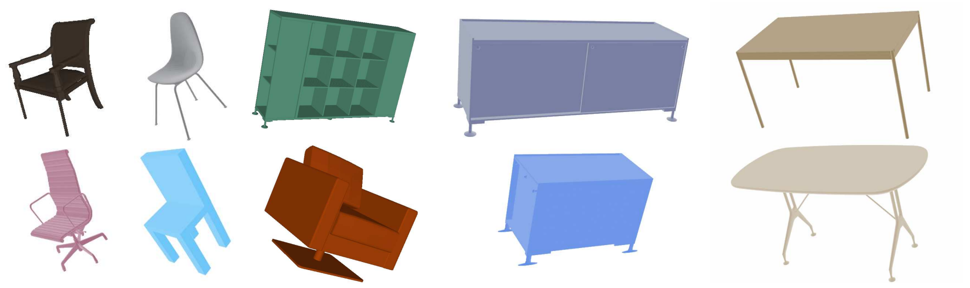


Figure 2.27: *In order to obtain training data for the different objects we want to detect, we downloaded several CAD models from databases like the Trimble (Google) 3D Warehouse, Vitra Furnishnet database for office equipment*[8]*, and the pCon.catalog from EasternGraphics*[9]*.*

We already created (Kanezaki et al., 2011c) and used such databases before as a resource for training methods dealing with object recognition in clutter, semantic mapping and environment adaptation (Mozos et al., 2011; Rusu et al., 2009d; Marton et al., 2013) as exemplified in Figures 2.27 and 2.28.

2.7 Multi-modal Part-based Classification

In order for household robots to become capable of localizing, recognizing, categorizing, and reconstructing the objects that are relevant for their manipulation tasks, they must be able to perceive the objects that are typical for kitchen environments. The materials that many kitchen objects are made of (including cups, glasses, plates, silverware, bottles, and boxes) present difficult challenges for state-of-the-art sensing technology: glasses and bottles are translucent and knives and forks are shiny. They are therefore challenging to detect. Also, silverware and the bottoms of plates are very flat and therefore difficult to discriminate from the table plane in noisy PCDs.

Because the perception task is too hard for any individual sensor type, we proposed the usage multiple sensors to recognize and analyze living environment scenes (Marton et al., 2009c). Here we will show, within a table setting scenario, that we can learn, classify and localize objects like cups, glasses, plates, silverware, bottles and boxes.

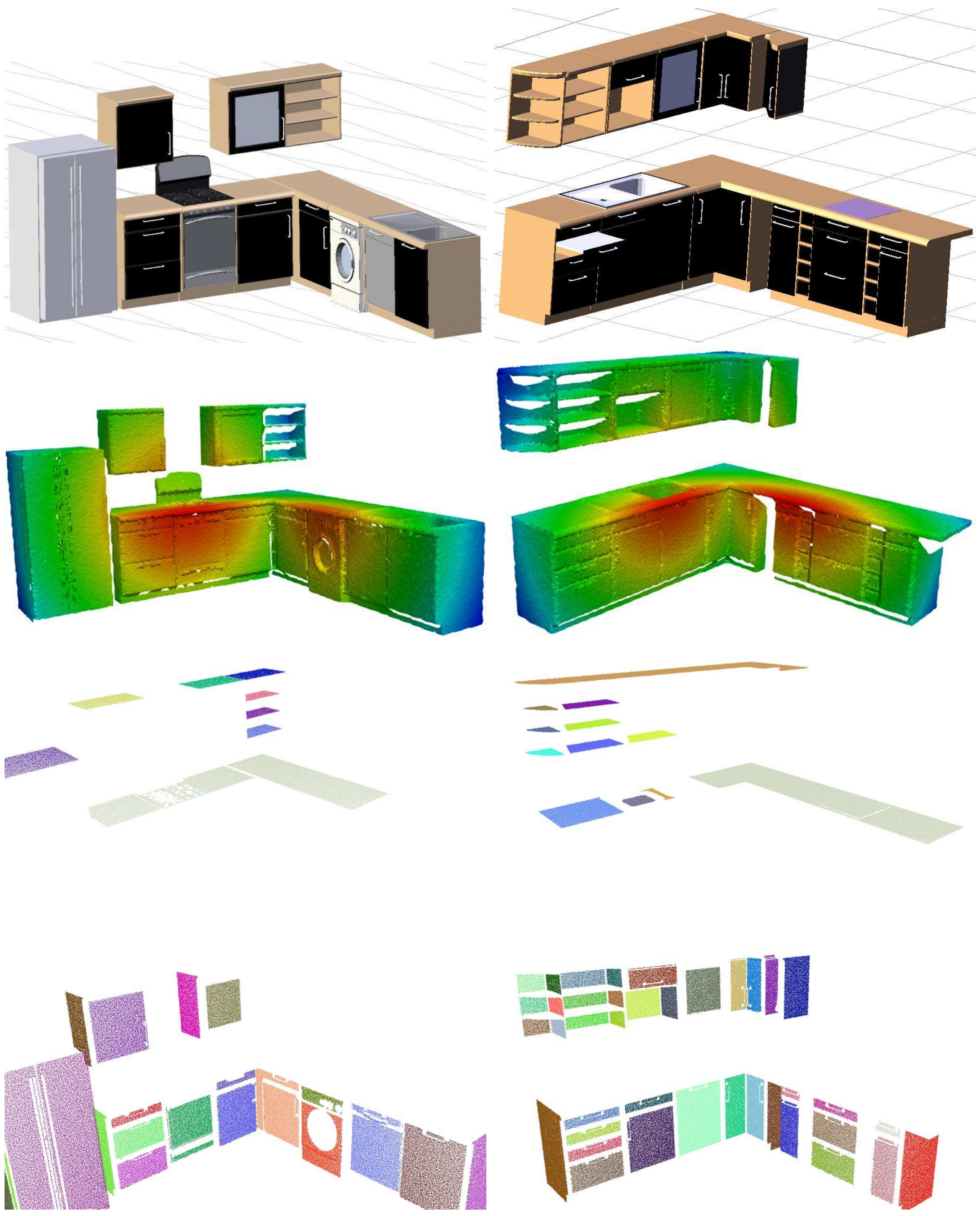


Figure 2.28: *Processed kitchen scans from Google Sketchup. From top to bottom: original model, simulated depth scan, automatically detected horizontal and vertical furniture faces. Since we have ground truth information for these scans, they can be used as training and evaluation data for the classification of container types.*

To perform these perceptual tasks, we use a composite sensor that includes a color stereo camera providing RGB images and VO, and a time-of-flight camera providing intensity, depth and confidence images, as illustrated in Figure 2.29. A 2D laser sensor is also present in the figure but it is not used for the purpose of the experiments presented in this section.

Each of the individual sensors cannot obtain enough information for the task of object categorization on its own. Stereo cameras, for example, cannot provide the necessary depth information for untextured planes, while glasses are transparent and not easily visible with any camera. However, combining the complementary perceptual evidence provided by the individual sensors gives us a much more valuable source of information.

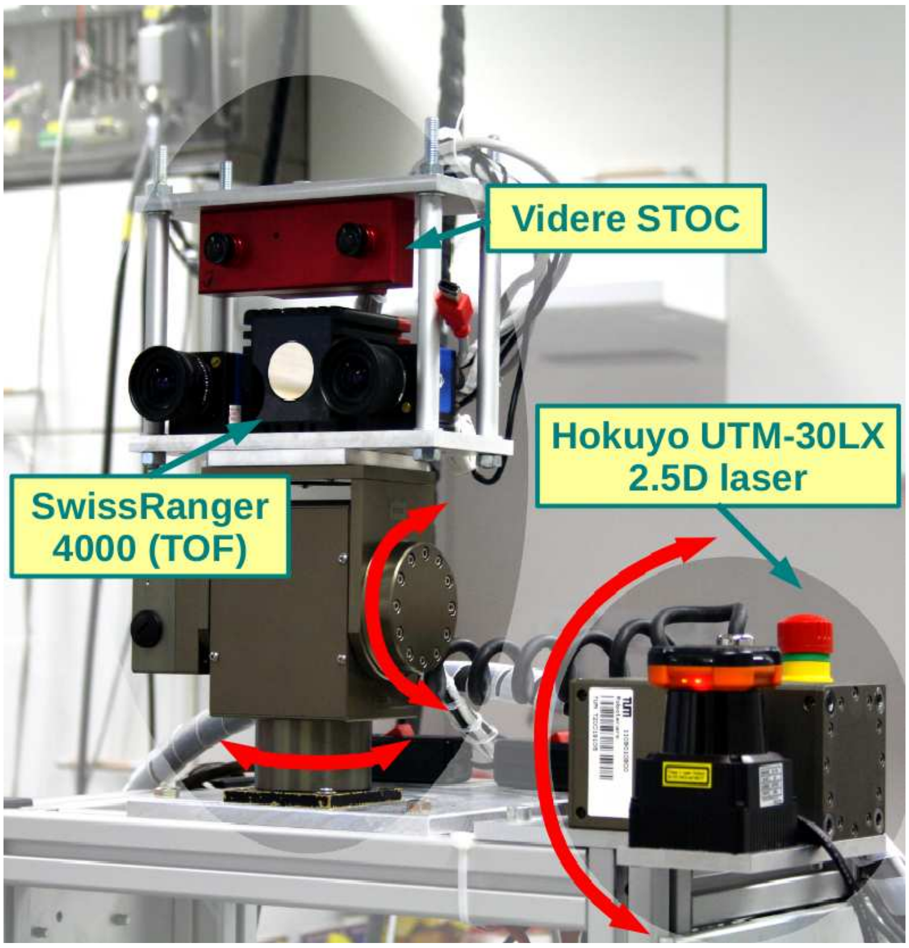


Figure 2.29: *Sensor mounted on a robot for capturing different visual modalities.*

The novel aspect of this approach is that using multiple sensors together with the proposed processing system enables robots to perform object categorization for a range of objects that are typical for domestic environments but represent challenges for state-of-the-art recognition methods. In particular, we are able to perceive and categorize transparent objects such as glasses and bottles, shiny objects such as knives and forks, and untextured objects.

Our work concentrates on combining multiple data sources from the used composite sensor, and extracting informative geometric and appearance-based features. These features are used in conjunction with a probabilistic Markov Logic Network which enforces contextual relationships between adjacent data, as developed and described by Dominik Jain (Marton et al., 2009c).

In this application we use the original RGB images for extracting the color information based features because of the bigger resolution, so we are interested in mapping the 3D information (the detected object hypotheses) onto the color images. Each cluster from 3D segmentation represents a region in the RGB images which can be extracted using the calibration parameters. From these regions we can extract image based features. Since we have structured objects with mainly round and straight edges, we employed an ellipsoid feature that is extracted using a method similar to the one proposed in (Hofhauser et al., 2008). This approach allows us to match any planar substructure of objects perspectively invariant.

The features from the other modalities, described in greater detail in the next subsection, will be combined with these for object categorization. In (Marton et al., 2010b) we combined the 3D and RGB-based modalities using a geometric classifier based on GRSD and a visual classifier based on SIFT, using a hierarchic classification approach. Here we treat the two modalities at the same level, and add another hierarchical level represented by relations between them.

In (Posner et al., 2008), the authors also propose a two-stage process to object recognition, where, at the local level, classification is based on appearance descriptors, and at the global scene level, Markov Random Fields (MRFs) are applied to model relationships. In this work, we take a similar approach in that we, too, consider a multi-stage process and leverage the promising combination of both 3D data and vision, where we first segment the point cloud data and process the corresponding image data, computing a rich set of features from it. In the literature, approaches that consider segmentation and classification at the same time – by defining an MRF directly over the scan points – have also been proposed (Anguelov et al., 2005), yet we believe that a hierarchical approach is preferable, not only from the perspective of computational efficiency. In (Triebel et al., 2006), the authors draw upon statistical relational learning methods, specifically associative Markov networks, to solve a similar problem – again, however, on laser data only and at a low level of abstraction.

2.7.1 3D Geometric and Intensity/Confidence Image Features

Unfortunately the TOF camera does not provide accurate enough 3D information for RSD/GRSD, so simple statistical features were used. On the other hand, the added modalities enabled us to segment various types of parts (see Section 2.4) and the confidence values also provide object relevant information.

Each part cluster representing an object candidate is analyzed based on the different sensor modalities. By analyzing the points in each cluster we can extract the features described below, based on the measurements of the time-of-flight camera. The features can be grouped into two categories: *attributes* (atomic values), and *relationships* (features computed by comparing parameters of the clusters' components). See the *feature extraction* and *object categorization* modules in Figure 2.1).

The list of attributes proposed in (Marton et al., 2009c) is as follows:

- *average normal angle*: the average angle of the estimated point normals (relative to the table's normal);
- *maximum and average height*: maximum and average distance of the points from the table's plane;
- *base area*: estimated area of the cluster's footprint on the table, based on occupancy of octree leaves;
- *volume*: estimated volume of cluster, based on the maximum heights above the cells in the base area;

- *average point density*: number of points in unit volume[10];
- *thickness, longness, wideness*: proportions of cluster along its principal directions;
- *points above and below the table*: percentage of points with positive / negative distance to the table's plane;
- *shadow and near points*: percentage of points marked with the *shadow* and *near* label;
- *zero and low confidence*: percentage of points which have 0 and respectively very low ($< 1\%$ of the maximum) confidence values assigned by the sensor [11];
- *number of components*: the number of *outside*, *near* and *shadow* connected components defined as detailed in the previous section.

The relationships which we defined between the components (if there are more than one components in the cluster):

- *above / below / same level*: the order of the components based on their centroid's height relative to the table;
- *front/behind*: the order of the components based on their distance from the viewpoint.

2.7.2 Categorization using Statistical Relational Learning

To evaluate our approach, we applied our system to the problem of object categorization in table-setting scenes. The types of objects include regular everyday kitchen objects, such as boxes of cereals, boxes of tea, different types of glasses and mugs, a few bottles, plates, and two types of silverware. The training data for the categorization was generated by placing two objects of the same type on the table, and creating

[10]An object which has shiny or transparent parts (like a bottle) will have a considerably lower density than the normal variations one gets by changing distances in indoor applications (especially considering the sensor limits too).

[11]The SR-4k grades each measurement accuracy with a value, 0 being the lowest confidence, typically assigned to points on metallic surfaces.

a series of views in a 180° arc of a circle, which were then processed through our geometric segmentation and feature estimation pipeline. A few examples of such scenes are presented in Figure 2.30, with the RGB images and the ellipsoidal features in the first row, and the 3D point clouds and the segmented clusters in the bottom row.

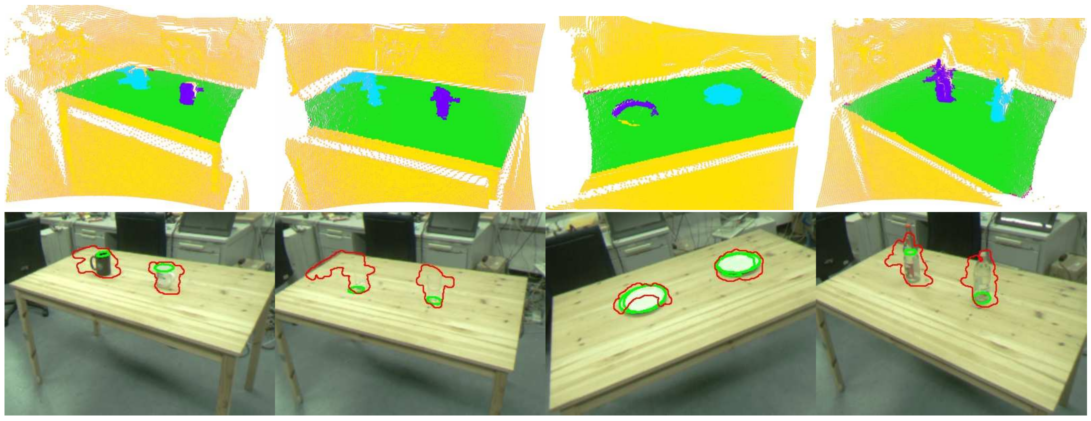


Figure 2.30: *Training samples for the classes mug, glass, plate and bottle. Please note that the glasses would be very hard to detect using camera images alone. The bottom row shows the RGB images with the 3D Segmentation projected and dilated with corresponding detected ellipses in green.*

We generated roughly 50 views for each object class which resulted in ≈ 80 training examples for bottles, boxes, glasses, mugs, plates and silverware. From some viewpoints, occlusions or measurement errors produced incorrect object clusters, leading to errors in the class label estimation. These cases were marked as *ambiguous*, and we trained the model by including a few of these cases, since they do occur in highly cluttered scenes like the one we used for testing. Please note that while we didn't train for all the special cases on how objects can occlude each other, the learned model still recognized some unseen examples of clustering failure.

Our training database contained 19224 facts on 570 objects that were comprised of 1851 spatially related components and featured 1308 ellipses. Training was completed in roughly 10 minutes. The learned model was applied to a more complex, partially cluttered scene (see Figures 2.3 and 1.3), which contained a mixture of new unseen objects with random objects picked from the training dataset.

Table 2.5 shows the classification results, which indicate an overall classification rate of about 54%, yet the accuracy on properly segmented objects is almost 70%. The

Class	Correct	Number	Ratio
1 - box	23	27	0.85
2 - plate	52	67	0.78
3 - glass	7	31	0.23
4 - mug	39	40	0.98
5 - bottle	4	5	0.80
6 - silver	21	43	0.57
7 - ambiguous	17	87	0.20
overall (1-7)	163	300	0.54
objects (1-6)	146	213	0.69
comparisons			
no relations between components	143	300	0.48
no 2D data	151	300	0.50
no 3D data	62	300	0.21
decision tree	148	300	0.49

Table 2.5: *Categorization results for all experiments performed.*

ground truth \ categorization	box	plate	glass	mug	bottle	silver	ambiguous
box	23	0	0	2	1	0	1
plate	0	52	0	0	0	5	10
glass	0	14	7	0	0	2	8
mug	0	0	0	39	1	0	0
bottle	1	0	0	0	4	0	0
silver	0	0	7	2	0	21	13
ambiguous	20	8	1	5	28	8	17

Table 2.6: *Confusion matrix for the results in Table 2.5.*

time taken for a run of the feature estimation and the classifier on a single scene was but a few seconds.

During the experiments we observed that the ellipsoidal features helped with the categorization of plates, and most of the mugs and glasses. Due to low contrast in the scene, the system extracted a few ellipsoid features on some of the spoons or several textured parts of the scene, which usually lead to false positives. However, because of the probabilistic nature of our learning scheme and the number of feature spaces we used, the model was not affected significantly. Additional features would also help distinguish the glasses better from other types of objects (plates in particular) and improve their classification rate.

The ambiguous category was also recognized only 20% of the time, but in some cases, our system recognized such ambiguous clusters instead as one of the object classes that do indeed appear within the cluster, even though the presence of clutter was not explicitly trained for. Such a case can be seen in Figure 1.3.

Finally, the results show that the fusion of sensory information is indeed helpful, since any model that considered only a subset of the information that was available failed to deliver equivalent results (see bottom part of Table 2.5). The following chapters will analyze this idea in greater detail.

2.8 Discussion

We have presented and motivated the use of the modules required by a perception system, in order to explore the scene, pre-process the data, detect objects, extract features from different sensor modalities, recognize and reconstruct objects for solving the problem of understanding scenes of different objects present in kitchen environments.

As we saw through the presented experiments, cue-integration at each level is a powerful source of robustness, be it surface reconstruction from multiple views, feature estimation through a hierarchy of point-, voxel-, patch- and relation-based features, or at the level of multiple sensing modalities. The next chapter will focus on designing the right interconnections between these modules to form a multi-cue perception system. This will be then used to address the problem of semantic modeling of the environment and its objects in cluttered scenarios, through the incorporation of spatio-temporal information and ensemble learning.

Chapter 3

Multi-cue Perception System for Spatio-temporal Integration

Painting is concerned with all the 10 attributes of sight; which are: Darkness, Light, Solidity and Colour, Form and Position, Distance and Propinquity, Motion and Rest. This little work of mine will be a tissue [of the studies] of these attributes, reminding the painter of the rules and methods by which he should use his art to imitate all the works of Nature which adorn the world.

LEONARDO DI SER PIERO DA VINCI
(OF THE 10 ATTRIBUTES OF THE EYE, ALL CONCERNED IN PAINTING)

This chapter investigates the required perceptual capabilities of personal robots that are to pick and place objects in the context of everyday manipulation tasks in domestic environments. Before presenting the technical details that make the perception system work, I would like to give a birds eye view of how it functions and combines various information sources to achieve its task.

Not only painters try to reproduce the visual world in an artificial medium, that is what should happen in the "head" of robots as well. The more realistic their internal representations (or 3D paintings, we could say) get, the better they can interpret and interact with the world. Therefore, the ten attributes of sight should be considered not only by those painters who aim for a verisimilar world presentation, but also by autonomous agents who only try to understand it.

Da Vinci wasn't a stranger to poetry either, and while he preferred painting, he was also able to write very capturing comparisons. He described the scattering of light by objects, and its importance to seeing as follows: *"Just as a stone flung into the water becomes the centre and cause of many circles, and as sound diffuses itself in circles in the air: so any object, placed in the luminous atmosphere, diffuses itself in circles, and fills the surrounding air with infinite images of itself."*

We saw already how to capture different aspects of objects from these images in the previous chapter, and that the more the robot knows about about an object the better it can recognize it. In this chapter the focus will be on strategies on how to combine the different sources of information from different timestamps and viewpoints, and on creating a system that integrates the presented methods.

3.1 Requirements of a Task-adapting Perception System

Let us start with analyzing what capabilities a multi-cue perception system should posses, and how the modules described in the previous chapter should interplay in order to achieve them. As we saw in the example application detailed in the previous chapter, we require the robot to achieve the following tasks: *finding* a certain object, *enumerating* everything at a certain location, and *modeling* objects for grasping either from scratch or by matching existing models. These require the robot to learn and create new representations and to fuse multiple sources of information for an accurate semantic interpretation.

Typically, research in the area of robotic perception focuses on the development of new or improved algorithms, solving a specific problem using a specif method. These approaches are valuable, and have come close to solving some of the perception problems like textured object detection, as shown in the Solutions in Perception Challenge[1]. However, it is unlikely that a single algorithm will perform sufficiently well in all the possible cases a household robot might encounter. In another machine learning application domain, the NetFlix Prize challenged researchers to predict user preferences based on information about their and their friend's ratings. As individual teams started to join efforts, ensemble learning became a popular, and very successful approach (Sill et al., 2009), similarly to the question answering scenario tackled by IBM's Watson system (Ferrucci et al., 2010). We expect a similar development

[1]http://opencv.willowgarage.com/wiki/SolutionsInPerceptionChallenge

in the perception field as well, as combining various approaches with complementary strengths can improve over the individual performance of the methods (Sewell, 2007; Ting and Witten, 1997) and make it generalize better (Lam and Suen, 1995).

The reason for success of these systems are not specific algorithms but rather that they solve the problems through ensembles of experts. According to Ferrucci et al. (2010), progress in the Watson project was mainly due to system-level advances which allowed rapid integration and evaluation of new ideas and components. As we mentioned in the introduction, data from both the real and the online world is mostly unstructured information for a machine, which is where approaches like that of IBM's Watson hold a great advantage and generality.

While there is large variance in the objects to be perceived by a household robot, in a typical environment, the objects (furniture pieces, items of daily use, etc.) do not change very often. Therefore, after an object was detected using a more general approach, *task and environment adaptation* (Horswill, 1995) can make the perception task less error-prone by using more specialized perception routines. For example, not all possible templates need to be matched for finding a bowl, or a color-based detector can be employed as well. If these methods fail, the robot can still fall back to the more general method, thus achieving an on-demand adaptation of process execution.

To achieve this, previously generated/reconstructed information (completed, registered CAD models, object position lifecycles etc.) can be reused as priors in future processing as well. Additionally, task adaptation requires to limit detectors to the ones that provide relevant information for the task at hand (Marton et al., 2011), and take into account for example intended object use (pouring, select storage location, etc.) for selecting the right approach. Similarly, context (task/domain/current room etc.) can shape object database selection, generate search regions of interest, select segmentation strategy, etc. as in (Aydemir et al., 2011).

As we saw in the previous chapter, we need multiple sensing modalities and data processing for being able to cope with all types of objects. Textured, textureless, 3-dimensional, flat, transparent objects, require different capabilities of the various sensors and features, some examples solutions including (Torres et al., 2010; Aldoma et al., 2012; Lysenkov et al., 2012). These can be combined into an ensemble of experts, and thanks to the synergistic effects produced by complementing functionalities, the resulting system can become more than the sum of the parts.

As a summary, it should allow bottom-up processing to extract primitive object annotations, then linking them with richer models for obtaining an improved representation (top-down). This combination of top-down and bottom-up processing is what is assumed to be done by humans as well (Frisby and Stone, 2010). Therefore, a perception system should:

- provide a common input-output defined for segmentation and detection methods,
- enable the specialization of each method to a subset of objects,
- support for consecutive or parallel methods to correct or support each-other,
- learn model representation from online sources or from experience to improve performance,
- incorporate information from multiple views to disambiguate complex cases.

Our proposed solution to integrate the approaches supported in the previous chapter for a modular, multi-cue perception system that takes advantage of the robot's exploration capabilities is exemplified in Figure 3.1. It builds on the lessons learned form the previously presented experiments, and on many discussions from people involved not only in perception, but also high-level planning, manipulation and knowledge engineering for example. A similar perception system was also described by Okada et al. (2007), where a particle filter based integration of multiple detectors and views was achieved, however only for searching for a given (known) object instance. As we will see in this and the following chapters, we applied our method for autonomously learning and modeling novel objects, be it container or appliance doors, furniture pieces, or (chiefly) objects in cluttered scenes.

The main idea is to obtain priors relating to the object from high-level planning, and providing poses and reconstructed models for grasping as an output. An ensemble of multi-modal and multi-view methods is used, followed by a geometric verification step. The following subsections give more details on the specific components of our multi-cue perception system.

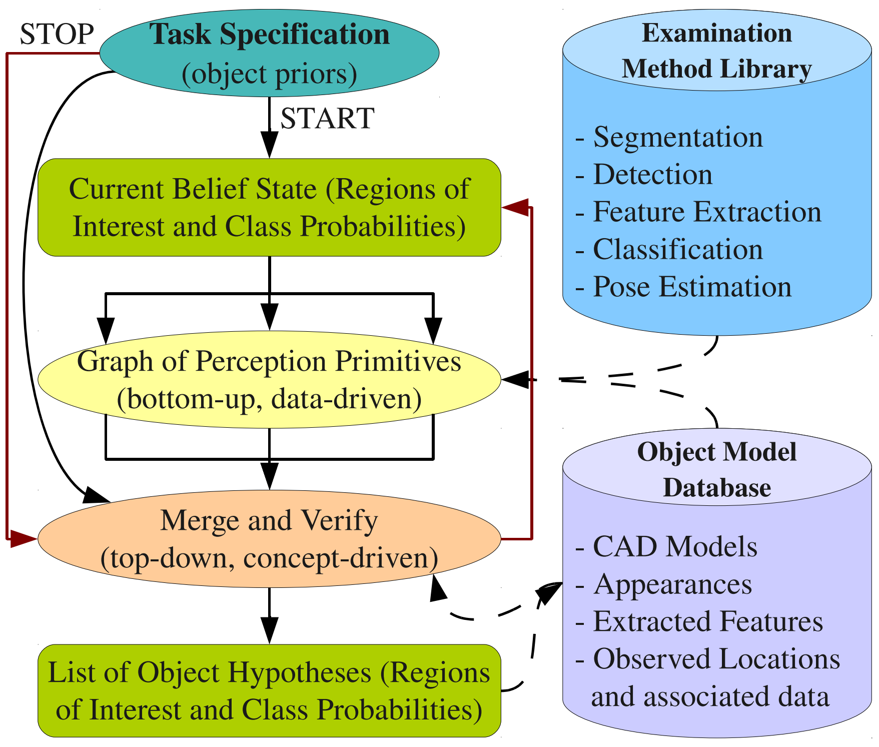

STOP
Task Specification
(object priors)
START
Current Belief State (Regions of Interest and Class Probabilities)
Graph of Perception Primitives
(bottom-up, data-driven)
Merge and Verify
(top-down, concept-driven)
List of Object Hypotheses (Regions of Interest and Class Probabilities)
Examination Method Library
- Segmentation
- Detection
- Feature Extraction
- Classification
- Pose Estimation
Object Model Database
- CAD Models
- Appearances
- Extracted Features
- Observed Locations and associated data

Figure 3.1: *Basic setup of the proposed system for iterative refinement of object hypotheses by multiple methods according to a task specification.*

3.1.1 Regions of Interest and Poses

Most related systems from literature are either doing segmentation or classification (or both at once), but in both cases a region of space is observed, and hypotheses are given about what objects, or parts of them, it contains. A segmentation routine for example breaks large regions up into smaller ones, and assigns to each of them a non-informative prior, i.e. from the point of view of the method each segment can contain anything. Subsequent processing (classification) steps then refine these possibilities. Template matching methods for example do both steps at once, by returning possible (scored) positions in which an object could be in the scene.

Therefore, we propose the use of volumes of space, or *regions of interest* (ROIs) as the basic input *and* output data for object perception methods. These can be for example the hulls of clusters for 3D data, or the estimated volumes of image pixels. These, and the associated probabilities of given objects being contained in it, are received and updated by the perception methods, and can be used to merge information coming from different sensors, and different views.

3.1.2 Integrating Results from Different Time-stamps

Since each examination method refines the result of its input, the ROIs are trimmed down (if necessary) and the class probabilities accumulated. In the *merging* step all the results can be united through ROI unification, and a decision can be made by an ensemble method. Subsequent sensor readings can be accumulated using the same procedure, and the object hypotheses and their poses can then be verified if they match the data as in (Mozos et al., 2011).

Accumulating or comparing object poses is more complicated, but a scored list of poses can also be maintained, and checked against the accumulated data in the given volume – similarly to (Drost et al., 2010). Merging the different votes for poses can be performed as in (Pham et al., 2011), which we successfully demonstrated recently in (Kriegel et al., 2013), but here we focus on object recognition.

3.1.3 Task-adapting Examination Methods

When performing perceptual examination, the system starts off with the complete workspace of the robot as its region of interest, with the different priors for the occurrence of the possible objects assigned to it. The list of these objects and their prior probability can then be considered by the different methods, and when summarizing their results.

We call all the segmentation, detection, fitting and classification methods *examination methods*, as they are the different modules the system is build of. They can use different sensors, extract various features, apply different recognition methods, and have only to respect the aforementioned input and output in order to be part of the "ensemble".

Classification methods such as those based on nearest neighbors, are easy to be retrained, and this allows simple integration of new data as well. However, with the addition of more and more classes, the accuracy can drop – this can be avoided by taking advantage of the known *task specification* (i.e. list of possible objects and list of sought objects). Similarly, the accuracy of other classifiers deteriorates with the increase in the number of classes (see Tables 3.1 and 3.2 created based on the dataset from Lai et al. (2011a)), something that can be alleviated by task and environment specialization.

Classifier	VFH	GRSD-	SURF	O.SURF	All
SVM-linear	0.081	0.270	0.154	0.163	0.0188
AdaBoost	0.087	0.293	0.254	0.202	0.0544

Table 3.1: *Error rate for 3D (VFH (Rusu et al., 2010) and GRSD-) and image based (SURF (Bay et al., 2008) and OpponentSURF (van de Sande et al., 2008)) feature descriptors, and their concatenation. Different classifiers were used for distinguishing 20 classes.*

Classifier	VFH	GRSD-	SURF	O.SURF	All
SVM-linear	0.133	0.409	0.281	0.301	0.031
AdaBoost	0.149	0.435	0.360	0.361	0.0991

Table 3.2: *Error rates in the same setup as Table 3.1, but for distinguishing 51 classes.*

Not all classes are as fast to be re-trained as nearest neighbors though, as shown in Figure 3.2, but methods like locally weighted logistic regression (Deng, 1997) could be used to avoid re-training by adjusting only the weighting of the examples.

3.1.4 Modular Classification Strategies

Modularity and redundancy are generally of benefit both to engineering solutions, and biological ones. This allows for specialization and adaptation in both domains (Noble, 2006), and is of great advantage in integrative fields like robotics. Identifying common characteristics of algorithms and defining common interfaces for them allows for simpler building of large systems, and facilitates easy integration of 3^{rd} party components. This has been a large contributing factor to the success of ROS and PCL.

The framework has a layered architecture consisting of *feature extraction tools*, *dataset tools* and the *classifier manager*.

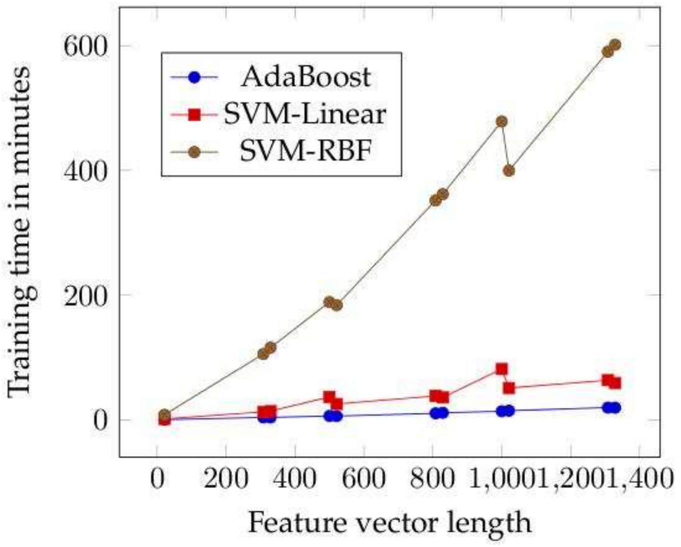


Figure 3.2: *Feature vector length vs. training time (using 7200 instances and 20 classes) on a standard desktop PC.*

The feature extraction tools are capable of walking directory structures and extracting various 3D and 2D features from the files encountered. Additionally it is labeling the instances according to the hierarchical relationships reflected by the directory structure of the dataset. This is especially important for the perception system to be able to import data from various databases found on the WWW and use it as an integral information source.

All the feature extraction tools use a file format to store the extracted features. The datasets produced in the *feature extraction* layer can be further processed by tools from the *dataset tools* layer. Here various operations are implemented, some of which are:

- serialization tools for loading/saving in ASCII or binary format,
- concatenation of features, combination of datasets and dropping of columns,
- re-labeling of hierarchical dataset to flatten them,
- splitting of datasets for creating training and test partitions (for example for cross-validation),

- bag-of-words (BoW) creation from a feature dataset (used for creating global descriptors out of local ones, like SURF),
- converting classifier confidences to datasets (used for stacking),
- pre-processing like dimensionality reduction, whitening.

The *classifier manager* is integrated in ROS to allow using the trained classifiers in ROS-based applications. It manages the life-cycle of classifier instances and acts as a facade for communicating with the classifier instances which must all be derived from a simple interface. Currently Support Vector Machines (Chang and Lin, 2001), K-Nearest-Neighbors (Muja and Lowe, 2009), Multi-Layer Perceptrons (Bishop, 1995) and one-against-all boosting approaches based on decision trees (Friedman et al., 1998) from OpenCV (Bradski, 2000) are implemented.

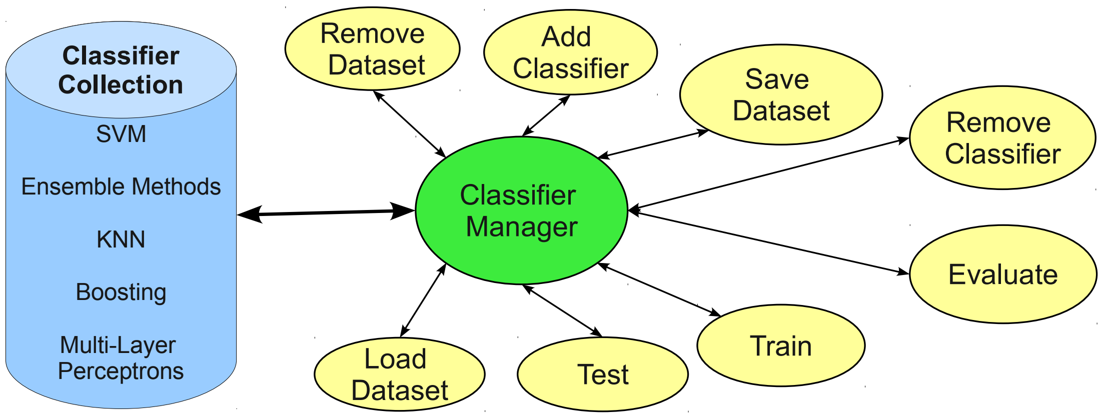


Figure 3.3: *Simplified scheme of the classification framework. The thin arrows represent possible service requests.*

The most common operations for classifiers implemented are shown in Figure 3.3. These are ROS services that are handled by a *classifier manager* node and each service can operate on different classifiers that run in parallel having a unique identifier. Other services that ease and make classification tasks more efficient include adding a common dataset to be shared by all classifiers, computing and storing a confusion matrix, returning various evaluation statistics (e.g. number of true positives) and timing of feature estimation and classification. All of these primitive operations form a simple domain specific language from which experiments can be constructed using shell scripts or ROS nodes, i.e. C++, Python, Java and Lisp code.

This classification interface made the creation of the large architecture needed for our multi-cue perception system possible, and enabled the seamless integration of various modules as we will see in the next two sections.

3.1.5 Conclusions

As we saw, there are specific challenges to be addressed by a multi-cue perception system, but as we will see in the next sections and the following chapters, it comes with substantial benefits for robotic applications. In order to detail and evaluate the implementation of our system, we will first examine the case of sequential examination method execution in a hierarchic setup (see next section), and then focus on running perception modules in parallel and fusing their results in Section 3.3.

3.2 Hierarchic Classification for Learning and Adapting Perception

Service robots, such as household robots, perform the same kinds of tasks with the same objects in the same environment over and over again. This enables them to learn and make use of more specific perception mechanisms for the particular objects and environments. Task and environment adaptation thereby enables the robot to better perceive the objects in its environment by exploiting its experience and considering only relevant objects.

One example of such an environment adaptation is that the a robot performing daily chores in a specific kitchen environment exploits the fact that it knows almost all objects in its environment. It can then use this assumption in order to specialize its perception mechanisms. It might turn out that cups are the only yellow objects in the environment. Thus in order to find the cups it suffices to look for yellow objects. The advantage of such specialized perception mechanisms is that they are often more reliable, more accurate, and more resource efficient than their general counter parts. However, as the knowledge of all objects is only an assumption and will often be violated as new objects are acquired and old ones thrown away, the robot must be prepared that its perception routines might not work as intended and adapt its perceptual apparatus in a lifelong learning process.

As discussed in Section 2.4 and (Marton et al., 2011), we consider that perceptions tasks can be intuitively categorized into *active* or *passive*. In the first type of task, a specific object or a set of objects needs to be located and thus the robot has to actively search for them. In the second case, the robot, possibly while performing other tasks, observes the environment or a specific region and identifies the different objects. For example, while setting the table for breakfast, the robot might spot a bottle of wine somewhere. Later, when the robot is to set a table for a lunch that includes a wine beverage, it does not need to actively start searching for a bottle, but instead retrieves the bottle's location from memory and navigates directly to it.

In both types of perception tasks, some sort of model is needed that describes what constitutes a certain object or an object of a certain type. Since we are most interested in pick-and-place scenarios like completing a table setting (which encompasses both *active* and *passive* perception), models that aid the robot in grasping are of great importance. While identifying objects based on their appearance works reasonably well (Kragic and Vincze, 2009), a 3D model is required in order to manipulate these objects and it has to be matched to the observed model. Also, as concluded in (Kragic and Vincze, 2009), a problem of current vision systems is robustness and scalability.

Our system uses general perception methods to initialize learning, adapt and specialize their strategies as much as possible to the environment and objects it has to use. By this, some of the problems become easier to deal with, thus improve efficiency. The proposed object perception system maintains an *Object Model Database* that provides information for the detection, recognition, pose estimation and grasping of the objects in the environment, and show how can use and build such a database under some assumptions. Our approach is to start with general detection routines and categorization modules to limit the possibilities for identities of objects and then add more and more details until an acceptable solution can be found (Marton et al., 2011).

Though the set of objects of daily use that a personal robot could encounter in its tasks is unlimited, there are certain regularities that can be exploited with respect to the objects' shapes, textures and locations. Therefore, the perception system can adapt itself to a specific set of objects that are usually present in the world while at the same time retaining a certain degree of flexibility with respect to the incorporation of novel objects in its database of models. For example, a new flavor of iced tea should be recognized and manipulated as an instance of a box from its geometry, even though the robot has never seen it before. The knowledge about the iced tea can then be easily enriched with its appearance, provided it can be re-detected using geometric information.

Also, certain information can be incorporated from external sources, for example a database of semantically labeled visual appearances of objects, which lacks the geometric appearance. In this case, the system should complete the missing information whenever possible. Similarly, in the case of autonomously learned object models, the semantic label or the correction of the assumed object identities could be done externally. To this end, our system can learn the specific models of the objects it encounters. These models can then be used to decide on highly specialized detection strategies, that give accurate results as long as the underlying assumptions hold. This enables the system to scale well with the number of objects in the environment. The accumulated data about each object can be used for the accurate verification of its hypothesized detection, and the detection strategy can be adapted accordingly.

As a high-level task specification is typically available, the system can take advantage of the fact that only some objects are probable to be at different places in the close surrounding of the robot, and of these ones, only some are relevant for the task at hand. Different perception methods can then be activated (or tuned) and combined, in order to improve detection rates.

Since we do not assume that every perceived object in the environment is known, the system can mistakenly assume, for example, that a certain feature is unique to an object when in fact it is not. Our view is, however, that these mistakes are unavoidable during autonomous object learning; they are compensated by the ability to automatically acquire models.

As an example, let us consider the case where the robot is to fetch a green object. Since this object happens to be the only green one it already knows about, it looks for the first green object it finds while heading towards the last observed position of the object. While it drives around, it observes various other objects that do not match the feature (color) it is looking for. But since navigating around and taking laser scans takes longer than identifying the objects (scans with adequate resolution take in the range of seconds), the robot is constantly executing an "identify-all-objects" routine. The latter updates the stored time-stamped position of recognized objects and assigns all unrecognized ones to new entries, assuming they are objects that were previously unknown to it. As a way to keep track of the detections, and to maintain the right associations between different viewing angles, we are using a simplified version of the probabilistic framework for identity resolution developed by Blodow et al. (2010).

The kinds of limitations of the system (that would require human supervision at some point) are as follows:

- previously unseen objects could be erroneously classified – this effect can, however, be minimized to the combined false classification rate of the detectors if the specialization step uses all of them;
- previously unseen faces of object could lead to the classification of a novel object if in the previous step it was not observed in the same location or identified correctly – this is unavoidable, but new views of known objects can be identified, given continuous observations;
- known objects can be miss-classified for example in cases where objects have identically textured faces (some cereal boxes for example) – this issue also occurs in re-detection-based systems, but if the object is continuously observed, and one of the ambiguous faces is not the first one that is seen, misclassification can be avoided.

3.2.1 System Overview

Figure 3.4 shows the schematic representation of the multi-cue perception system integrated into the whole robot control architecture (Marton et al., 2011). The system obtains the task from high level planning and executes it, performing the necessary processing steps on the data coming from the sensors of the robot used for localization and perception of the environment. These include the possible *Object Detection Methods*, the selection of modules from the *Examination Method Library*, and object recognition and learning using the information stored in the *Object Model Database*.

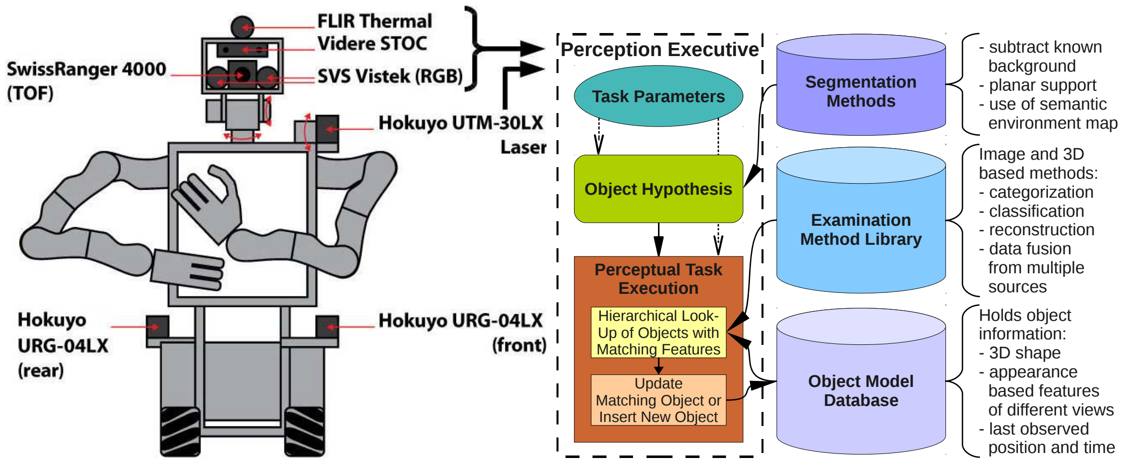

FLIR Thermal
Videre STOC
SVS Vistek (RGB)
SwissRanger 4000 (TOF)
Hokuyo UTM-30LX Laser
Hokuyo URG-04LX (rear)
Hokuyo URG-04LX (front)
Perception Executive
Task Parameters
Object Hypothesis
Perceptual Task Execution
Hierarchical Look-Up of Objects with Matching Features
Update Matching Object or Insert New Object
Segmentation Methods
Examination Method Library
Object Model Database
- subtract known background
- planar support
- use of semantic environment map
Image and 3D based methods:
- categorization
- classification
- reconstruction
- data fusion from multiple sources
Holds object information:
- 3D shape
- appearance based features of different views
- last observed position and time

Figure 3.4: *Object processing pipeline 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 (please see Figure 3.5 and the previous chapter) 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.

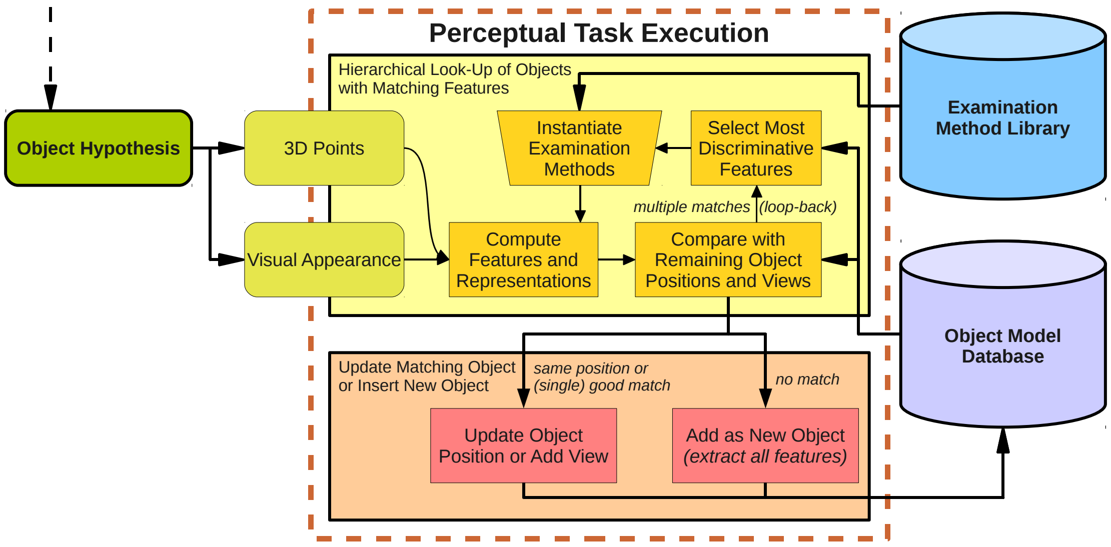

Perceptual Task Execution
Hierarchical Look-Up of Objects with Matching Features
Object Hypothesis
3D Points
Visual Appearance
Instantiate Examination Methods
Select Most Discriminative Features
multiple matches
(loop-back)
Compute Features and Representations
Compare with Remaining Object Positions and Views
Examination Method Library
Object Model Database
Update Matching Object or Insert New Object
same position or (single) good match
no match
Update Object Position or Add View
Add as New Object (extract all features)

Figure 3.5: *Details of the* Perceptual Task Execution *module's operation. Selecting and combining methods from the* Examination Method Library *to recognize known objects (either based on their stored features, or by co-locality in subsequent cycles), and to update their position or add their new view. The loop-back is not needed if a single object is being searched for. In case new objects were found, all features and representations are computed from the available data in order to be inserted into the* Object Model Database.

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. Note that this is the case only if the classification step does not assign each observation to at most one modeled object.

Since the computation of the features for an object hypothesis is not too expensive, as shown in the corresponding sections, the aim of the procedure is to 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.

The following sections detail the functioning principles of the building blocks of our system as shown in Figure 3.4.

Our system is developed within the ROS open-source framework as a collection of modules (so called “nodes”), that each tackle a specific sub-problem, and can either be run separately or as dynamically loadable libraries (i.e. “plugins”). Thus, the data flow between nodes can either occur through shared memory, when optimal performance is required, or over the network, which is convenient for decentralized processing.

The use of ROS and general examination methods based on PCL enables us to be platform independent in the sense that the system can be run on different robots, and results can be interchanged between them. So far, we have successfully performed our experiments on two different robots, TUM-Rosie (Pangercic et al., 2010; Marton et al., 2010a; Blodow et al., 2010) and the PR2 (Wyrobek et al., 2008; Bohren et al., 2011).

Currently we employ the following methods to achieve the overall classification:

- auxiliary modules (noise removal, *k*d-tree, octree – see below, and the next chapter for registration and segmentation);
- local feature estimation (see previous chapter): Moving Least Squares (MLS) and Radius-based Surface Descriptor (RSD);
- global (view-based) features: Global RSD (GRSD) feature and Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) feature using Vocabulary Trees (Marton et al., 2011);
- building geometric models: box/cylinder/rotational model fitting and triangulation (see previous chapter);
- geometric categorization using Support Vector Machines (SVM) and instance level classification using SIFT descriptors with vocabulary trees.

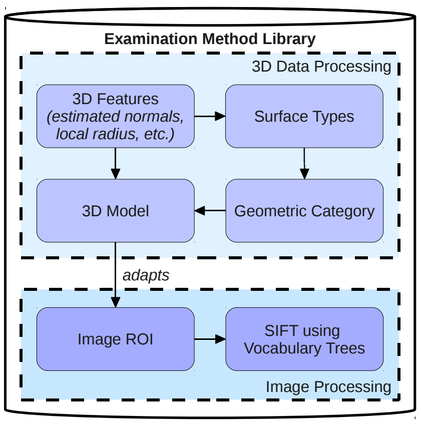

Examination Method Library
3D Data Processing
3D Features (estimated normals, local radius, etc.)
Surface Types
3D Model
Geometric Category
adapts
Image ROI
SIFT using Vocabulary Trees
Image Processing

Figure 3.6: *The conceptual representation of the complete pipeline that can be formed by the examination methods.*

To increase the performance of the 3D data processing steps, we rely on noise removal (Rusu et al., 2008c), and efficient spatial decompositions such as kd-trees and octrees. Noise removal is performed to remove jump edges in the laser scans and to remove points from noisy or low density areas of object clusters. We use kd-trees to find a fixed number of neighbors or all the neighbors in a sphere with a given radius, while octrees are useful for down-sampling and occupancy grids, but also for the verification of the geometric models (Blodow et al., 2009), and for computing object-level features efficiently, as described in the previous chapter.

Examination methods and their possible connections are shown in Figure 3.6. Each method has a common interface, with initializing, processing, and result validating and returning functions, that enable the easy extension with additional methods if needed. Each method lists the data type it requires and provides, and the parameters are obtained through the parameter server of ROS.

To facilitate object recognition we maintain a database of the descriptors provided by the examination methods. It is important to note that not all descriptors must be known due to the multi-modal nature and incremental learning feature of our system. The view-dependent variations of the descriptors are stored for every view, thus each view of an object that has been observed can be looked up and compared with object hypotheses during classification. The examination methods described above provide the following data on a per-view basis: the geometric category, the appearance descriptors, the model parameters and the 3D mesh. The timestamped position of the last detection is saved for each object as well.

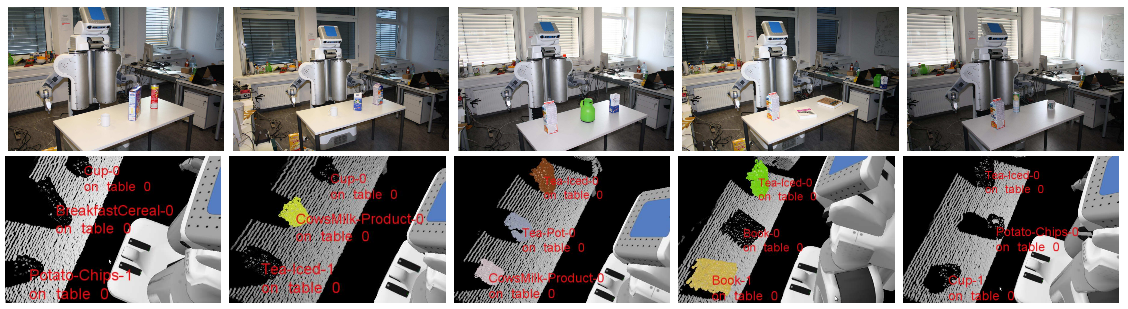
The following subsections detail the object identification and model learning modules using the above presented subsystems. We have to differentiate between the two operating modes, *active* perception, when a given object needs to be located, and *passive* perception of objects for updating their locations and models. Since the *Object Model Database* does not have to contain every possible information about the objects, it is straightforward to look for a novel object by adding a new entry that contains known information about the object.

3.2.2 Object Classification

For classification, we are iterating over the examination methods (Figure 3.6), run them on the object hypotheses to extract features (if possible), and search through the list of available objects in the database to see which ones present the same or similar features. Those that match are kept in the list for the next iteration of comparison, and the others are removed. This way, at each step the set of possible objects gets reduced. As discussed before, if multiple objects remain after checking all features, the result is deemed ambiguous, whereas if at a given step no objects match, we consider the object hypothesis to be a novel object.

Due to the combination of features, an object candidate is accepted only if it matches all of the checked features. This reduces the effect of miss-classifications, at the cost of observed views of known objects presenting previously unobserved features being added as new objects (unless observed in subsequent scans). As the preceding process is started only if the object was not identified in the previous step (otherwise the previous identity is assumed), new views can be learned through continuous observation of an object as presented in 3.2.4.

Figure 3.7 shows results we obtained with a batch of 12 testing objects. The tests show correct classifications, achieved in under 2 seconds (not considering acquisition time). Please see the next subsections for a more detailed evaluation.



(a) Cup-0
BreakfastCereal-0
Potato-Chips-1

(b) Cup-0
CowsMilk-Product-0
Tea-Iced-1

(c) Tea-Iced-0
Tea-Pot-0
CowsMilk-Product-0

(d) Tea-Iced-0
Book-0
Book-1

(e) Tea-Iced-0
Potato-Chips-0
Cup-1

Figure 3.7: *Classification results for a variety of objects. Clusters shown in random colors.*

3.2.3 Detection based on Known Views

In order to evaluate our approach as a whole, we performed the detection and recognition test in our kitchen laboratory (see left column of Figure 3.8). The test was carried out on a total number of 13 objects located at 4 different scenes.

The robot navigated to each of the scenes and captured point clouds and images from several different views by moving along the collision free paths around the scenes. The environment map aided object detection, as for the scenes 2 and 4 the supporting planes are too high, thus impossible to scan with either of our robots (see Section 2.4).

The texture based detection obtained the correct label with 83% accuracy and geometric classification achieved just below 80% success rate (see Marton et al. (2011) for details).

There were five SIFT-based classification failures for Scene 2, due to the reflective metallic surface of a green chips cylinder which made the classification using SIFT descriptors practically impossible in that position. And this is where the power of our multi-modal system becomes apparent. It was in all 6 measurements of this object for this scene correctly categorized as a tube-like cylinder, which for the set of our 13 objects, made the *Perceptual Task Execution* to decide for the instance of green chips cylinder – the only tall cylindrical object that was currently in the *Object Model*

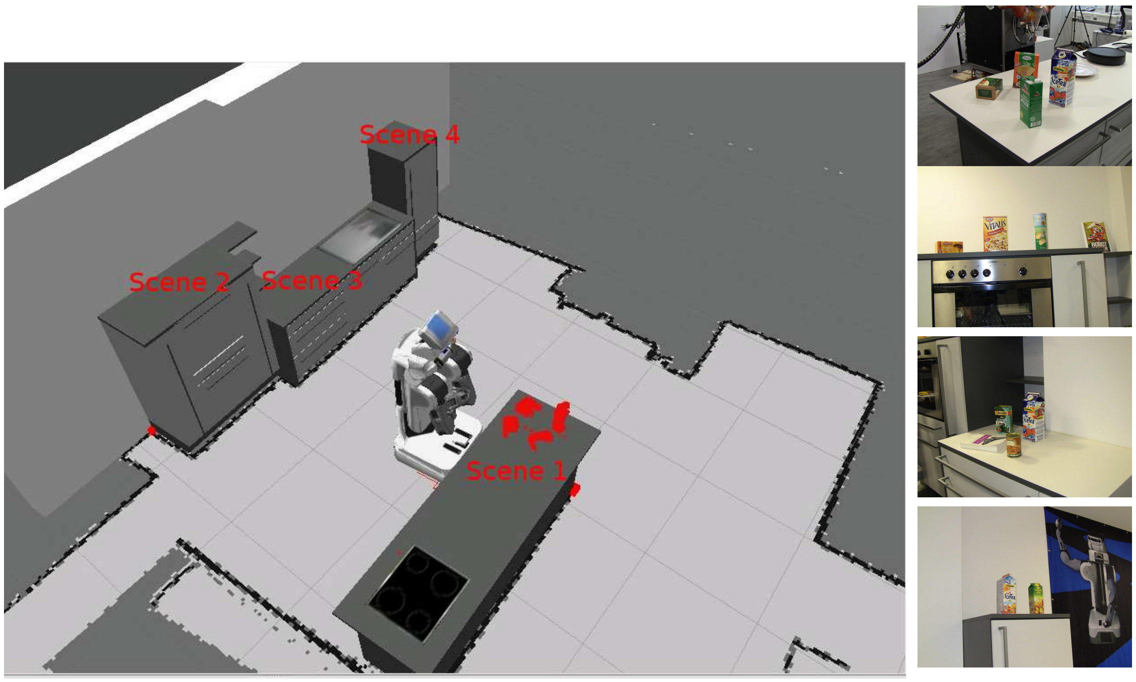


Figure 3.8: *We performed our tests on a total number of 13 objects located at 4 different scenes in our kitchen lab (denoted with Scene 1 ... Scene 4 and depicted in top-down order in the right column).*

Database. In a database with a larger number of objects containing e.g. multiple tall cylinders correct geometric categorization would still provide a significant hint at the possible class instance. Additionally, the geometric model reconstruction provides the means for eventual manipulation.

For a large-scale evaluation of the hierarchic classification strategy, we have performed several experiments on around 400 partial views of 29 objects (shown in Figure 3.9). We used data taken using a turntable for training, and additional views of table scenes for testing (examples in Figure 3.10). Please see (Marton et al., 2010b) for more details.

To evaluate the overall performance of our approach we carried out three types of object recognition test: (*i*) test with different splits of the training dataset (around 300 views), (*ii*) test with 76 views of the objects in regular table settings, and (*iii*) test with 15 previously unseen objects/views (i.e. geometric classification alone) on the table. While the best accuracy (95.45%, see bottom of Figure 3.10) was obtained in

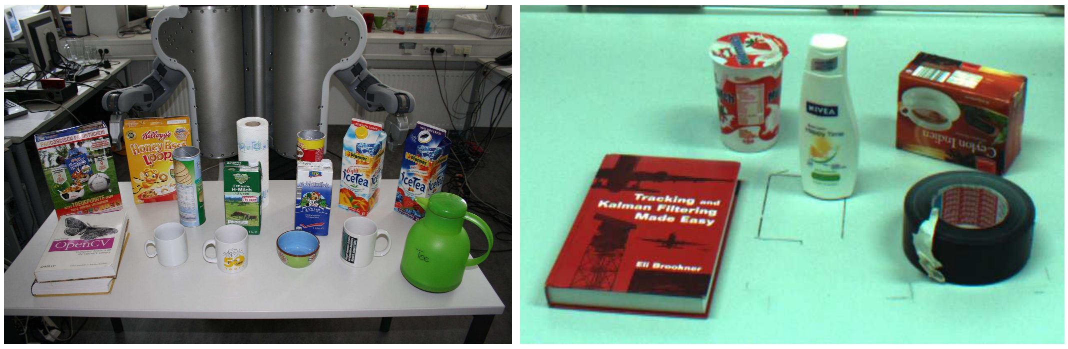


Figure 3.9: *Left: some of training objects. Right: some novel objects for test* (*iii*).

the first case, the accuracies for the remaining tests (85.53% and 80% respectively) were still encouraging.

For the full evaluation of the image-based recognition module developed by Dejan Pangercic we kindly refer the reader to (Marton et al., 2011).

3.2.4 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 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 examination methods returns no matching views for the given object hypothesis, we store the current observation as a new view of the object in the *Object Model Database*, as shown in Figure 3.11.

To demonstrate the capability of our system to acquire new object models on the fly we set up Scene 1 with one unknown object (green milk box), which of course could not be identified. Knowing that its views don't 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 ROI at the given 3D position in the world coordinate frame contain the same object.

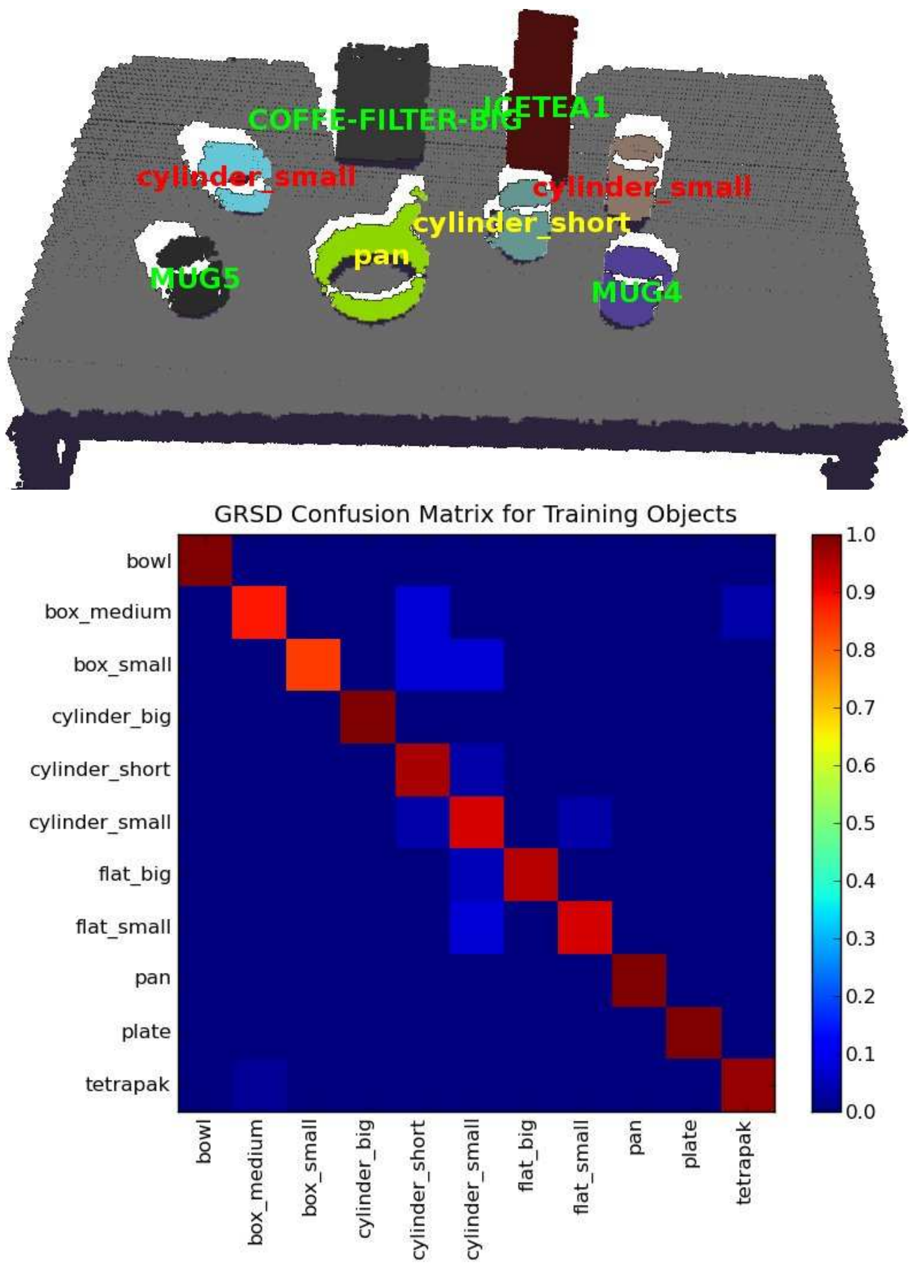


Figure 3.10: *The table scene at the top shows hierarchic classification results of test* (ii). *Red labels denote geometric miss-classification, yellow ones correct geometric category but incorrect visual classification, while green labels signal the correct object identification. The confusion matrix in the bottom provides the true and false positive normalized GRSD statistics for the test with training objects.*

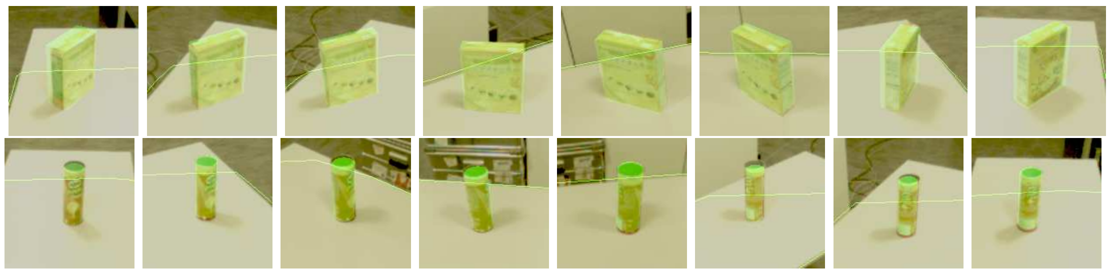


Figure 3.11: *Different views of a cereal box (top) and chips (bottom) learned by multiple observations in the same position. The green line represents the table's edge as detected by the laser (some parts of it were not scanned), and the green markers denote the projections of the fitted 3D model onto the image data.*

Scene 1	**#Views/#Known**	**#Failures**	**#Unknown**	**Success** (with/no unknowns)
Before	52/42	0	10	**80.8%**/100%
After	52/52	2	0	**96.2%**/96.2%

Table 3.3: *Improved detection after learning the new object for Scene 1 from Figure 3.8. All in all, more views got the correct label.*

After this we performed another test run on the Scene 1, and were able to reduce the number of not detected objects down to 2 as shown in Table 3.3. Subsequent detections of this type of milk box thus have a better chance of being recognized. Since most large databases (e.g. GeranDeli.com) offer only single pictures of objects, incremental learning is an important feature for a perception system that needs to develop over time.

3.2.5 Conclusions

In this section, we presented a system for automatic and efficient object detection and modeling that can improve its models with limited human supervision. Multi-modal methods (both data and model driven) are combined for building and extending an open-ended object model database by learning novel objects and new information on known objects online. The high number and variety of objects in a typical household require the use of multiple descriptive features and the automatic construction of geometric models for performing robotics tasks autonomously, as we believe that having an object database built by the robot itself in the operating environment (or adapting/enriching a seed database) is more suitable to deal with the specifics of the environment efficiently than a generic approach of providing as many models of as many objects as possible.

3.3 Ensemble Learning for Fusing Parallel Processing Results

So far we considered a hierarchic execution of the examination methods, but this way errors at a higher level get propagated, without a way for the lower levels to correct it. Therefore, instead of a sequential execution, this section will focus on merging the results of parallel perception modules, using the method we developed in (Marton et al., 2012b).

As we discussed, these object classification tasks can be made significantly more robust if multiple sources of information are used. Since each perception method captures only some aspect of the objects, the situation is similar to the old story about the six blind men trying to describe an elephant based on a single touch. Clearly, a correct combination of different sensor modalities for classification would improve results.

Combining multiple sensor modalities to improve detection can be done in general either by combining multiple features in a single classification pipeline, or by separate processing pipelines for each modality whose results are combined. The former approach is pursued by Lai et al. (2011b), where a combination of visual and depth cues is used. We explored the latter approach in the previous section, highlighting the limitations of the different sensors, and exploiting that not all features need to be checked if there is a subset of them that uniquely describes the object (Marton et al., 2011).

While both approaches use different features from different sensing modalities, each of them has inherent drawbacks. Combining different features by concatenation produces very high-dimensional features that produce excessively long training (and possibly classification) times. Additionally, any change in the used features requires a complete re-training of the whole classifier. On the other hand, it is a simple way of obtaining a very accurate classifier that considers all the useful information. The sequential method is very modular, and some features can be left out if needed (e.g.

those from camera images in low light conditions) or new ones added without the need to re-train the whole system, but the errors get accumulated from each step, and correlations can not be exploited. Ideally, one would like to have a separate classifier for each feature, evaluate what it is good at and what mistakes it makes, and then produce a final classification by considering the results of all the individual ones.

This is exactly the idea behind ensemble methods in machine learning and pattern recognition, which combine a number of different models, trained for the same task, in such a way that the performance of the ensemble surpasses the performance of the best of its members. In this work we add another requirement to the ensemble, namely to outperform the classifier that uses the concatenation of the features, thus to bring the best properties of the two approaches together for robotic object recognition.

During our evaluations we found that it is quite difficult to surpass the results obtained by concatenating all the features, but that it is possible with the right training strategy and ensemble method. More specifically, we explored a novel technique for creating uncorrelated ensemble members that is substantially different from previously used methods. The technique consists of training strong learners on subsets of a set of features extracted from different sensor modalities and to combine them using either simple heuristics or stacking. This is in sharp contrast to many other ensemble methods which combine weak learners on a indivisible set of features by perturbing, for example, the data distribution by re-sampling or re-weighting.

We also gave special consideration to ensemble methods that do not need to be trained together. While this approach produces slightly inferior results, its modularity makes it especially advantageous for large systems integrating multiple sensors, where (with the improvements in hardware and addition of new sensors) novel features are being developed that need to be integrated.

Furthermore, in the area of autonomous robotics, the ability to direct gaze only with certain sensors or changing environmental conditions can have the effect that some of the sensing modalities can not be used for observing an object (e.g. low light conditions for cameras, or objects with specular, transparent or reflective surfaces for 3D sensors), thus online modularity is an additional plus.

In this section we will present the ensemble methods we implemented and evaluated to find an alternative to concatenating features. As Figure 3.12 summarizes, the key idea is to create separate classifiers for different features and to combine their output for the final classification.

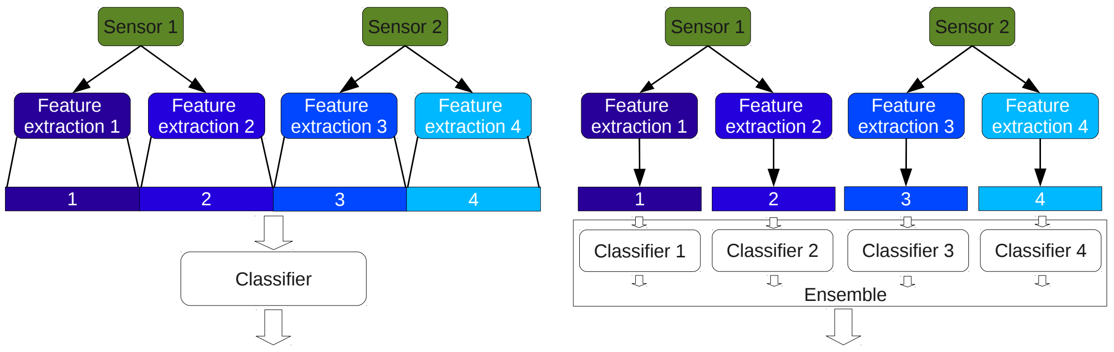


Figure 3.12: *Technique of concatenation (left) compared to ensembles (right).*

Fumera and Roli (2005) analyzed linear combinations of classifiers and found that the performance depends on the correlation of the outputs of the ensemble members. This is intuitively clear, since if two classifiers often make the same prediction little additional information is gained by observing the output of both. This is obviously also true for all other ensemble methods.

A lot of the best performing ensemble methods have some scheme for re-weighting or re-sampling the dataset for training the ensemble members to achieve this lack of correlation. One example is bagging, which stands for bootstrap aggregation, in which a random subset of the dataset is used for training each member. Another method is the popular AdaBoost in which the classifiers are trained in sequence and the training data is re-weighted for training a new classifier in the sequence. When using neural networks the random initialization of the weights leads to different local minima and it is then possible to combine these networks to committees.

In the next subsections we present the various ensemble methods we used grouped into four categories, starting with the simpler "max rules" and finishing with stacking.

The accuracy of the classifiers are estimated from a validation partition of the dataset. In some of the rules it is quite likely to get a tie between two decisions. The ties are broken by making a random decision. To keep things simple this is not stated explicitly for each rule for which it applies.

3.3.1 Max and Product Rules

The max accuracy rule is to choose the classifier whose classification decision has the highest class-conditional classification accuracy. Similar to this, the max confidence rule is to choose the classifier which has the highest confidence in it's decision. The combined max confidence and accuracy rule is to choose the classifier whose classification decision has the highest class conditional classification accuracy multiplied by the classifiers confidence. These are simple ways of combining classifiers that do not require a common training and consider the result of only one selected classifier. Another well known method we explore that does not need optimization of parameters but does merge all the available information is the product rule.

3.3.2 Weighted Voting

The weighted votes method assigns an additional weight to every classifier. We experimented with various voting approaches that do not require combined training and are thus modular. In accuracy weighted voting each classifiers vote counts weighted with the accuracy of the classifier. Confidence weighted voting is equivalent to simple voting with the addition of having a vote count with the classifiers confidence. Confidence and accuracy rated voting is equivalent to simple voting with the addition of having a vote count with the classifiers confidence multiplied with it's class conditional accuracy.

3.3.3 Error Correlation

For regression problems it is possible to find a closed form solution for the weights of a linearly combined ensemble of classifiers if the expected sum of squares loss is to be minimized. Bishop (2006) analyzes this approach and a closed form solution is given and shows that for all convex loss functions this method's expected performance is higher than simple averaging and can not produce an increase in the expected error. He discusses the topic in the context of regression but it is also possible to use this approach for classifier ensembles. The derivation goes through unchanged for the classification case.

3.3.4 Stacking

Stacking or stacked generalization as it was originally called by Wolpert (1992) is an ensemble method in which classifiers are stacked to achieve improved classification performance. Stacking classifiers is done by collecting the outputs of so called level-0 classifiers into a new dataset and to train one or more level-1 classifiers on the outputs of the level- 0 classifiers. When doing this it is of importance to build the dataset on which the level-1 classifiers will be trained from training data that wasn't used to to train the level-0 classifiers, since what we are interested in is to improve the performance on the test set, not on the training set, in short to improve generalization.

This is usually done by separating the dataset into n folds, removing one, training one of the classifiers on the remaining folds and evaluating the classifier on the fold which was left out. This is repeated for all the folds. Together with the labels this process creates the level-1 training dataset, which has the same size as the level-0 training set. Formally, the level-0 training set D_0 is split up into $m = |E|$ equally sized folds $P_1, .., P_m$. By P_{-i} we denote $D \backslash P_i$. Each $e_i \in E$ is trained on P_{-i} and evaluated on P_i to yield a part of D_1, the level-1 training set. Then the level-1 classifier is trained on D_1. The accuracy of the stack is evaluated on a separate test dataset. Stacking is illustrated schematically in Figure 3.13.

The procedure described above is the classical formulation. We take a slightly different approach which is motivated by the abundance of data we have, the multi-modal nature of our features and the goal to easily try feature combinations without having to retrain a classifier on the combined feature vector every time. Let $D_1, ..., D_N$ be N different datasets obtained from N different feature extraction processes. The datasets are split up into three parts each (D_i^0 to D_i^2), the level-0 training sets, level-0 evaluation sets and level-1 evaluation sets. The classifiers $e_i \in E$ with $|E| = N$ are trained on D_i^0 and evaluated on D_i^1 to produce D_i' the level-1 training-set parts which

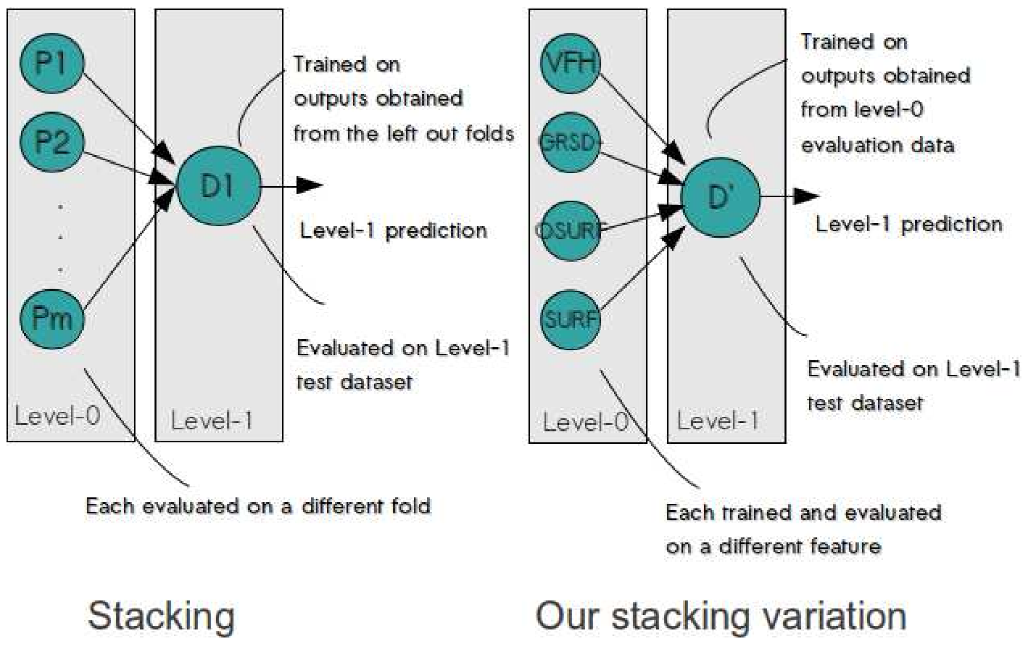

P1
P2
Pm
Level-0
D1
Level-1
Trained on outputs obtained from the left out folds
Level-1 prediction
Evaluated on Level-1 test dataset
Each evaluated on a different fold
Stacking
VFH
GRSD
OSURF
SURF
Level-0
D'
Level-1
Trained on outputs obtained from level-0 evaluation data
Level-1 prediction
Evaluated on Level-1 test dataset
Each trained and evaluated on a different feature
Our stacking variation

Figure 3.13: *Original stacking compared to our stacking version.*

are combined to form D' the full level-1 training-set on which a level-1 classifier is then trained. The whole stack is then evaluated on the D_i^2 datasets.

So far we only spoke of the “outputs” of the classifiers without defining them more precisely. Ting and Witten (1997) evaluated stacking on four different classifiers and their combinations and found that for stacking to work the level-0 classifiers have to output a confidence rating for each label. Since our AdaBoost multi-class version outputs confidence ratings we were able to use it as a level-0 classifier. The SVM implementation we use is also capable of training a probability model, unfortunately the training process is very time consuming and so we were not able to try SVMs as level-0 classifiers.

3.3.5 Experimental Results

The following subsections present the results we got during the evaluation of the presented method. The results will be discussed in the next section.

3.3.5.1 Dataset preparation and feature extraction

The full dataset presented in (Lai et al., 2011a) contains roughly 250,000 scans of the 300 objects organized in 51 categories[2]. This is a lot of data and handling datasets of this size poses a challenge in itself. Therefore, like the original authors as well, we decided on using only every 5th sample in the dataset. This still leaves us with about 1 day of extraction time for the whole dataset, but it assures that enough views of the objects are considered in order to make a proper evaluation.

After extracting the VFH, GRSD-, SURF and OpponentSURF features, the Bag of Words (BoW) model is used to create a global descriptor for the objects when using OpenCV features. Then the datasets were split up further by removing every second instance to obtain the training partitions of the dataset. Of the removed instances every second was used to form a validation set for the level-0 classifiers and the remaining instances were used to form a validation set for the level-1 classifiers. Table 3.4 contains the number of instances for each partition of the datasets.

We use PCL to extract the VFH and GRSD- features. The point clouds in the dataset are very dense so to speed up normal estimation the point clouds were down-sampled using a voxel grid filter with 0.001 meters grid size. The local neighborhood for estimating the normals is determined using a 0.02 meter radius search. In our experiments we use the OpenCV 2.2 implementation of SURF. We detect features using the dynamic SURF detector, an extension which lowers the detection threshold until a sufficient number of points have been found. We also scaled each dimension of the VFH and GRSD- features to the range $[0, 1]$ by subtracting the minimum value along a dimension and dividing by the maximum along that dimension (determined after subtracting the minimum). The SURF BoW feature vectors were linearly scaled so make the bins of the histogram sum to one.

As a next step the datasets were concatenated to yield six datasets containing the concatenations of each group of two features, four datasets containing the combinations of each group of three features, and one combination containing concatenations of all four features. We call the single features (VFH, GRSD-, SURF and OpponentSURF) “singles”, the concatenations of two features “doubles”, of three features “triples”, and of all features “all” in the following.

[2]http://www.cs.washington.edu/rgbd-dataset/

Table 3.4: *Number of instances in the datasets containing 20 classes (left) and all of the 51 classes (right).*

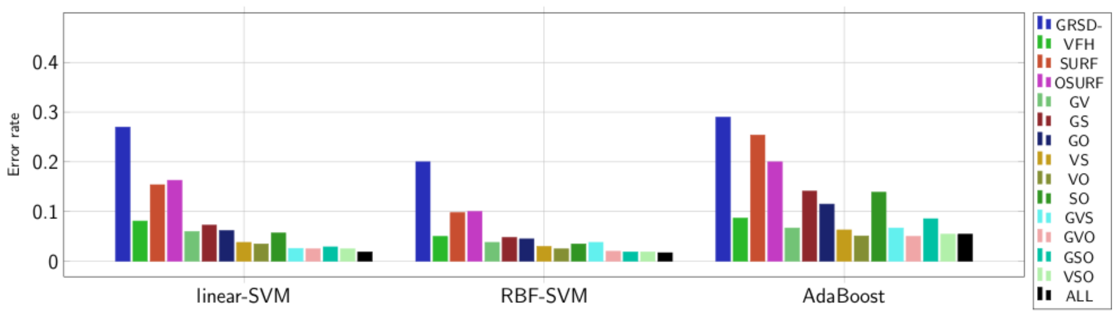
Number of classes	20
Training instances	3600
# eval level-0	1800
# eval level-1	1800
Total scans	7200

Number of classes	51
Training instances	20738
# eval level-0	10369
# eval level-1	10369
Total scans	41476

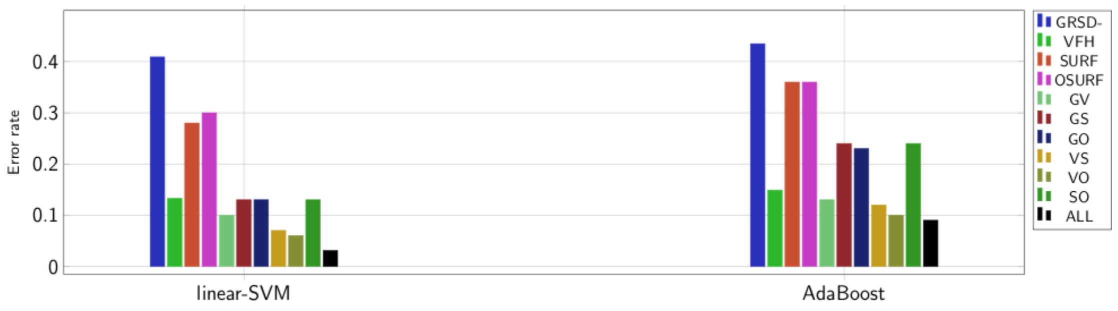
Next the comparisons of results obtained by the different classifiers and ensembles are presented, considering the simple features as well as their combinations. As we will discuss later, outperforming the classifier trained on the concatenation of all the features requires the use of pairwise combinations even with the most complicated ensemble method. Nonetheless, this solution leaves us with a relatively good modularity, as the cost of adding a new feature to the list of level-0 classifiers is linear in the number existing ones (training the pairwise combinations), and only the level-1 classifier has to be re-trained.

3.3.5.2 Evaluation of Features and Classifiers

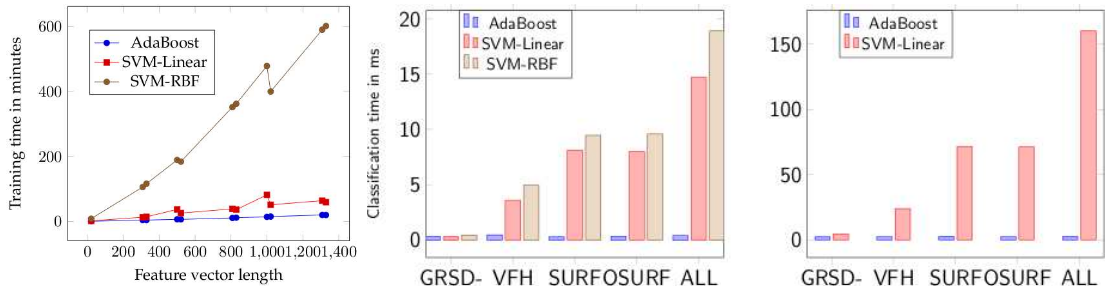
Throughout the experiments we will present performance using the error rate, i.e. the fraction of object scans to which a false label was assigned. As shown in Figure 3.14 AdaBoost is not very good with the concatenations and/or is very bad with the BoW model. Regarding the SVM classifier, the linear kernel is as good as the RBF kernel for high dimensional features. This implies that AdaBoost is inferior to SVM. For the 51 class problem the accuracy is roughly halved for both methods, but SVM scales marginally better. On the other hand, as it can be seen from the classification times, the AdaBoost classifier can be trained in less time and the time for classification with AdaBoost is independent of the feature length. These findings were to be expected given the generally known behaviors of the used classifiers. Therefore, for some tests, using SVM with RBF kernel was attempted, but was found to be prohibitively time-consuming and was left out.



(a) Error rate for the 20 classes problem



(b) Error rate for the 51 classes problem



(c) Feature vector length vs. training and classification time (for 20 classes)

Figure 3.14: *Results using AdaBoost compared to those using SVM.*

3.3.5.3 Simple Ensemble Methods

We tested the simple ensemble methods described in Section 3.3 using several feature-classifier combinations, as shown in Figures 3.15 and 3.16. The provided error rate for the concatenation and the best member being the error rates achieved with the respective classifier. Due to time constraints we did not train a probability model for the SVM and consequently all the ensemble methods involving confidences were left out.

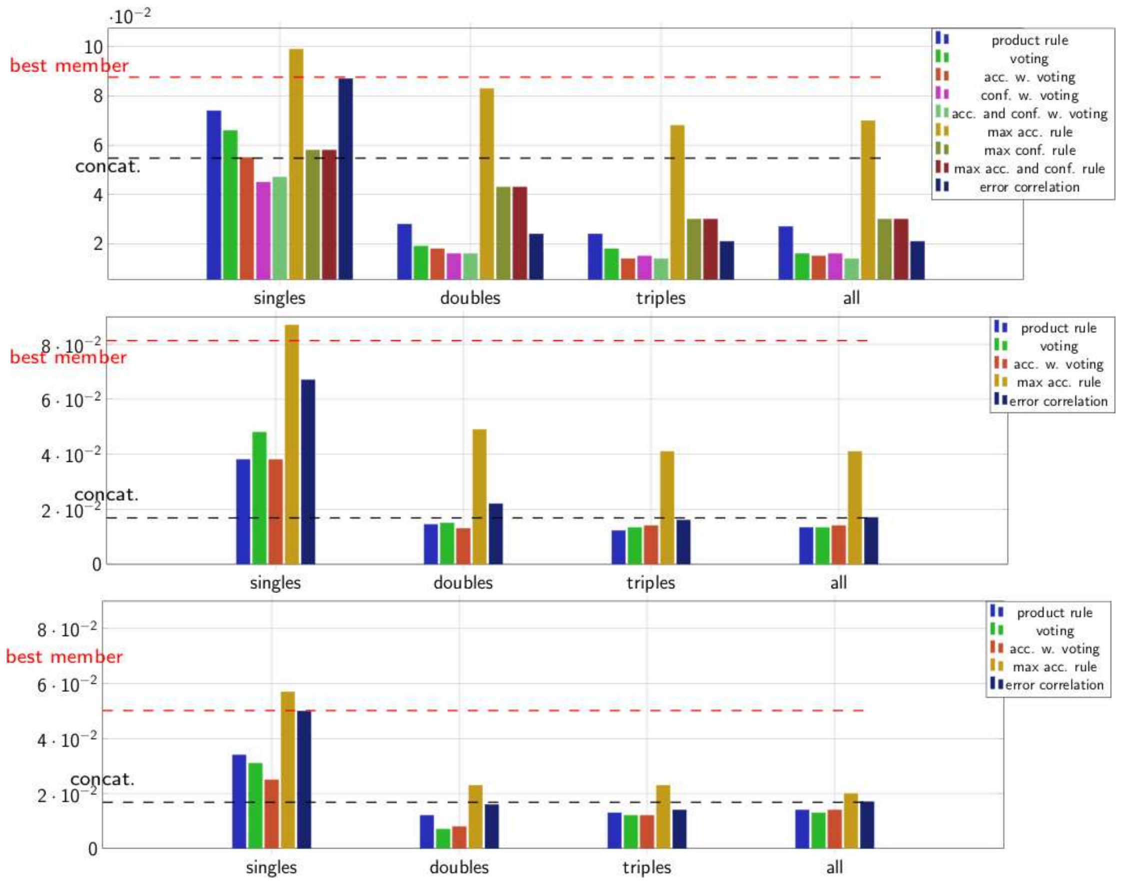


Figure 3.15: *Error rate of all simple methods on the 20 classes problem with (from top to bottom) AdaBoost, linear SVM and SVM with RBF kernel as base classifier.*

3.3.5.4 Stacking Compared to Simple Ensembles

In this experiment we evaluate performance of our variation of stacking as described in subsection 3.3.4. We used Real AdaBoost as level-0 classifiers and Real AdaBoost, LogitBoost and Gentle Boost as well as a linear SVM and a SVM with Radial Basis Function kernel as level-1 classifiers. As for the simple ensemble methods we combine level-0 classifiers trained on the singles, doubles and triples for the 20 classes problem. For the 51 classes problem we combined the singles and doubles only due to excessively long run times. The results are shown in Figure 3.17, with the provided “concat.” error rate as the best error rate achieved using the concatenation approach.

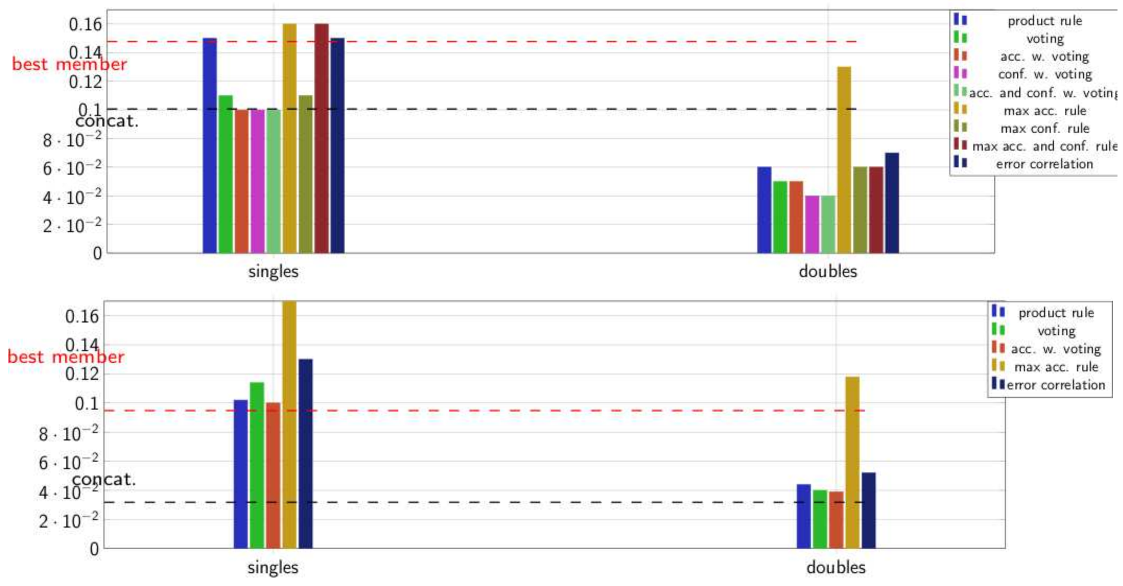


Figure 3.16: *Error rate of all simple methods on the 51 classes problem with AdaBoost (top) and linear SVM (bottom) as base classifier*

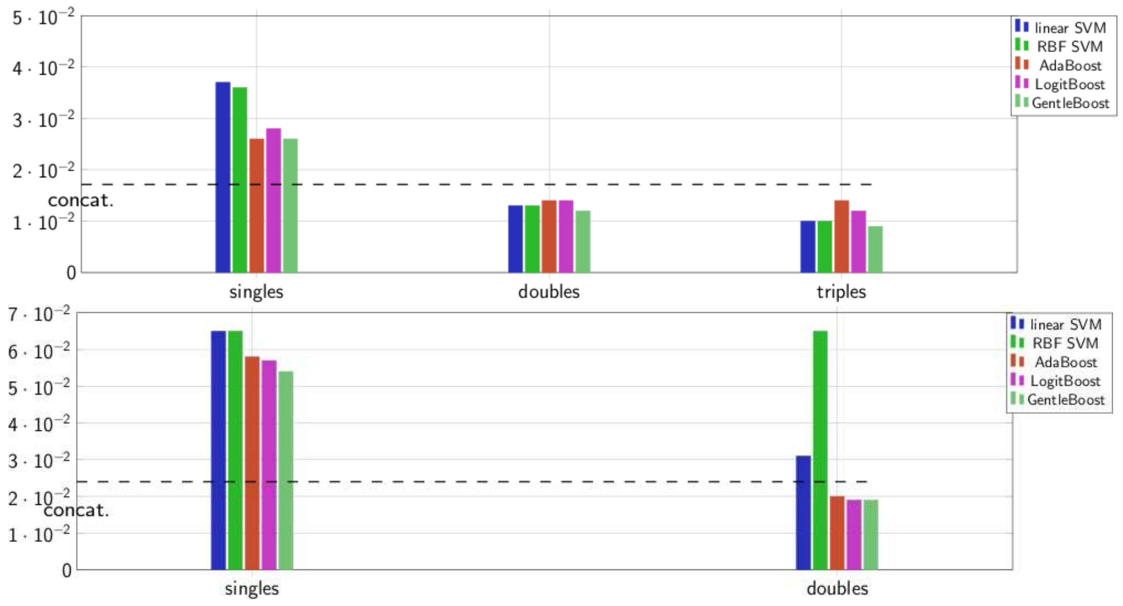


Figure 3.17: *Stacking with AdaBoost as level-0 classifier and various level-1 classifiers, for 20 classes (top) and 51 classes (bottom).*

3.3.6 Conclusions

To establish a baseline to which we can compare the performance of the ensemble methods we evaluated a small number of classifiers (linear-SVM, RBF-SVM and AdaBoost) on a number of features, which capture different aspects of the objects, and their concatenations. From the experiments we conclude that VFH is the most descriptive feature, among the features considered, for the task at hand. Not surprisingly, the accuracy of all the classifiers and the time required for training increases with the number of features used. Furthermore, the direct comparison of the classifiers shows, that the SVMs can utilize the information contained in the diverse features better than AdaBoost and also that the linear-SVM approaches the RBF-SVMs accuracy for concatenations of features of increasing length – as suggested also in (Chang and Lin, 2001). Regarding the training time, we see an approximately linear increase with feature vector length for all classifiers.

In every ensemble method the goal is to produce a number of uncorrelated classifiers and the experiments described in the previous section provide ample evidence to suggest that our feature recombination method leads to uncorrelated classifiers. Most noteworthy is the significant increase in accuracy obtained over the single feature ensembles by the ensemble whose members were trained on two element subsets of the feature set. In virtually all cases for the 20 classes problem this setup was able to improve even on the classifiers trained on the concatenation of all the features, and for the 51 classes problem at least the stacking approaches were able to achieve this goal. However, adding further classifiers trained on larger subsets of the feature set to the ensemble did not lead to much improvement in most cases and to decreased accuracy in some cases. As shown in Figure 3.18 the results of the concatenated features were almost reached already by stacking the single original features.

One important advantage of the simple rule ensembles over monolythic classifiers and stacking is their modularity which, as we could show, comes at the cost of decreased accuracy. The hybrid model of the simple rule ensembles trained on two element subsets, however, is an approach balancing the conflicting goals of accuracy and modularity while incurring only an insignificant increase in the time needed for training. It therefore presents an attractive addition to the machine learning toolchest for applications which can take advantage of multiple data sources. In particular, accuracy and confidence based voting approaches seem to be the method of choice as they deliver the highest accuracy among the simple rules.

All in all, we can say that multi-modal and multi-featured object classification has a clear advantage over single-featured one. In contrast to our earlier work where a se-

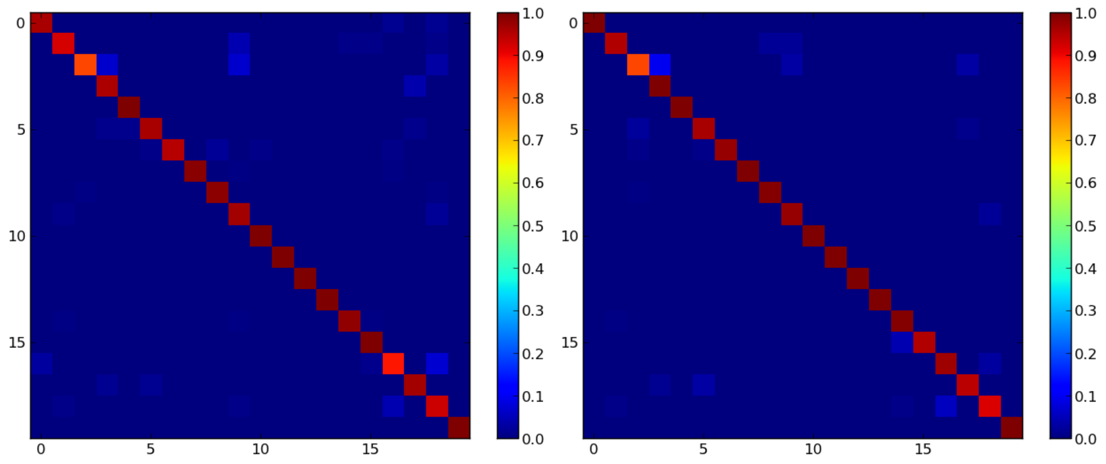


Figure 3.18: *Confusion matrices for the AdaBoost ensemble (left) as compared to the classifiers of the concatenation of all features with a linear SVM classifier (right) for the 20 classes problem. Best viewed in color.*

quential approach was used (Marton et al., 2011), here all modalities are considered together. Ensemble methods are a viable way of combining the various sources of information, and they allow for a higher modularity and efficiency than simple concatenations of features. As we saw, with the right approach, we can improve not only over the best ensemble member, but also the concatenation.

3.4 Discussion

In this chapter we introduced a perception system that is capable of multi-cue information fusing for object recognition tasks. In the following chapters, the presented approach for taking advantage of multiple sources of information was shown to reliably deal with a large number of objects and environmental conditions.

In the experiments presented until now we assumed a correct segmentation, that is however difficult to obtain in cluttered scenes. Therefore, segmenting and categorizing objects in such scenarios requires part based approaches, which we explore next (for furniture pieces and small objects). As we will see, the voting-based ensemble learning method of our perception system was used to merge information from different detected parts and viewpoints, leading to superior performance over alternative solutions.

Chapter 4 will deal with creating an environment model, while Chapter 5 with object recognition, both in complex scenes, with high clutter and occlusion levels.

Chapter 4

Modeling the Robot's Workspace

Up to now it has been assumed that all our cognition must conform to the objects; but [...] let us once try whether we do not get farther with the problems of metaphysics by assuming that the objects must conform to our cognition.

IMMANUEL KANT

Having described a repertoire of scene and object examination methods, and the integration strategies employed by our system, this chapter will focus on how to use them in real-wold settings to achieve complex tasks. First and foremost, the robot needs to understand the environment in which it is operating. This conceptual understanding is a semantic map of the surroundings and the objects therein, that is either provided to the robot or built autonomously. This chapter will detail and evaluate the segmentation, recognition, and modeling methods for automatic semantic mapping in cluttered settings, i.e. where different objects are nearby or touching, such that a correct segmentation is challenging.

Semantic maps are resources which robots can use to aid them during the execution of their task. Whenever a robot has to act in environments designed for humans, it can take advantage of the wealth of common sense information to interpret and recognize the different common objects in it. Therefore, to build semantic maps it needs to find known or learned structures that are typical in man made environments.

When modeling the robot's workspace, the imposed logical and physical constraints must be satisfied by the built models. This can be achieved by "hallucinating" detection hypotheses based on our understanding of the objects and checking if these illusions conform to our knowledge about the real scene. After the detection of larger structures, these can guide the detection of smaller objects. Thus, the recognition of typical objects like chairs and armchairs, tables and sideboards will facilitate more powerful task planning, because they could be important for the task at hand. Tables can be used to place things for the use of the human, sideboards are containers that could hold objects that the robot has to find, and chairs could be potentially moved out of the way if needed.

To enable modeling in cluttered scenes, we will focus on the combination of bottom up reconstruction from parts and top down hints (from predefined common sense knowledge, or learned concepts from the World-Wide-Web). Physical constraints are incorporated by assuming stable, non-intersecting configurations, and also by the knowledge about occupied and unoccupied space that 3D technologies provide. Additionally, the benefit of over-segmentation and multiple viewpoints is introduced, to be explored further in the next chapter.

4.1 Segmenting Furniture and Appliance Doors

First, I will give an overview of the segmentation and model reconstruction methods that we employed for detecting container doors during the creation of Semantic Object Maps (Rusu et al., 2008c; Marton et al., 2008; Rusu et al., 2009d; Blodow et al., 2011a), as shown in Figure 4.1. For this, we used and extended the modules described in Chapter 2 to be able to deal with incremental data acquisition, use geometric fitting of handles to guide door segmentation, and to create 3D models that can be used by the robot for planning its interaction strategies.

4.1.1 Incremental Data Acquisition

A single scan typically is not enough to capture the complete workspace, so multiple viewpoints are needed. To construct complete models from partial views, the existing model needs to be updated by aligning it with newly acquired data scans, using the presented efficient methods for exploration and point cloud registration. These scans are overlapping with the current surface mesh, and all the new points have to

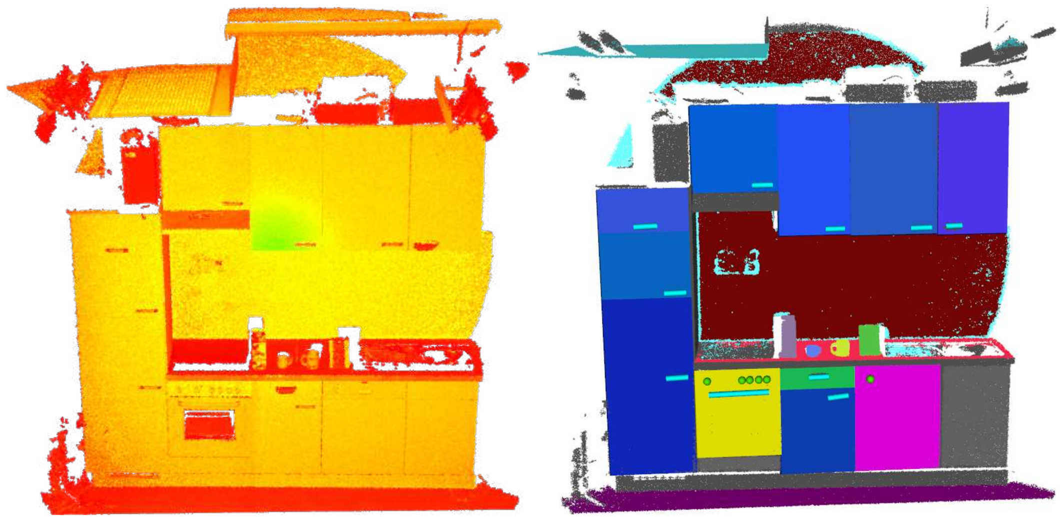


Figure 4.1: *Segmenting touching furniture and appliance doors from raw data. Doors and their corresponding handles are segmented, and different container types classified by the properties of their parts and the relations between them, based on online training data (see 2.28). Blue: cupboard, green: drawer, yellow: owen, purple: dishwasher, red: table, dark red/purple: wall and floor, grey: unrecognized/obstacle, and detected objects in random color. Please note that since the bottom right furniture face (below the sink) did not have a handle it was not classified as a container.*

be incorporated into the model. However, neglecting all the previously constructed triangulations and creating a new mesh containing all the points each time a new scan is acquired does not constitute a feasible solution.

To solve this problem, our method triangulates the new vertices of the point cloud *into* the existing mesh (Marton et al., 2009b). By determining the overlapping area between each new data scan and the current surface mesh, and removing those points which are close to existing triangle faces, we can grow the existing model without recreating the entire triangular mesh.

In order to include points that are in the vicinity of a triangle into the mesh, the corresponding triangle is *neglected*, and is treated as a hole containing the new points inside it, by initializing the advancing front with its vertices. The method is similar to the normal initialization, where a seed triangle is *filled* and the advancing front is initialized in the same way, with its vertices. This guarantees that no change is required in the actual triangulation algorithm – it is basically a restart of the previously

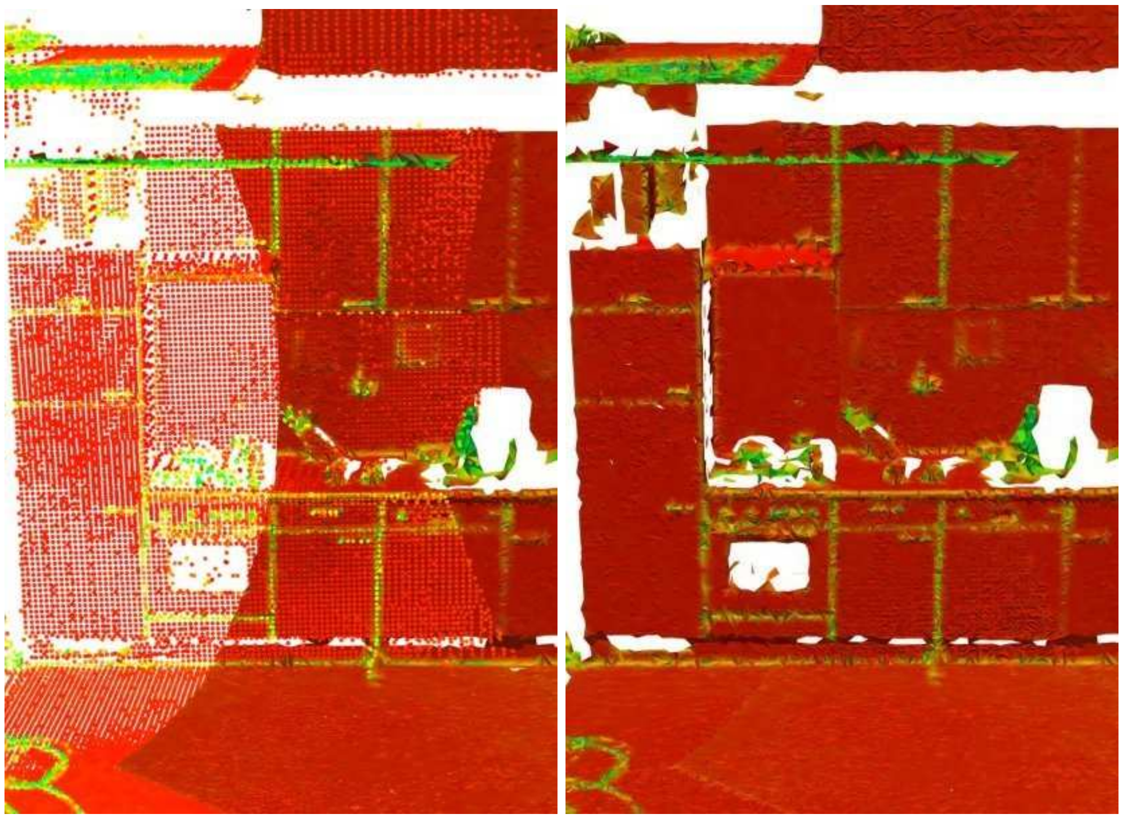


Figure 4.2: *Left: a new registered data scan has been acquired and the mesh model needs to be updated; right: the results after incremental triangulation, by preserving the mesh in the parts where no data update exists, and triangulating the new vertices. The colors represent the estimated surface curvature (red meaning low, and green high curvature).*

finished step. The points in the new parts of the scene are also straightforward to include, as the previous step's boundary loop has to be treated as the advancing front that will expand into the new area.

Figure 4.2 presents the result of these re-initialization methods in the kitchen scene which was extended with a new scan performed by the robot. To obtain a more compact representation, the high density of points can be reduced by adaptively re-sampling the surface in the overlapping area, as described in subection 2.3.1.

4.1.2 Detecting Doors and Containers based on Fixtures

Man made environments typically contain a large number of planar structures, of which the ones possessing handles (indicating that they can be manipulated) are some sort of door that can be relevant for a household robot. As an exhaustive search for all planes is computationally intractable, we are only searching for those that are aligned with the walls of the room. The orientations of the main walls are determined using a RANSAC based approach on the normal sphere, as in (Marton et al., 2010a). 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.

After extracting the principal axis-aligned planes, they are classified into floor and ceiling based on horizontal orientation and height, and parts of the walls based on the observation that they are typically adjacent to the ceiling. The remaining planar connected components constitute candidates for tables or furniture faces. In order to detect fixtures we first find point clusters that are within the polygonal prism of the furniture faces using euclidean distance measure and then fit RANSAC lines or circles to those clusters and thereby differentiate between handles and knobs.

Kitchen appliances, doors and drawers typically have fixtures that allow interaction with them. The existence of fixtures is a good indication to the presence of these objects, so the algorithm searches for clusters of points in the vicinity of detected vertical planar structures. Since the ultimate goal is the manipulation of the handles by our robot, we discard clusters that are too big in diameter for grasping. These filters are simple enough to be performed for all possible clusters and explain all of the fixtures in typical kitchen environments. The result of this process can be seen in Figure 4.3.

In our previous work (Marton et al., 2008), we found gaps between adjacent cabinet doors using curvature from laser data alone. We used handles as guides for segmentation as well for finding the correct breakup of regions that are likely to be under-segmented. Since the accuracy of our current laser sensor is considerably lower than the one employed there, we cannot purely rely on geometry, but perform the segmentation of furniture faces using camera images registered onto our laser data (see Figure 4.4).

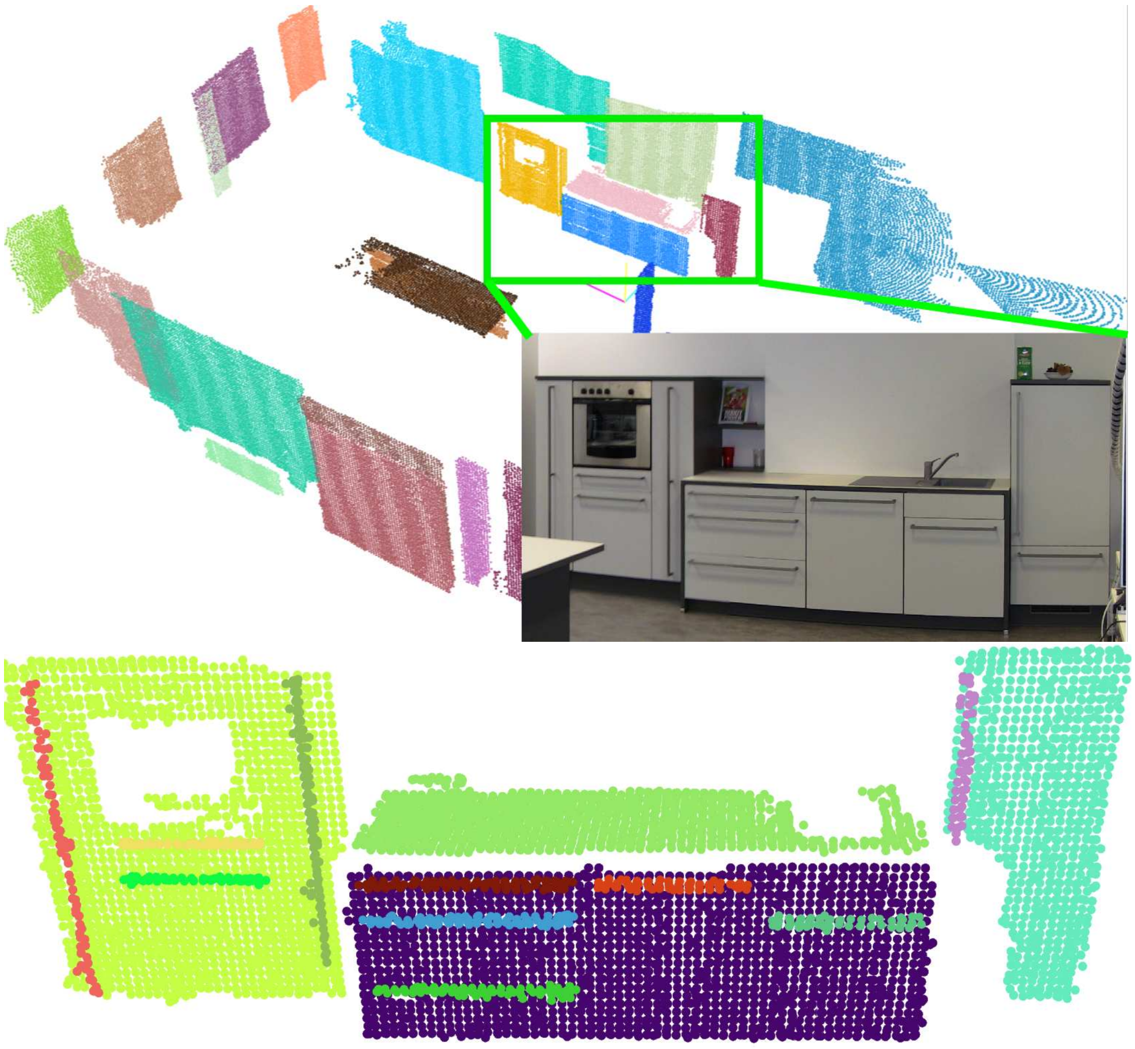


Figure 4.3: *Segmentation results for a kitchen visualized with floor and ceiling points removed.*

The algorithm uses seed points around the footprint of fixtures to estimate an initial model of the color distribution of the door, consisting of the intensity values' median $\tilde{i}$ and median average distance (*MAD*). The seed regions are expanded by adding neighboring points whose colors match the estimated color model (using region growing) based on the assumption that points on the door border are surrounded by points with different color. The color model for a region is updated after all possible points 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 map, along with rectangular approximations to the found planar segments.

The algorithm is parameterized on the maximum search radius for fixture footprint points, the equivalent threshold for the region growing phase, and a color threshold α, which defines how much the color of a point is allowed to deviate from the door color model. In our experiments, we found a value of $\alpha = 2 \cdot MAD$ to yield stable results. The method deals well with the shadows of handles and doors of different appearance (metallic for the oven and light gray for the rest of our kitchen). Stronger shadows did prevent small parts of some doors to be segmented correctly, but the rectangular approximation still included them.

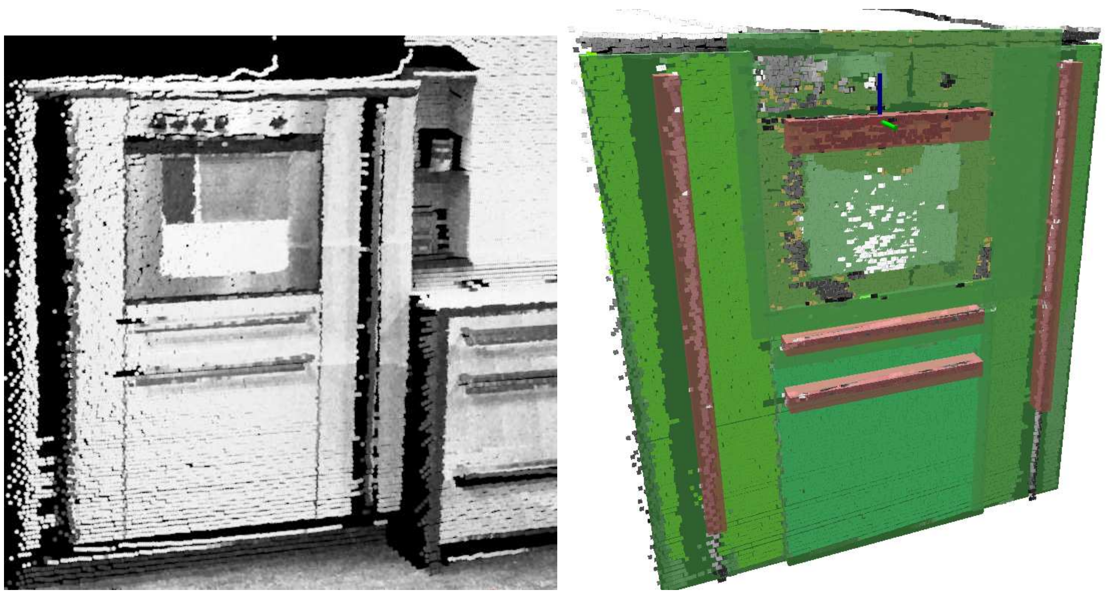


Figure 4.4: *Region growing based generation of drawer and door hypotheses. Left: point cloud data overlaid with reprojected intensity information. Right: the same kitchen part overlayed with segmentation results for doors (shades of green) and handles (red). Note the difficulty of segmenting the shiny oven door.*

Since the robot can interact with the environment, doors can also be segmented by opening them, and evaluating the temporal differences. However, this process is relatively slow, so one idea would be to only do this for ambiguous segmentations. However, when opening them, we can also determine the type of joint (rotational or prismatic, i.e. translational). This means that for mapping whole kitchens, including estimating articulation models, we do perform this for every handle found, but if necessary, we can fall back to geometric and visual segmentation in cases where time is critical or where the handles can not be operated by the robot gripper (Blodow et al., 2011a).

4.1.3 Semantic Interpretation

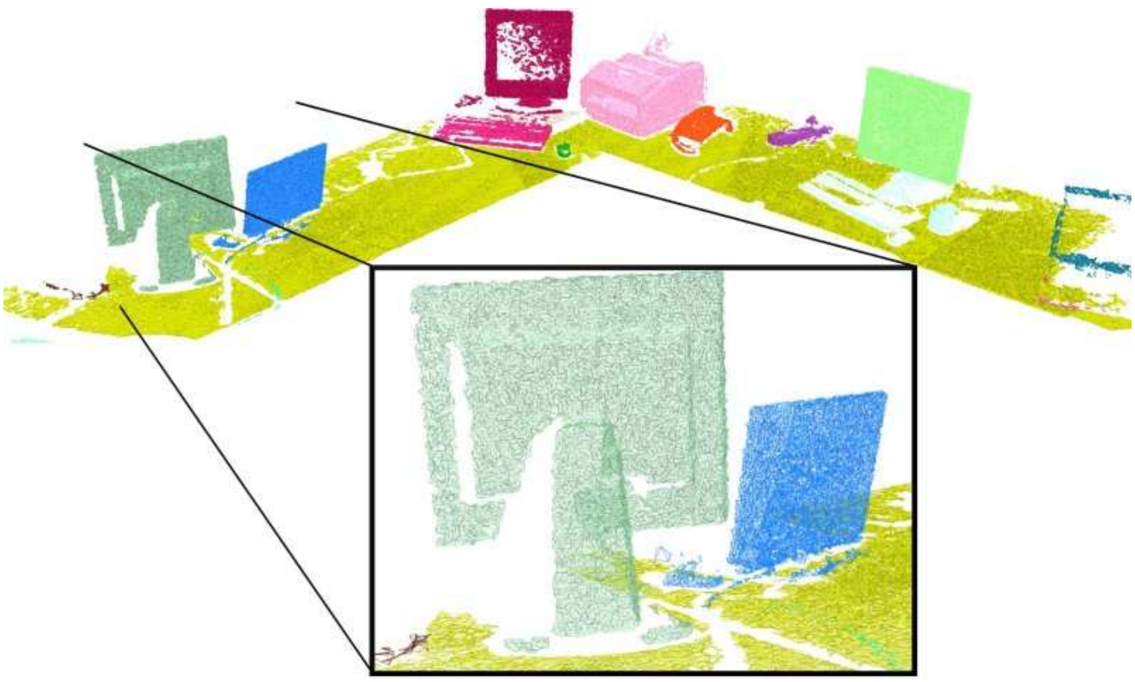
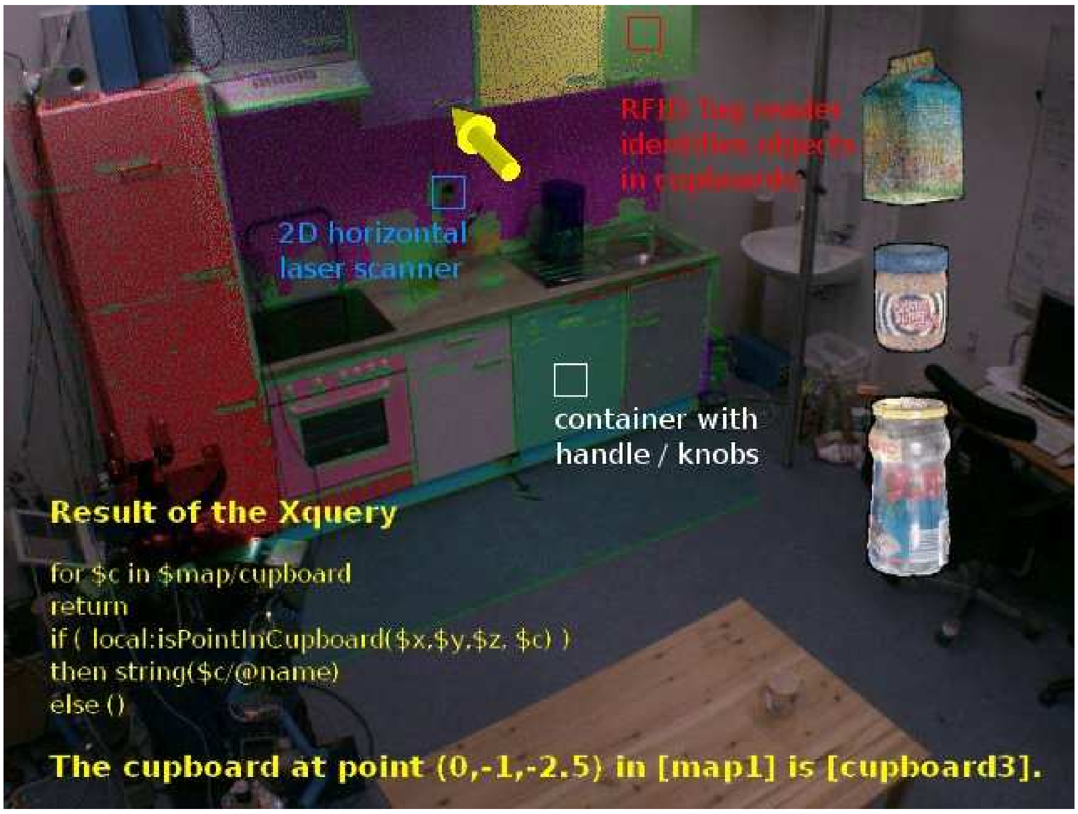


Figure 4.5: *Surface models generated for detected objects standing on a table.*

As seen in Figures 4.1 and 4.5, the environment includes several objects that can be classified as described in the previous chapter and their surface models incorporated into the map.

The objects in our maps are instances of object classes where the object classes are organized in a specialization hierarchy. Thus, a cupboard is a specialization of a container and can therefore be used to store objects, and an oven is a specific kind of container that is used to prepare meals. As an intermediary representation XML is used, such that we can import them into different applications. For example, in the



```
for $c in $map/cupboard
return
if ( local:isPointInCupboard($x,$y,$z, $c) )
then string($c/@name)
else ()
```

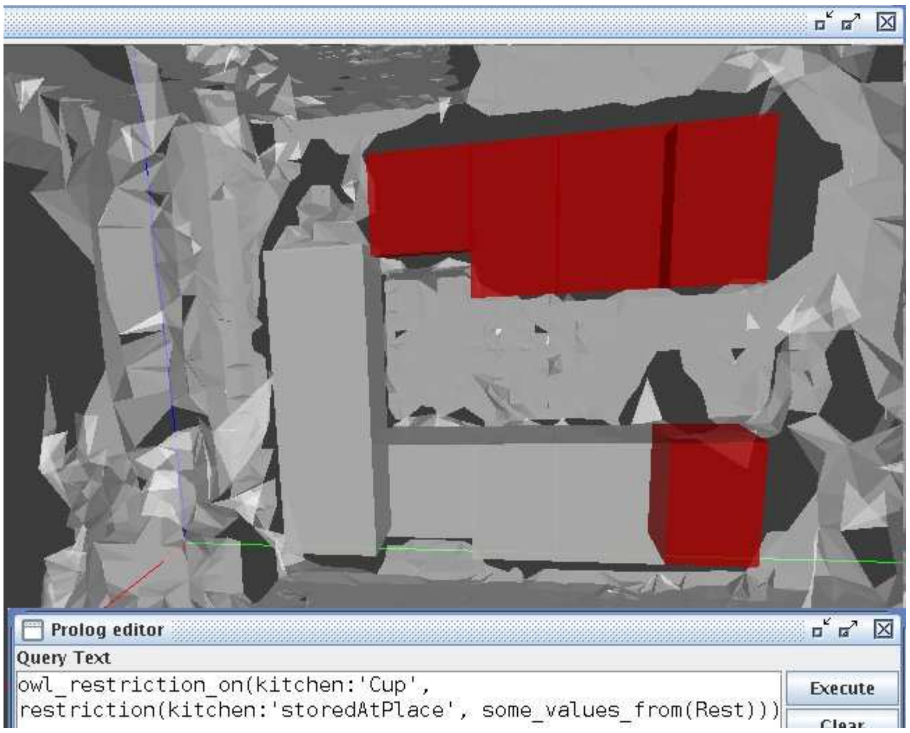
```
The cupboard at point (0,-1,-2.5) in [map1] is [cupboard3].
```

Figure 4.6: *Querying the object map using XQuery and the visualization of the results (yellow arrow) superimposed on a photograph of the scanned kitchen. Labeled 3D points and semantic information from other sources is also overlayed.*

Gazebo 3D simulator[1] it is possible to perform a validation of the estimated door hinges and object classes (Marton et al., 2008), while the map can be queried using XQuery in order to retrieve necessary information from the model, as shown in Figure 4.6.

In order to import the map into our knowledge processing system, it is transformed into the Web Ontology Language OWL-DL, with objects being identified as instances of the classes in our knowledge base. This allows for querying all relevant knowledge and performing reasoning about objects which are grounded in actual sensor data. This makes it possible to query for objects that serve for a purpose (see Figure 4.7), and get their spatial properties as the result. For every detected furniture object, we record a dataset that contains an ID, the type of the container/articulation model

[1]http://gazebosim.org/



```
owl_restriction_on(kitchen:'Cup',
restriction(kitchen:'storedAtPlace', some_values_from(Rest)))
```

Figure 4.7: *The results are exported to KnowRob (Tenorth, 2011) in OWL-DL representation that allows for semantic queries. The integration with KnowRob can be used for example to locate containers where cups can be stored (results highlighted in red). In the figure, all objects identified as cupboards are returned since the query asks for objects where cups are stored. (Image courtesy of Moritz Tenorth.)*

found, geometric extents in depth, width and height, and the position and orientation in space. We also record hierarchical information such as a kitchenette consisting of several pieces of furniture or which handles were found on which furniture pieces. It is being used in the knowledge processing system KnowRob (Tenorth et al., 2010) and can be exported to URDF (Unified Robot Description Format), which we use in order to get the collision and the articulation models of environments (see Figure 4.8).

This map was used to reason about and interpret human gestures in (Blodow et al., 2011b), where we combined it with human tracking and our 3D segmentation methods[2] in order to generate plans for the robot. For example, virtually grabbing an

[2]See our entry to the ROS 3D Contest:
http://www.ros.org/news/2011/02/ros-3d-entries-teleop-kinect-cleanup.html

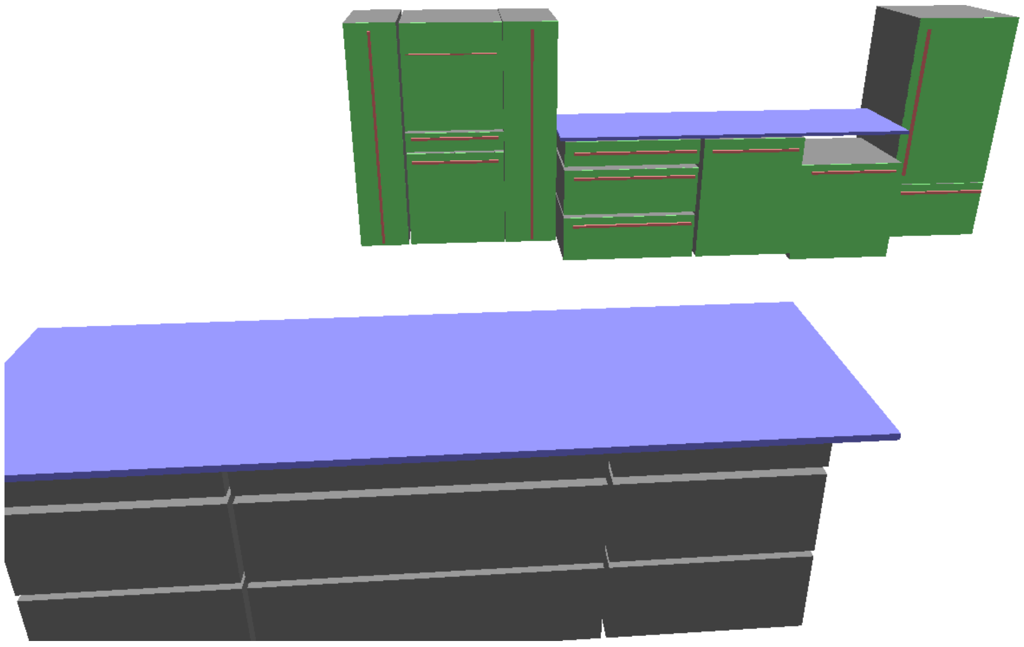


Figure 4.8: *Semantic map of the kitchen from Figure 4.3 as represented for KnowRob.*

object and hovering it into the fridge could be interpreted as grasping, door opening, releasing and door closing by KnowRob.

Such complex interactions can be further improved by obtaining information on other furniture pieces as well besides appliances, containers and doors. Since other furniture pieces are not as simple to describe as a having a door with fixtures, we explore this topic further in the next section.

4.2 Detecting Furniture Pieces by their Part Configurations

In this section, I will address the problem of exploiting the structure in today's workplace interiors in order for service robots to add semantics to their sensor readings and to build models of their environment by learning generic descriptors from online object databases. These world models include information about the location, the shape and the pose of furniture pieces (chairs, armchairs, tables and sideboards), which allow robots to perform their tasks more flexibly and efficiently.

To recognize the different objects in real environments, where high clutter and occlusions are common, our method, introduced in (Mozos et al., 2011), automatically learns a vocabulary of object parts from CAD models downloaded from the Web. After a segmentation and a probabilistic Hough voting step, likely object locations and a list of its assumed parts can be obtained without full visibility and without any prior about their locations. These detections are then verified by finding the best fitting object model, filtering out false positives and enabling interaction with the objects.

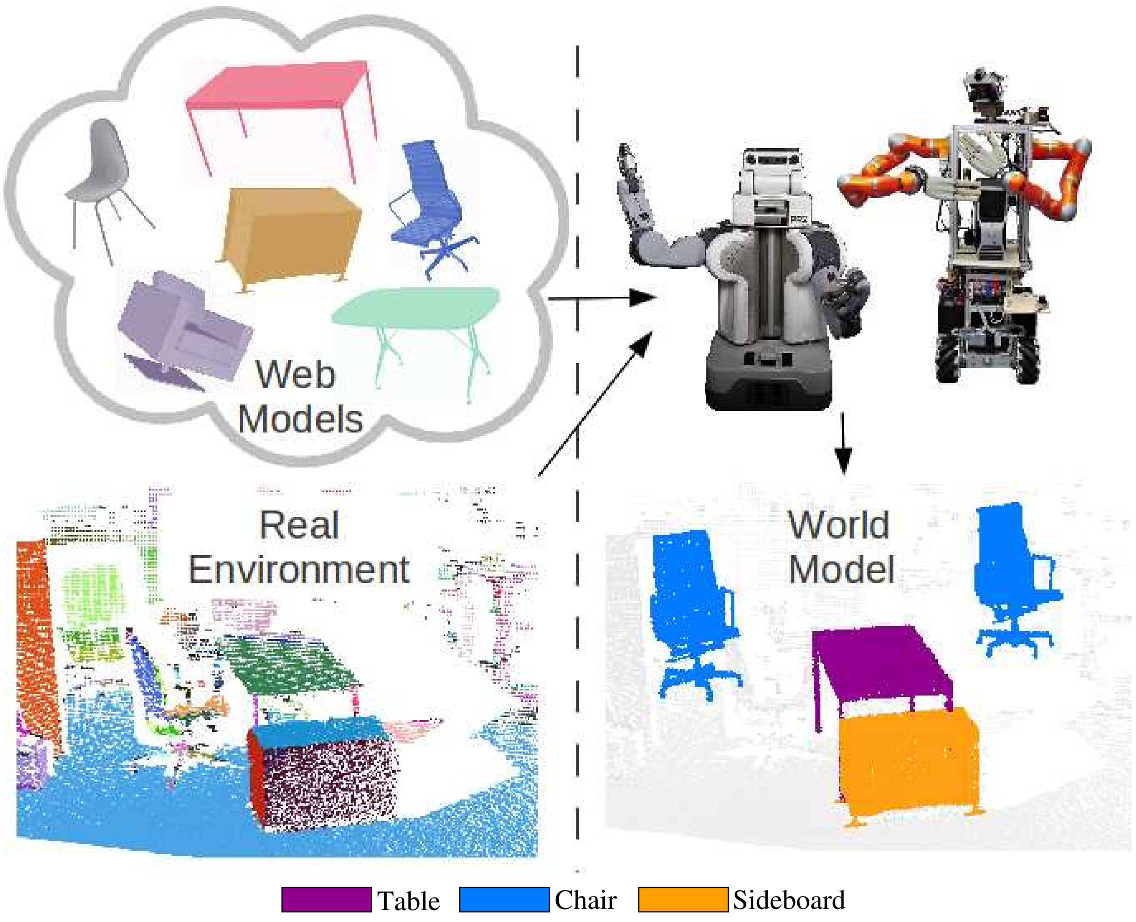


Figure 4.9: *Using furniture models from the WWW together with a segmented scan of its real environment, the robots create a world model (different parts are indicated by random colors).*

This application scenario will demonstrate how autonomous robots can exploit the high quality information already available from the WWW concerning 3D models of office furniture. Apart from the hobbyist effort in Google 3D Warehouse, many companies providing office furnishing have already modeled considerable portions of the objects found in our workplaces and homes. In particular, we present an approach that allows a robot to learn generic models of typical office furniture using examples found in the Web. These generic models are then used by the robot to locate and categorize unknown furniture in real indoor environments as shown in Figure 4.9.

Furniture pieces share many common parts, especially if their purpose is similar. For example, most chairs have an approximately horizontal and vertical part, and rectangular planar patches are quite common. Thus the key idea of this work is to represent the object models learned by the robot using a vocabulary of these common parts, together with their specific spatial distributions in the training objects.

During the training process the CAD (Computer Aided Design) models from furniture Web catalogs are converted into point clouds using a realistic simulation of laser scans, as described in Section 2.6. These point clouds are segmented and the resulting parts from the different training objects are clustered to create a vocabulary of parts. In the detection step, we match similar parts and apply probabilistic Hough voting to get initial estimates about the location and categories of objects found in the scene. Finally, these detections are verified by finding the CAD model that best fits to the measurements. This last step allows the robot to reject false positive detections.

Using larger regions (instead of points) as basic units for learning offers many advantages. As was already shown in Sections 4.1 and 2.7, the knowledge about different functional units of objects can contribute significantly to a correct detection. Moreover, we use training examples that are similar but not necessarily the same as the objects encountered during the operation of the robot, thus improving the generalization of the final classifier.

4.2.1 3D Point Cloud Segmentation

Our classification of objects is based on the detection of the different parts that compose them. To determine these parts, we segment the 3D point clouds representing the objects and scenes using the normal-based part decomposition algorithm described in Section 2.4. A segmentation defines a disjunct partition $\mathscr{P} = \{S_1, \ldots, S_M\}$ of the 3D point cloud. Our segmentation method follows a criterion based on a maximum angle difference between the surface normals. This condition is easily checked and can be applied to any type of surface (see Figure 2.20). We smooth the PCD and for each point we calculate its normal using RMLS (see Equation 2.4).

Using the obtained normals for each point in the cloud, we apply a region growing algorithm where we mark a point p as belonging to a part S if the distance between the point p and some point in S is closer than $\delta = 5$ cm, and if the angle formed by the normal of p and the seed normal of S is less than $\alpha = 40^\circ$. Seed points are iteratively selected as points with the lowest curvature that do not belong to any part yet. This ensures that flat parts are identified first and makes the identification process more robust. The parts that have less than 10 points are considered to be too small, and are most probably produced in regions with high normal variations or by spurious points. Thus we perform a simple distance-based region growing to group them together and the resulting parts that are still too small are discarded. The parameters were selected because they provide good partitions in our setup (see subsection 4.2.5). An example segmentation of an object is depicted in Figure 4.10, and segmentations of point clouds in indoor environments are shown in Figures 4.12 and 4.13.

Finally, our segmentation method produces parts with only slight curves. As explained in the introduction, the reason is that this kind of surface is very common in furniture objects in indoor environments. However, cylindrical objects would be broken up into parts covering at most 2α degrees, and the extracted features account for the curvature of the part.

4.2.2 Training Objects from Web Catalogs

In this work we use established online databases, to obtain information about typical objects found in indoor environments, as shown in Figure 2.27.

To obtain realistic point cloud representations for these objects, we simulated our laser scanner's sweeping motion on the robot, intersected each beam with the CAD model of the object, and added realistic noise to the depth measurements. Then, each obtained scan was segmented to obtain its part structure. An example process for obtaining training data for a chair is shown in Figure 4.10. The whole process takes, on average, 4.67s per view on a single core using a good graphics card.

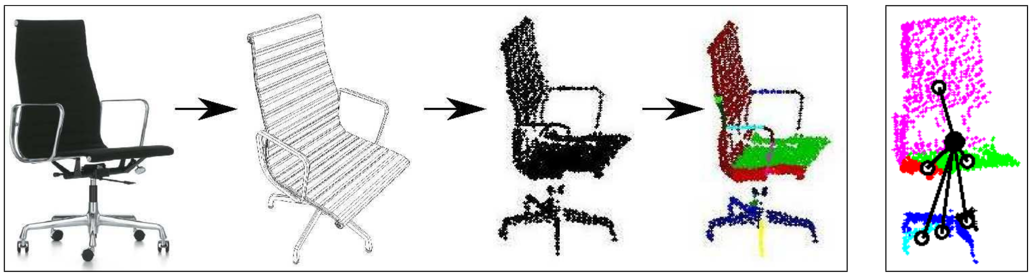


Figure 4.10: *Left: Example point cloud acquisition and segmentation of a chair. Right: Example shape model for a partial view. Please note that one of the legs and the extensible beam of the chair were occluded in the scan.*

4.2.3 Vocabulary of Parts

We build a common vocabulary of parts for all the classes in the training data, since most of the objects contain similar parts (Usenko et al., 2012). The vocabulary is constructed by segmenting the training objects into parts, each of which is then represented by a feature vector encoding its geometrical properties, and the feature vectors are clustered to form groups of similar object parts. Next we describe the last two steps in detail.

4.2.3.1 Feature Vectors for Parts

We found that the various part types differ quite substantially from another (e.g. legs, table lanes, chair back-leans, etc), so a simple statistical features that considers part orientation as well (like in the case of inaccurate depth data in Chapter 2) can outperform global 3D features designed for general object recognition (Marton et al., 2014). Therefore, for each part S obtained in the segmentation from 4.2.1, we calculate the following set of geometrical features:

1. Proportion of boundary points in S computed as in (Rusu et al., 2008c).
2. Average curvature of S computed as the smallest eigenvalue's proportion to the sum of eigenvalues in the local neighborhoods of all points.
3. Volume occupied by the voxelized points in S.

4. We calculate three eigenvalues, e_1, e_2, e_3, of S and calculate six proportions: e_1/e_2, e_2/e_3, e_1/e_3, e_1/sum, e_2/sum, e_3/sum, where $sum = e_1 + e_2 + e_3$.
5. We obtain three eigenvectors, $\vec{e_1}$, $\vec{e_1}$, $\vec{e_1}$, of S, project the points onto each of them, and calculate three metric variances v_1, v_2, v_3 (which we used instead of e_1, e_2, e_3).
6. Orientation of the eigenvector corresponding to the smallest eigenvalue, indicating the orientation of S.
7. We project the points onto each eigenvector and get the distance to the farthest point from the medium in both directions $l^1_{\vec{e_1}}, l^2_{\vec{e_1}}, l^1_{\vec{e_2}}, l^2_{\vec{e_2}}, l^1_{\vec{e_3}}, l^2_{\vec{e_3}}$. We then calculate the following values: $(l^1_{\vec{e_1}} + l^2_{\vec{e_1}})$, $(l^1_{\vec{e_2}} + l^2_{\vec{e_2}})$, $(l^1_{\vec{e_3}} + l^2_{\vec{e_3}})$, $(l^1_{\vec{e_1}}/l^2_{\vec{e_1}})$, $(l^1_{\vec{e_2}}/l^2_{\vec{e_2}})$, $(l^1_{\vec{e_3}}/l^2_{\vec{e_3}})$, $(l^1_{\vec{e_1}} + l^2_{\vec{e_1}})/(l^1_{\vec{e_2}} + l^2_{\vec{e_2}})$.
8. Three proportions between features 5 and 7: $v_1/(l^1_{\vec{e_1}} + l^2_{\vec{e_1}})$, $v_2/(l^1_{\vec{e_2}} + l^2_{\vec{e_2}})$, $v_3/(l^1_{\vec{e_3}} + l^2_{\vec{e_3}})$.
9. Proportion between the occupied volume (feature 3) and the volume of the oriented bounding box of the part.

Each part S is finally represented by a vector containing 24 features normalized to the range [0,1].

Similar features were used in parallel to our work for grouping similar object and body parts together with great success in an assembly-based 3D modeling work (Chaudhuri et al., 2011). As there for humanoid and vehicle parts, in our case for furniture parts as well, these types of features create semantically meaningful clusters.

4.2.3.2 Words and Shape Models

The resulting set of training feature vectors is clustered to obtain the words forming the vocabulary of parts. In our approach, we apply k-means, since it has given good results in previous works (Csurka et al., 2004; Coates et al., 2010). After applying k-means to the training feature space, we obtained a clustering $\mathscr{C} = \{C_1, \ldots, C_V\}$, which represents our vocabulary of parts. Each cluster C_i is called a *word*. An example word

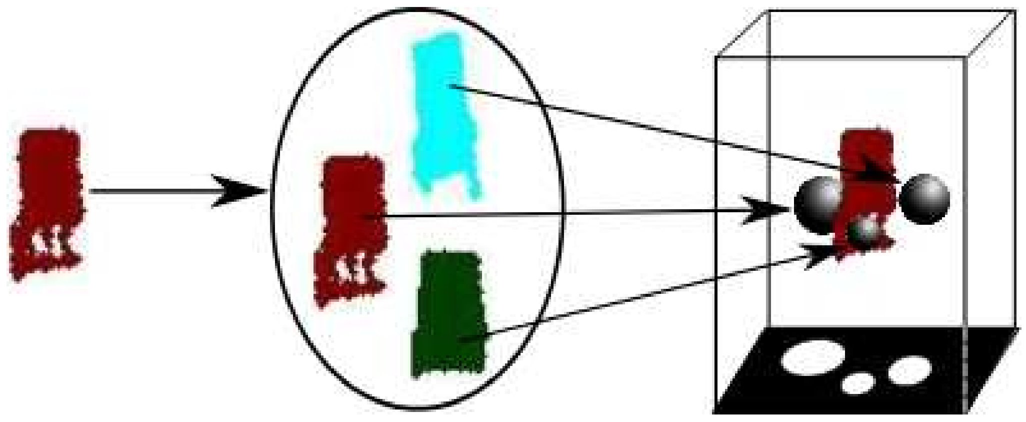


Figure 4.11: *Example word activation and corresponding 2D voting space.*

is shown in the center of Figure 4.11 representing the back of a chair from different views.

In addition, and following (Leibe et al., 2008), we learn a shape model for each training object view. This model specifies the distance relations among the different parts that compose the object. We extract the center of mass $s \in \Re^3$ of each part S, and the center of mass $p \in \Re^3$ of the complete point cloud. We then calculate each relation as the 3D vector $\vec{d} = p - s$. An example is shown in the right image of Figure 4.10.

4.2.4 Object Recognition

In contrast to previous works (Nüchter et al., 2005; Klasing, 2010), we do not isolate possible objects in the scene before the classification. In our approach, we detect the objects by simultaneously collecting the information provided by all the parts in the scene.

Our object recognition process is composed of three main steps: i) a set of hypotheses is generated which indicate possible locations for objects in the point cloud, ii) a selection of the best hypotheses is done following a set of criteria, iii) model fitting and verification is applied.

4.2.4.1 Hypotheses Generation

Here we make use of the ensemble learning method that we found to be very robust in the previous chapter: weighted voting. Given an initial partition $\mathscr{P} = \{S_1, \ldots, S_M\}$ of a 3D point cloud representing an indoor environment, we first obtain the corresponding feature vector f_i for each part S_i. Each feature vector is then matched to a subset of words $\mathscr{A} \subset \mathscr{C}$ from the learned vocabulary (activation), which constitute possible interpretations of the part S. Each element of an activated word casts a vote for a possible object location. This scheme is known as probabilistic Hough voting (Leibe et al., 2008), and an example is shown in Figure 4.11. In this case, a part in the scene activates a word representing backs of chairs. Each element in A casts a vote for the center of the chair inside the point cloud.

Formally, given a feature vector f representing a part S located at position l in a 3D point cloud, the probability of finding an object of class o at position x is

$$p(o, x \mid f, l) = \sum_i p(o, x \mid A_i, f, l) p(A_i \mid f, l). \tag{4.1}$$

The term $p(o, x \mid A_i, f, l)$ represents the probability of finding object o at location x given that feature f at position l activated the word A_i. The term $p(A_i \mid f, l)$ indicates the probability that the feature vector f matches the word A_i.

As we discussed in (Mozos et al., 2011), we make two further assumptions. First, the probability of finding an object given an activated word A_i is independent of the feature vector f. Second, the activation of a word A_i by feature vector f is independent of the location of the corresponding part S. Thus,

$$p(o, x \mid f, l) = \sum_i p(o, x \mid A_i, l) p(A_i \mid f). \tag{4.2}$$

It remains to describe each term in (4.2). The term $p(A_i \mid f)$ represents the activation of A_i by f. In order to do ensemble learning, we do not activate just one vocabulary word, but several of them. Each activation is assigned a probability indicating how well the feature vector f matches the codebook entry A_i as

$$p(A_i \mid f) = \frac{\mathrm{w}(f, A_i)}{\sum_j \mathrm{w}(f, A_j)} \tag{4.3}$$

where $\mathrm{w}(f, A_i)$ is a distance function proportional to the inverse of the Euclidean distance between vector f and the mean vector of cluster A_i.

The term $p(o, x \mid A_i, l)$ represents the probability of finding object o at position x given the activated word A_i at location l. Once a word is activated, each of its elements a casts a vote to the position $x_v = l + \vec{d}$. Here l is the position of the part S found in the point cloud, and $\vec{d}$ is the relation vector of the element's shape model (see subsection 4.2.3.2). Finally the probability $p(o, x \mid A_i, l)$ is given by

$$p(o, x \mid A_i, l) = \begin{cases} \frac{1}{\#(A_i, o)} & \text{if } x = x_v \,; \\ 0.0 & \text{otherwise,} \end{cases} \tag{4.4}$$

where $\#(A_i, o)$ indicates the number of elements a in word A_i that pertain to object o.

4.2.4.2 Hypothesis Selection

The result of the voting process is a set of weighted hypotheses for possible object locations as shown in Figure 4.11. The final score $V(o, x)$ of a hypothesis representing an object and its position can be obtained by marginalizing over all the parts found in the scene,

$$V(o, x) = \sum_k p(o, x \mid f_k, l_k) p(f_k \mid l_k) \,. \tag{4.5}$$

The first term in this expression is calculated as in (4.2), and the second term $p(f_k \mid l_k)$ is assumed to be a uniform distribution.

To find the locations of the different object hypotheses, we search for the most likely positions (analogously to the most likely class label in the previous chapter). Local maxima in the voting space are identified using a search window whose size corresponds to the width of the particular object class we are looking for. We project the votes onto the (x,y)-plane in order to simplify this search (see Figure 4.11). After obtaining the local maxima, we apply a threshold to select the best hypotheses. Moreover, hypotheses are checked for inconsistencies in the space. In our case, two hypotheses for the same type of object at two different locations x_i and x_j are inconsistent if the 2D-convex hulls of the objects centered at x_i and x_j overlap. This condition assumes that all objects are lying on the floor.

4.2.4.3 Model Fitting and Verification

Having detected positions where a certain object type is likely to be found, we verify the detection and select the best model for the object, along with its pose. To be able to do this, we first need to retrieve the most important parts that voted for the location. Some of these parts are incorrect but their weights are low.

Our task is now to select the model and its pose from a 3D model database that explains most of the detected object parts. We employ a more detailed model database for this, in order to be able to account for the variations in shapes and sizes of the different objects. To fit these models to our 3D scans robustly and efficiently, we transform them to noiseless point clouds by filling the outer triangle faces with points, and use a RANSAC-based algorithm to select the best pose and the best model for each detected object. Since we take the complete model into account at once instead of each view separately, we can reduce the number of trials considerably and we do not require the point cloud to come from a single scan. This can become important later, as detailed in subsection 4.2.5.3. The search time is further reduced by assuming that objects are upright, thus only the location in 2D and the rotation around the up axis is required to be found.

In a regular RANSAC algorithm, one can estimate the number of required iterations T to find the best model with probability $p_{success}$ as

$$1 - p_{success} = (1 - p_{good})^T \Rightarrow T = \frac{log(1 - p_{success})}{log(1 - p_{good})}. \tag{4.6}$$

The value of $p_{good} = w^n$ is the probability of randomly selecting the best model, and can be estimated after each fit (that is better than the best one found so far) as the probability w of selecting a good sample to the power of the number of samples n. As the algorithm finds models that are increasingly better, the value of w can be updated as $\#inliers/\#points$, resulting in a decrease in the number of iterations needed. Thus the number of iterations adapts to the estimated number of matches, starting at infinity when no inliers are known and decreasing to a number that ensures that the chance of finding a better model than the current one drops below $1 - p_{success}$.

As the runtime depends on the number of samples, it is advisable to keep it as low as possible. If we were to pick both the model and the scan points at random, we would have had to use $p_{good} = (\#correct\_matches/\#possible\_matches)^2$ a very small number that is also hard to estimate. However, if we assume that by selecting a random point from the scan, we can find the corresponding model point. This simplifies the equation to $p_{good} = \#covered\_scan\_points/\#all\_scan\_points$. The number of covered scan points can be found by nearest neighbor searches with a maximum distance in a search tree.

We found that selecting the corresponding model point to a scan point can be done by iterating over the possible matches (points at similar height), and selecting the one that would result in a transformation that covers most scan points. Significant speedups can be achieved by selecting only a subset of the possible correspondences and scan points. Our experiments provided good results with around 50% of the points checked. Additionally, by keeping track of the best score found, subsequent searches can be stopped early if it becomes clear that they can not produce better scores. The same principle can be applied over multiple models as well.

Using these techniques, we were able to identify the best pose of the good model in around 15s on a standard dual-core laptop, without parallelization or other optimizations. Our models contain between 5000-15000 points, while the identified object parts around 2500. For non-optimally matching models, the verification takes around 60s, while non-matching models are rejected in under a second. After each object model from the identified category is fitted to the object parts, we select the one that explains the most model points.

Before this model and its pose can be used for various applications, it first needs to be checked in order to filter out false positives. We first check if the model covers less than 50% of the points of the parts, and reject such detections (e.g. the chair in column two of Figure 4.17). The threshold of 50% is tied to how well the CAD models and the real objects match. The real chair and the best fitting model are not identical; however, all CAD models we used cover much more than 50% of the points if a good match is found.

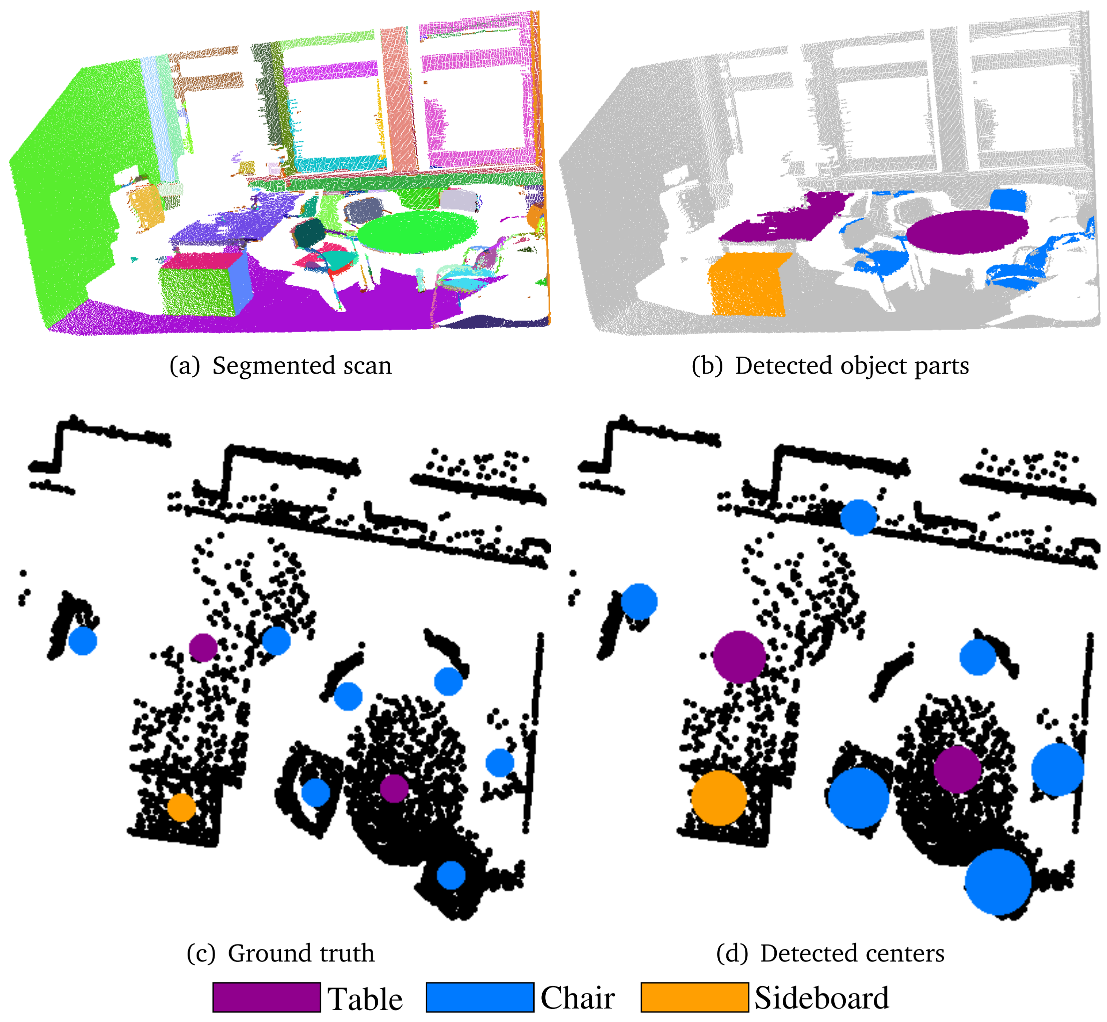
We then check if the identified pose contradicts the measurements, i.e. if it has faces that should have been visible while scanning, but the laser returned points behind them. This can be done efficiently by building one occupancy grid per scan, in which each voxel is labeled as free, occupied or unknown (Blodow et al., 2009). We reject models that have large parts in free space, e.g. the sideboard in the first column of Figure 4.17. Such a grid is needed for the operation of the robot for tasks like collision avoidance and path planning. Thus it is natural to use it for the verification of model fittings attempts, as we did in our experiments involving smaller objects as well (Marton et al., 2010a).

Finally, when the scan point has a normal vector which is close to being parallel to the upright axis, the rotation of the model can not be accurately estimated. In these cases, we use a random rotation and since we check a high percentage of points, the models are correctly identified. Tables are especially affected by this, and we see larger fitting errors in finding them but the best models get identified correctly and a subsequent fine registration using ICP (Besl and McKay, 1992) or one of its many variants can be used to correct the pose.

4.2.5 Experiments

The goal of the experiments is to demonstrate that a robot can learn models for 3D office objects found on the Web, and using these models, is able to locate and categorize pieces of furniture in 3D point clouds representing indoor environments.

To carry out the experiments, we first downloaded the CAD models of a chair, a table and a sideboard (colored as blue, red and yellow in Figure 4.9). These objects were obtained from different Web catalogs. Following the procedure of subsection 4.2.2, we placed the objects in front of the simulated laser and rotated them around the up axis, obtaining 16 scans of each object which were segmented using our algorithm. A final common vocabulary of parts was created with $k = 40$. These values provide a good balance between performance and computational costs (see Figure 4.15).



(a) Segmented scan (b) Detected object parts

(c) Ground truth (d) Detected centers

Figure 4.12: *Furniture recognition results on a scan of an office. The segmented scans of an office is depicted in (a). The detections of the objects' parts for the office is shown in (b). In (c) we can see the ground truth of the positions for the objects' centers (bird's eye view). The detected centers obtained by our method are shown in (d), where bigger circles indicate a bigger weight.*

To obtain point clouds from real scenes, we used a Hokuyo laser mounted on a pan tilt on a robot approximately 1.5m above the floor (see Figure 4.9). Service robots working in human environments typically have their 3D sensor at that height such that its perspective is similar to that of a person. Both of the robots we used had such a setup, and since they both ran ROS, changing the robot was transparent to our algorithm. However, different robots or sensors at different heights can be used if the scanning of the training data is adapted accordingly. Finally, the point clouds

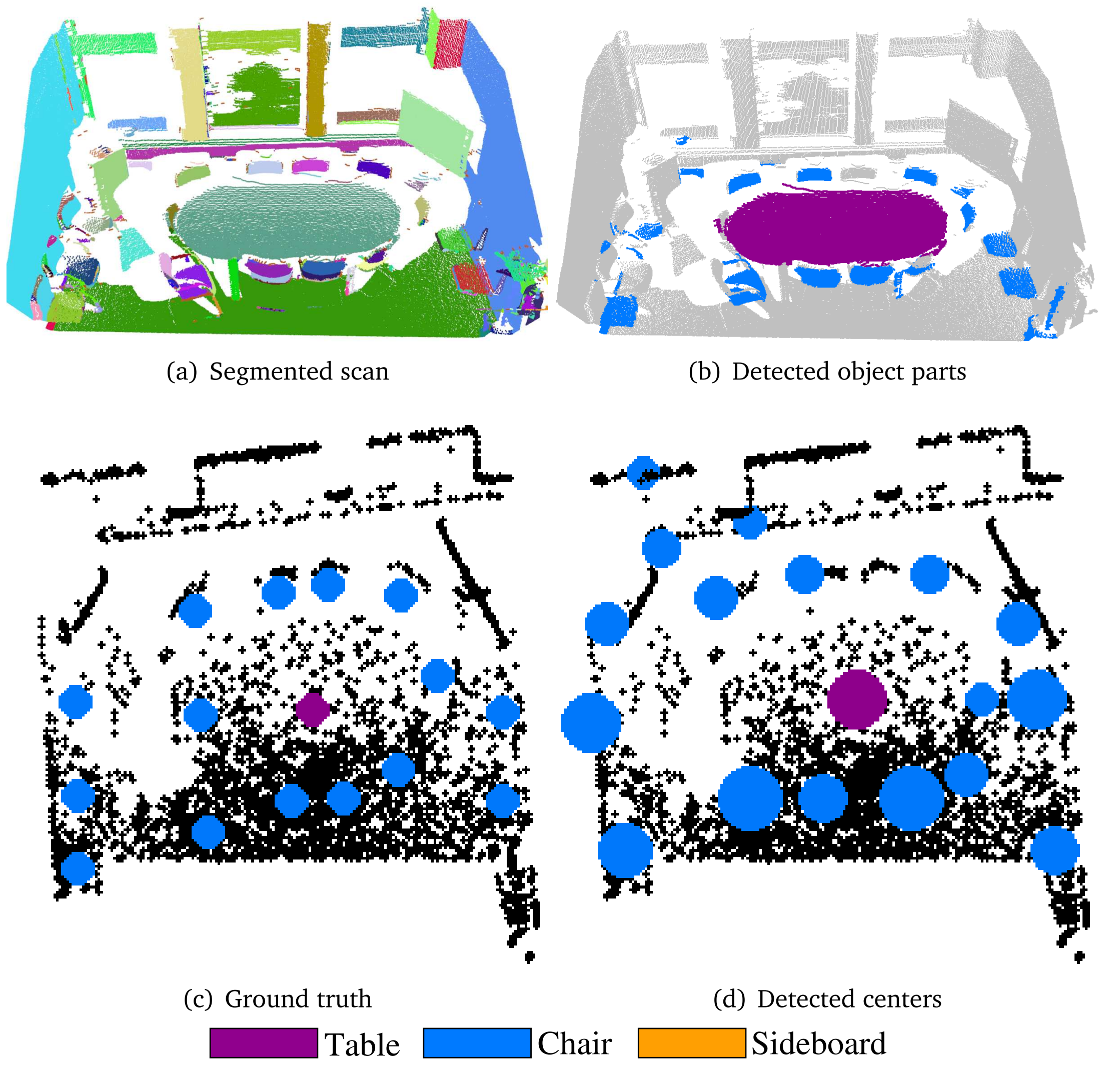


Figure 4.13: *Furniture recognition results on a scan of a seminar room. Please see Figure 4.12 for details.*

were segmented using the same algorithm as for the training data. Each obtained part activated the best two vocabulary entries.

4.2.5.1 Recognition of Objects in Real Environments

The first experiment was carried out in an office of our floor at TUM. The corresponding segmented 3D point cloud is shown in Figures 4.12 and 4.13. This office contained two tables, different armchairs and a sideboard. We would like to point out that the objects located in the office were similar to but not the same as the 3D models we downloaded from the Web. In particular, the table in the middle of the chairs was round and we did not have any round table in our training set, and the chairs around the table were different from the training models. This setting aimed to demonstrate that our approach is well-suited for detecting categories of objects instead of concrete instances only.

We set the thresholds for the final hypotheses to 1.0 for tables, 0.6 for chairs, and 1.4 for sideboards. These values were selected as desired working points in classification performance plots (Figure 4.14). The result of applying our detector is shown in Figures 4.12 and 4.13. The two tables were correctly detected and the system was able to detect 5 chairs out of 7, although including one false positive. An additional experiment was carried out in a seminar room containing a big table and 15 chairs around it. The resulting classifications are also shown in Figures 4.12 and 4.13.

Data acquisition using our laser takes 10 s. Smoothing, segmenting and extracting the features takes another 7.12 s on average using C++ code on a single core. The time needed to select the final hypotheses for an object center mainly depends upon the total number of parts in the scene and the number of clusters in the vocabulary (since each of the parts should activate the closest clusters). It also depends on the number of final selected hypotheses because they are checked for inconsistencies (see 4.2.4.2). Using our current configuration, and without any optimization, our approach took 0.20 s for the office, and 0.24 s for the seminar using Matlab on an Intel Core i5.

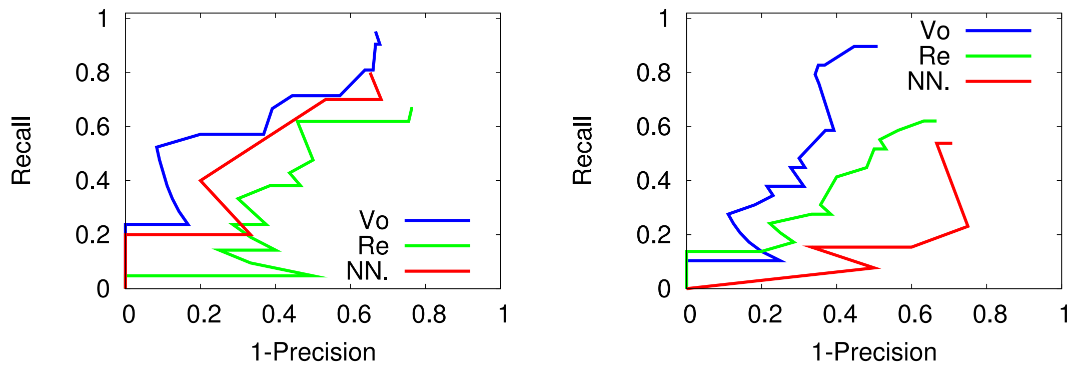


Figure 4.14: *Classification results for the office (left) and seminar (right) using our approach (Vo), nearest neighbor (NN), and a reduced set of features (Re). More details are given in the main text.*

To quantitatively evaluate the performance of our approach, we generated recall vs. (1-precision) plots for the two previous scenes. Different false positive values were obtained by increasing the threshold for the final hypotheses selection. A true positive is obtained when the distance between the detected center and the original center is less than half of the width of the object. The test set included two scans of the office and two scans of the seminar obtained from different viewpoints. The resulting plots are shown in Figure 4.14. In each plot we additionally compare our approach based on a vocabulary of parts (Vo) with two simplified versions: a version in which only the nearest neighboring part in the training data is used for voting (NN); and a version with a reduced number of features (ignoring features 4, 7 and 8) (Re). This last version aims to show the importance of all the features in the final classification. In both scenes our approach outperforms the simplified ones.

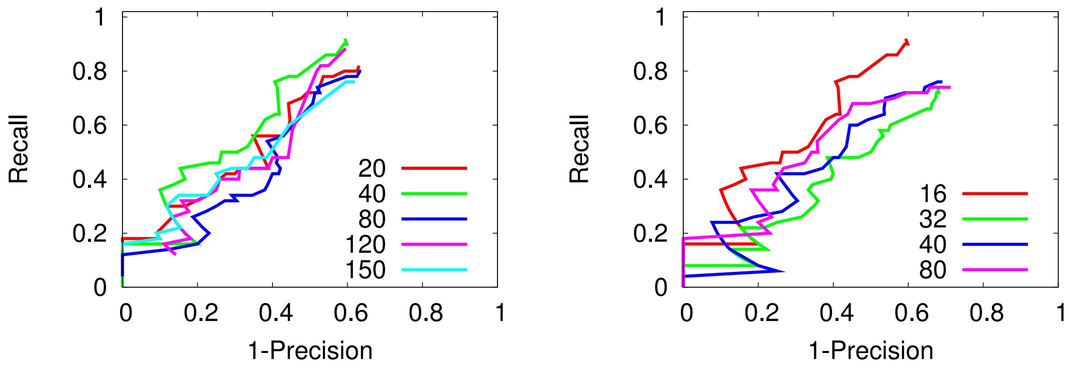


Figure 4.15: *Left: Classification performance using different number clusters k and a fixed number of 16 views. Right: Classification performance using different number of views and fixed number of clusters k=40.*

We also analyzed the influence of the vocabulary size in the classification performance. Results are shown in the left image of Figure 4.15. Finally, in the right image of Figure 4.15 we analyzed the performance of the classifier when increasing the number of training views. Interestingly, using 16 views for training gives slightly better results than using more. This could be because the segmentation and features produce similar patches. Improving the feature extraction is on our agenda, thus exploiting our classification to the fullest. However, the results are quite similar for the different parameters.

4.2.5.2 Model Fitting and Verification

In this section we present results on applying model fitting for the final verification of the detected objects. In the first experiment we used the classification results obtained in the previous section for the office and seminar environments. The classification outputs consist of the center of the detected objects together with the parts that contributed to its detection (see Figures 4.12 and 4.13). Using these outputs we fitted the best original CAD model for each final hypothesis using the corresponding retrieved parts and verifying the fitting as described in 4.2.4.3. The results are shown in Figure 4.16.

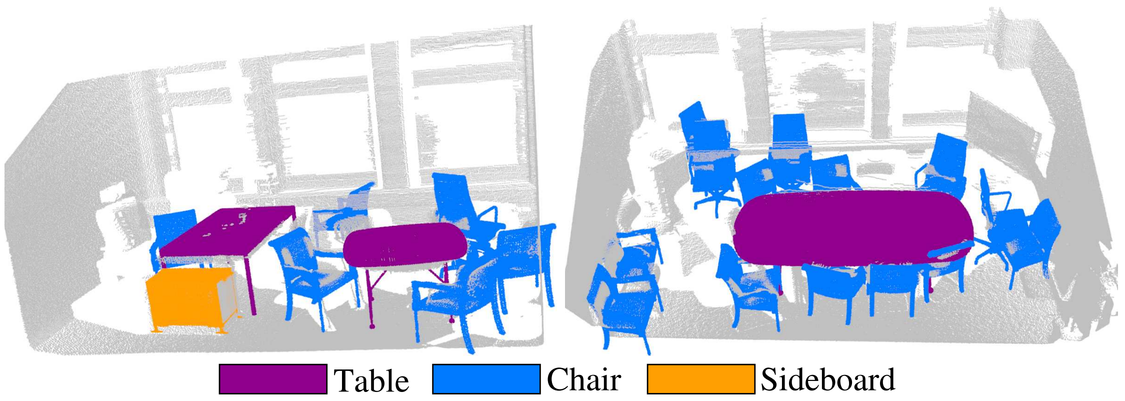


Figure 4.16: *Model fitting in the scenes from Figures 4.12 and 4.13.*

The poses are good for most of the objects and a subsequent ICP step would improve them further. However, here we focus on the possibility of using it to reject false positives. The two false positives in the right side of the seminar room were indeed removed, but the four chairs in the walls of the office and seminar room could not be filtered, as they covered the parts (which were quite small) well and were in occluded space. These could be filtered for example by having a 2D floor map available, as they are outside the boundaries of the rooms. False positive rejection is also shown in Figure 4.17.

There were four occasions where the fitted models were incorrect. One chair in the office and one in the seminar room are oriented backwards because their seats were occluded and the best matching CAD model that was available fits the data best when it is rotated in the wrong direction. Because of the large occlusions, these orientations could not be filtered. Two other chairs in the left part of the scans had very small parts (31 and 32 points), rendering correct matching impossible. In the office the bad orientation was preserved due to the high occlusion, but in the seminar room it was rejected.

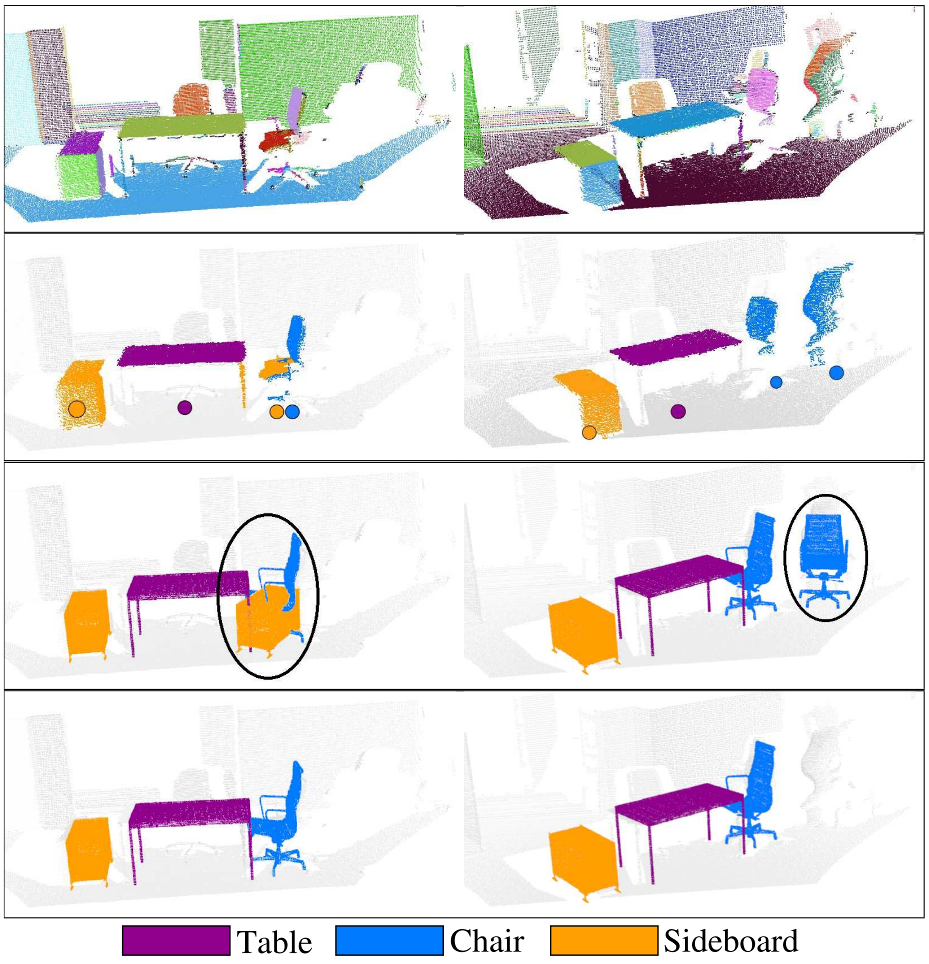


Figure 4.17: *Two examples of the whole process.*

4.2.5.3 Recognition Using Multiple Views

In the following experiment we showed how to take advantage of multiple views of the same scene to alternatively improve the final detection of objects. We used the two scenes of Figure 4.17 and applied the voting approach on each of the scenes as in the previous experiment. We then translated the resulting voting spaces into a common coordinate system. This process is equivalent to the alignment of the two scenes and the simultaneous voting in both of them. As shown in Figure 4.18, after merging both voting spaces the system is able to reject both false positives detections: the sideboard sharing place with the chair in the first scene; and the shadow of the man in the second scene (see previous experiment). The improvement in the recognition is a consequence of the simultaneous accumulation of evidence from both views. Correct detected objects received a much higher weight since they are detected twice, whereas false positives objects get evidence from only one view.

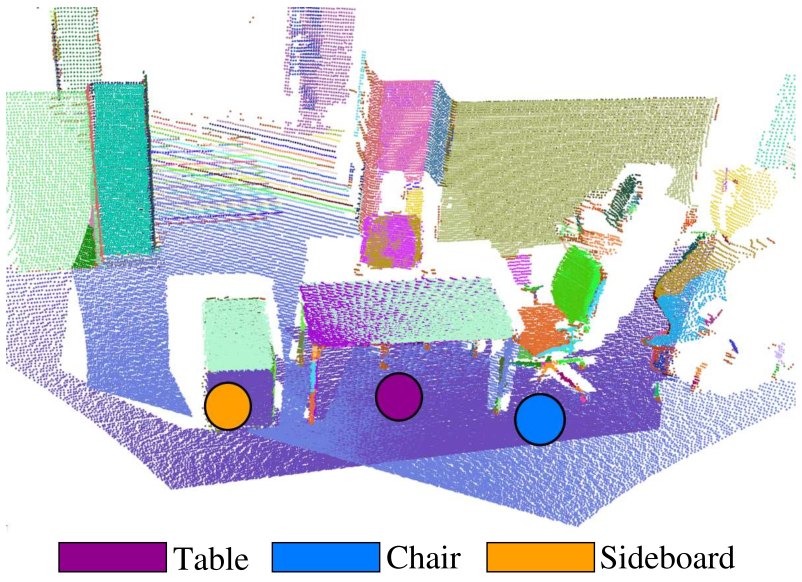


Figure 4.18: *The two scenes from Figure 4.17 are shown in a common coordinate system, along with the detected object centers after merging the corresponding voting spaces (see legend for center colors).*

4.3 Discussion

If maps are to be used for more than navigation and mere obstacle avoidance, a semantic interpretation of the observed scenes is required, which necessitates a meaningful labeling of objects appearing in mapped scenes. This chapter presented an approach that allows a robot to achieve this by learning models of objects downloaded from the Web and using them to detect and localize instances of these types of objects in new scenes represented by 3D point clouds.

Nowadays, typical indoor objects like chairs, tables or sideboards are found all over the world and globalization makes it likely to find the same objects in different places.

The ultimate goal would be to enable a robot to download 3D models of objects from the WWW and to learn general representations of them in order to be able to recognize similar objects in a new environment somewhere else in the world. Taking the idea one step further, robots could also map their environments, and the point clouds corresponding to high-scoring detections could be used for training and verification or could even be shared with other robots, thus enriching existing databases by distributed world modeling and experience-based learning.

What is needed, however, is the initial set of training examples, in an easily accessible way. As previously pointed out by other authors (Lai and Fox, 2009), one of the main problems when working with 3D point clouds is the availability of labeled training data. Model representation needs to be enriched as well, to account for different measurement units and orientations. Not to mention the problems arising from different data formats. Fortunately, there are efforts to solve these issues, and additional data can be obtained by shape morphing – e.g. using techniques such as the one from Hillenbrand (Hillenbrand, 2010).

In conclusion, by taking advantage of the explosion in the number of WWW resources that is driven by industry, we can avoid instrumentation to a large degree and allow robots to act successfully in unknown environments. The successful approach to capture information on part relations and multiple views, by accumulating them in a robust voting approach for ensemble learning, will be explored further in the next chapter.

Chapter 5

Detection of Objects in Complex Scenes

Eventually everything connects – people, ideas, objects... the quality of the connections is the key to quality per se...

CHARLES ORMOND EAMES, JR

We saw in the previous chapter how part-based learning of the overall structure of objects from online databases can aid the detection of similar ones in real-world scans. Additionally, the actual 3D models can be used to verify the detections to filter out false positives, and to obtain a reconstruction of the scene. In this chapter the lessons learned from environment modeling will be applied to the more difficult case of object detection in cluttered scenes. New difficulties arise since the possible object poses are not restricted to be always upright, there is typically greater variance of smaller objects in a scene, and the assumption on the planar supporting plane does not always hold, or is more difficult to detect. Additionally, there might be no well matching model in the CAD database, and thus the final 3D shape has to be reconstructed from the partial and noisy real-world data.

Therefore, in this chapter we introduce novel ways of dealing with these challenges of scene understanding, relying heavily on the unsupervised learning (and detection) of part structure. Large online databases are used as training data here as well, but as we will see, the usefulness of the learned information is enhanced by the right combination with domain-specific knowledge. To capture the topology of object parts a graph-based approach is developed that facilitates efficient (approximate) subgraph isomorphism checks through the use of additive geometric features and hashing. Information from multiple segmentations and views is accumulated and geometric object models fitted to the data. This allows the robot to obtain a CAD-like reconstruction of the scene and plan grasps for the objects to execute its task. During the complete process we will see that the quality of the connections between the points is indeed an important measure of the overall quality of the reconstruction.

5.1 Part-based Multi-modal Categorization

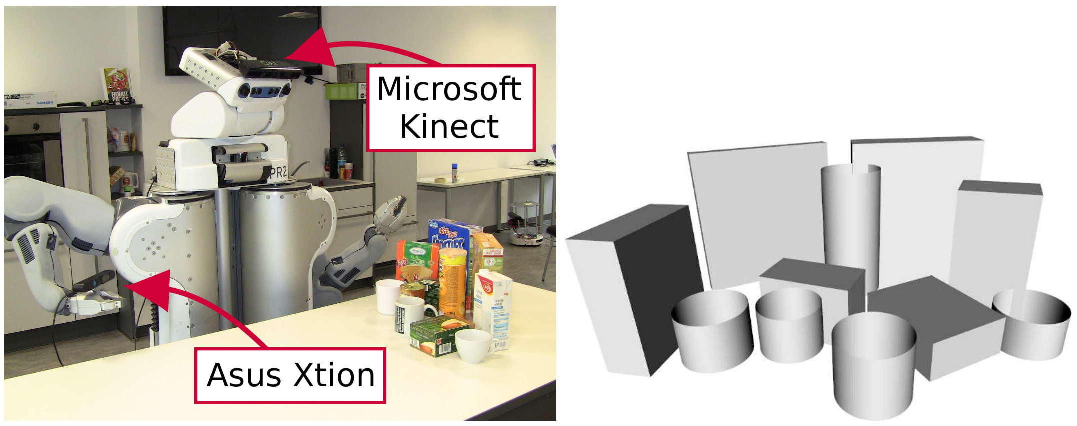
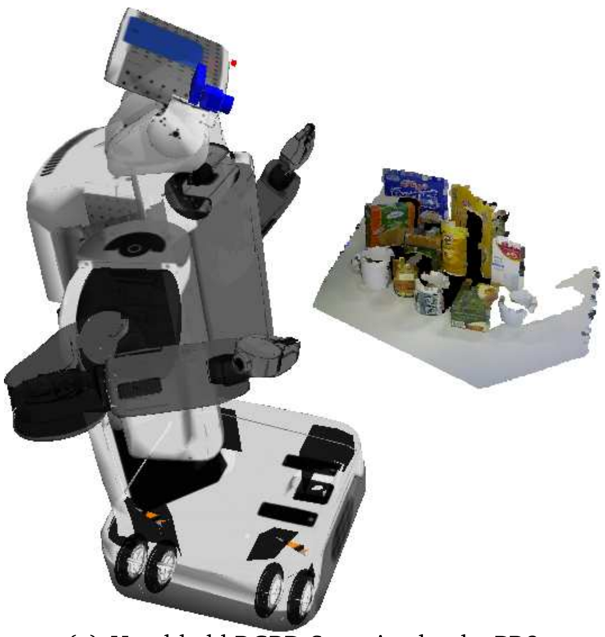
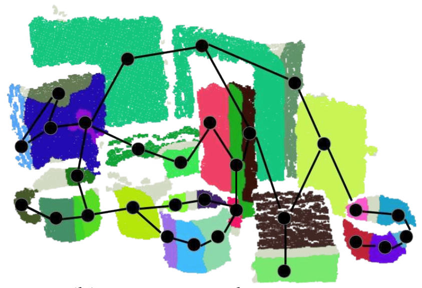


Figure 5.1: *Multi-cue perception system enabling robot to interpret complex scenes of unknown objects. Left: PR2 mobile manipulator equipped with RGBD cameras used in our experiments. Right: geometric reconstruction based on the part-based categorization.*

In this section, we present the application of our multi-cue perception system to enable a robotic household assistant to deal with detection of general geometric categories of previously unseen objects in clutter, as depicted in Figure 5.1. Since the robot could encounter new objects during its operation, no matter how large a training database is, geometric (edge-based or 3D) categorization and perceptual grouping offers specific advantages over instance-level (template-based) recognition (Dickinson, 2009; Marton et al., 2011). Additionally, we present how to improve detection results through multiple views of the scene, multiple interpretations of the data, learning the part structure from online databases and locating these through subgraph matching, in a recognition-by-components approach (Biederman, 1987). The main steps of the process are shown in Figure 5.2, and described in (Marton et al., 2013, 2014).

In the previous chapter we found that a part-based approach lends itself easily for solving such problems. Learning the different parts and their combinations is a scal-


(a) Hand-held RGBD Scanning by the PR2


(b) Scene-graph Creation

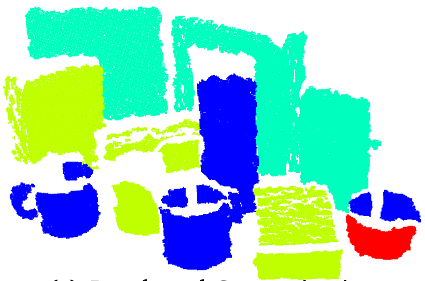

(c) Part-based Categorization

Figure 5.2: *Categorization in clutter using part-graph hashing. The left image shows our service robot PR2 equipped with a hand-held RGBD camera, capturing a 3D scan from a table-top scenario. The images on the right show the process of categorization of the different segments in the scene. On the top, the scene is over segmented and a graph is created with neighboring segments. On the bottom, the final categorization results is shown where cylinders are marked with blue, boxes with yellow, rectangular flat faces with cyan, and (half) spheres with red.*

able way to capture the different object categories a service robot could encounter in a domestic environment. For example, a mug is typically a cylindrical part next to a handle, or a teapot is a combination of different rounded shapes. This idea is supported by research on visual perception (Biederman, 1987), and by our previous results on part-based object categorization (Marton et al., 2009c; Mozos et al., 2011; Marton et al., 2012a).

Similar ideas were explored also in purely image-based approaches as well, but these may fail for textureless objects, or under bad lighting conditions for example, as seen in Figures 5.3 and 5.4. However, our geometric-based part categorization approach lends itself easily for solving such problems.

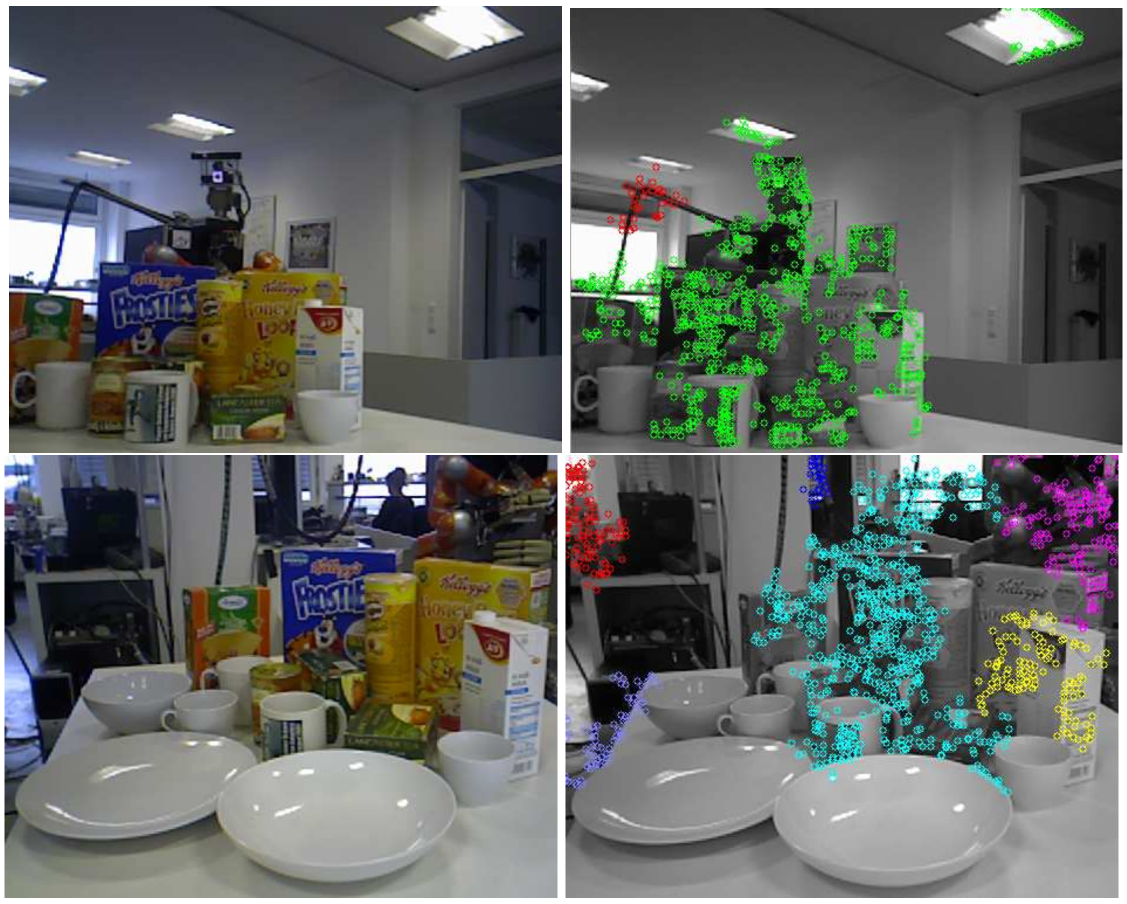


Figure 5.3: *Clustered keypoints detected using the SIFT-based ODUfinder (Pangercic et al., 2011a), showing the difficulties encountered by texture-only methods.*

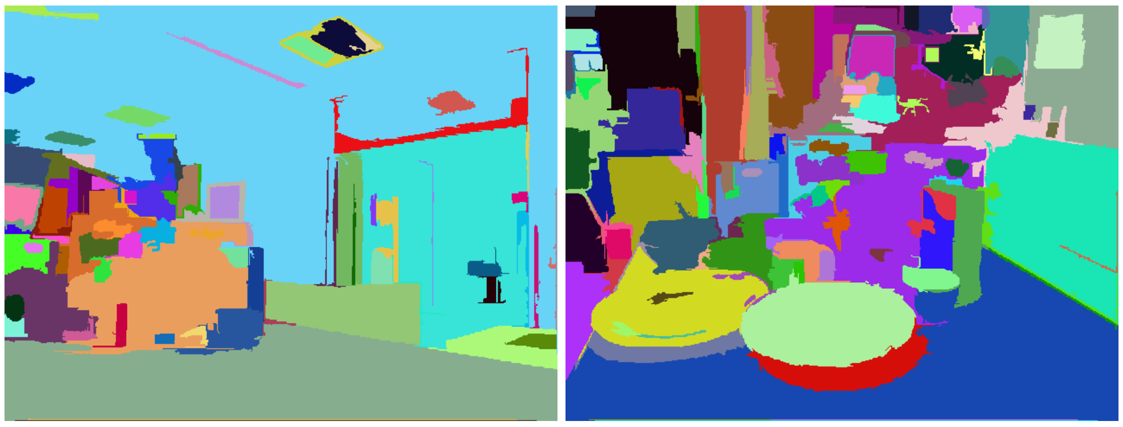


Figure 5.4: *Image-based segmentation results of cluttered scenes using the method by Felzenszwalb and Huttenlocher (2004).*

In Lai and Fox (2010) the authors also propose a similar system for understanding cluttered scenes, in that it performs part-based categorization in cluttered scenes. We combine the over-segmentation from (Mozos et al., 2011) with an extended version of creating multiple groupings of these “parts” (Lai and Fox, 2010), and present our approach that was designed to handle multiple instances of objects from several categories, that were labeled according to their general 3D shape.

To be able to deal with more complex shapes and greater variance in pose than in the case of furniture pieces, we identify to what object does each part belong by considering its descriptor and that of neighboring parts, together with the local topology of the scene. In this sense, it is an improvement over the vocabulary of parts and simple vote accumulating approach from the previous chapter. In an extension of this work, classification of part groupings (and a better part categorization) was found to improve performance in the case of furniture pieces as well (Usenko et al., 2012).

Our classification is less complex than the one presented by Lai and Fox (2010) and still manages to capture relations between parts in the “soup of segments” approach of (Malisiewicz and Efros, 2007) and can be used to achieve similar effect as their “domain adaptation”, as we will show in our experiments. Additionally, we trained on the same number of categories, but with multiple objects per category by grouping objects based on their geometry, and described the topology of the segment groups by graph-theoretic properties in order to improve and speed up classification through hashing.

Since multiple objects of the same type could be located in close proximity, assigning the correctly identified parts to two separate objects is difficult using this method and the one from Lai and Fox (2010). For this step the spacial relations between identified parts and the object center and the geometric validation step from (Mozos et al., 2011) is needed, for which CAD models would be needed, however. Therefore, we incorporate a general geometric model reconstruction method that takes the output of the categorization and produces completed (but simplified) 3D models that are directly usable for grasp planning, as we will detail in Section 5.4. The whole process is designed such that it can handle cases where objects are only partially visible and that it assigns a label to each detected part, in contrast to methods like (Lai et al., 2011a,b).

Detection of small objects in clutter using a sliding window was explored in (Kanezaki et al., 2010) using an additive feature. If a feature is additive, the descriptor that would be computed for the object is the same as the sum of the features of its parts. Thus it is especially useful for detecting objects based on features computed only for parts of it, for example by using the Linear Subspace Method (LSM) on the feature space, as presented by Watanabe and Pakvasa (1973). However, the Linear Subspace Method does not exploit the relations between the different parts of the objects. We used the additive property of 3 features (GRSD- (Marton et al., 2012a), C^3-HLAC (Kanezaki et al., 2011a) and VOSCH (Kanezaki et al., 2011c)[1]) to compute the descriptor of grouped parts by summing up the parts' descriptors. The advantage of additive features for our part-grouping method is that we only need to create the descriptor for each part, and all the possible part combinations can be described by the sum of the features of the constituent parts much more efficiently than computing a feature for each combination. We compare our classification method to that presented in (Kanezaki et al., 2010) and (Mozos et al., 2011).

For our experiments we used the large RGBD Dataset from Lai et al. (2011a) and created groups containing different categories of them, similarly as in (Marton et al., 2011), and validate the choice of our geometric categories. We evaluated our geometric categorization approach on RGBD scans of cluttered tabletop scenes of previously unknown objects, and experimented with enriching the training set by combining different databases. Geometric, color, and combined features were compared and the advantages and disadvantages contrasted. As was shown previously by Lai et al. (2011a), geometric features are more appropriate for categorization, while image features for instance recognition. However, their results suggested that overall, color/texture features are more suitable for categorization as well. In contrast, in our experiments we employ geometric, color, and combined features for categorization and find that color features do not generalize as well as geometric ones.

[1]VOSCH and GRSD- are rotation invariant, and all three features work on general 3D data, not only depth images. Therefore, the presented method can be applied to other types of data as well, like registered scans for example.

5.1.1 Scene- and Part-graphs for Object Recognition in Clutter

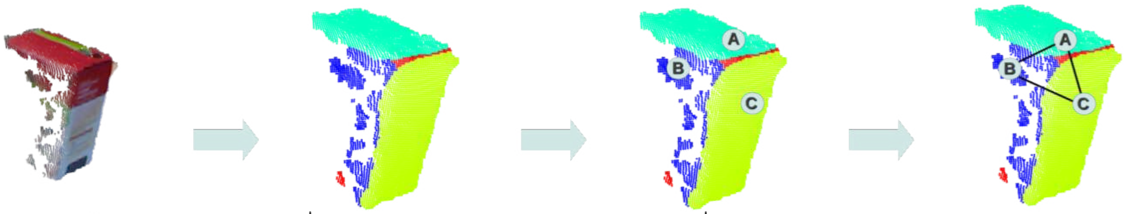
Our geometric categorizations' basic idea is that segmenting objects accurately does not always work robustly and will result in labeling mistakes, but over-segmentation is easily realizable (Malisiewicz and Efros, 2007; Lai and Fox, 2010). We use the normal-based segmentation criteria presented in Section 2.4 to over-segment the scans, such that patches with a relatively small curvature are considered, as shown in Figure 5.2. The obtained segments represent only a sub-part of objects but can be used to compute features, and combined to build up object candidates.

There are of course multiple ways of combining parts and not all of them create a valid object. However, we can test if a combination is valid by checking if the combined feature vector is known. We also exploit the fact that parts and their connections (neighborhood relations) can be treated as a graph, and only certain types of sub-graphs are present in the graph formed by the parts of an object.

We extract the part neighborhoods by checking if the physical distance between two parts falls below a threshold, and build a connectivity matrix. Starting at each vertex, we then create all the possible groupings up to a certain size (number of regions in the grouping) in order to obtain our "soup of segments" and create a hash code for each of the groups, as shown in Figure 5.5. Note that since the graph vertices can be sorted, it is possible to efficiently enumerate all sub-graphs containing a given vertex without repeating already generated ones.

As motivated in the previous chapter, the over-segmentation into parts does not have to be completely reproducible, given a robust voting method that creates an ensemble of the (possibly inaccurate) part categorization results. In a typical scene consisting of around 10^5 points, this method created around 50 segments, and over 100 groupings of parts.

Checking for subgraph isomorphism is not practical, but there are several descriptors one can employ to rule out isomorphism. Thus, during training we decompose our objects into parts, compute the features for each part, build the part-graph, and generate all sub-graphs along with their combined features. These features along with the object's class are then saved in a multi-level hash table, where the keys are: i) the number of parts, and ii) the identifiers of the subgraph topology. When testing, the



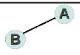
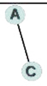
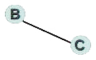
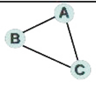
Groupings	Arrangement Key	Descriptors
A	1	$d_a = (f_a^1, f_a^2, f_a^3 ... f_a^n)$
B	1	$d_b = (f_b^1, f_b^2, f_b^3 ... f_b^n)$
C	1	$d_c = (f_a^1, f_a^2, f_c^3 ... f_a^n)$
	11	$d_{ab} = d_a + d_b$
	11	$d_{ac} = d_a + d_c$
	11	$d_{bc} = d_b + d_c$
	222	$d_{abc} = d_{ab} + d_c$

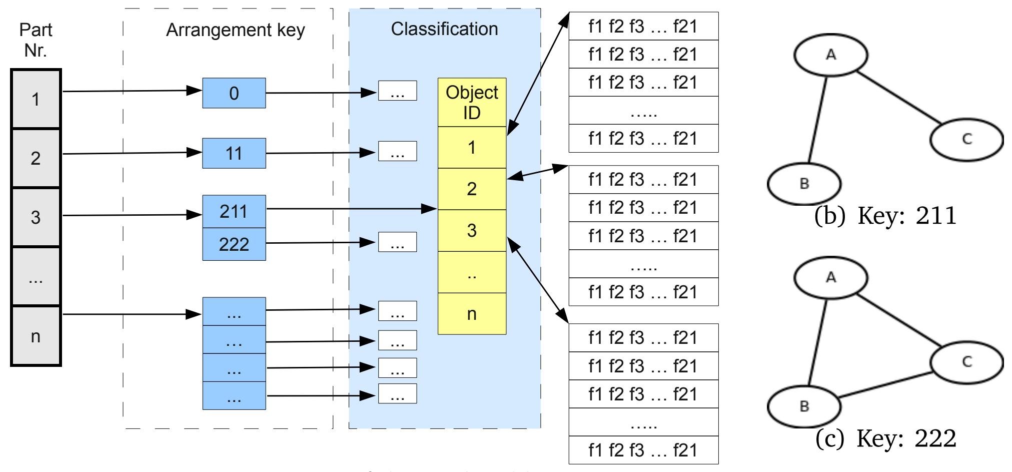
Figure 5.5: *Overview of part-graph hashing (using a single object, as during training)*

procedure of decomposition and part-graph building is repeated, and starting at each part, all the subgraphs are grown that are not larger than anything seen during training. These are then classified for which objects can they be parts of and similarities are accumulated in the source part for final classification.

In the following, we will detail the hashing procedure and analyze the selection of appropriate geometric classes.

5.1.2 Object Part Hashing

As mentioned before, the summed-up training features are saved in a hash table, a classifier is built for the training exemplars in each entry, and the classifier to be used for a testing features are looked up in it based on the two keys. The structure of the hash table is shown in Figure 5.6.



(a) Structure of the Hash Table

(b) Key: 211

(c) Key: 222

Figure 5.6: *Structure of the hash table and two different arrangements of three parts.*

Since the hashing procedure partitions the training data, training classifiers for each key is computationally less expensive and speeds up classification time as well. For the hash codes, we use the the number of vertices/parts at the first level, and at the second level we concatenate the sorted list of vertex degrees. This degree order is unique for isomorph graphs, however different graphs can have the same degree order. The same property holds also for the eigenvalues obtained by spectral analysis of the sub-graph's connectivity matrix, but during our experiments we found no significant increase in performance when using them (or combining the two keys).

The part grouping process can be applied for training data, where the scan contains a single object, or to test data containing a complete clutters scene. The task is then to classify the subgraphs created for a test scene using the part-graphs created during training. During lookup, the same hashing procedure is repeated for the query, and the part grouping's feature is classified using nearest neighbors classification. This allows for fast training and testing times, but more powerful classifiers from the repertoire of our perception system could be used (see Section 3.1.4).

The obtained probability distributions are accumulated in the constituent parts, using weighted voting (see 3.3) assigning lower weight to larger groups. In contrast to the product of the class probabilities for each grouping that was used in (Lai and Fox, 2010), we found that the (confidence weighted) voting approach performs better. Similar findings supporting voting were made by Lam and Suen (1995) when evaluating combinations of classification results. For efficiency, if a certain group of parts does not produce high-enough scores, its growing can be discontinued. Similarly, if a class gets very low scores it can be discarded for the sub-graphs grown from the current grouping. While the runtime performance improves by using these cut-offs, we found better performance if all scores are computed and considered.

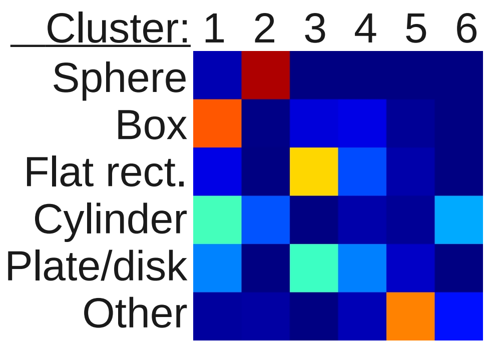
If there is a known assignment of parts to objects, like in the experiments described in 5.2.3, or following a Hough voting procedure for identifying the object center as in (Mozos et al., 2011), the final distribution over the category labels can be obtained by merging those stored in the part.

5.1.3 Geometric Categorization

As described earlier, for each grouping we compute one of the additive features. These features are then used to classify the parts as forming an object of the following geometric categories: *sphere*, *box*, *flat rectangle*, *cylindrical*, *disk/plate*, or *other*. These intuitive categories match most of the objects for which we had appropriate training data (and the remaining ones were assigned to the *other* category), and also the categories we found in public household objects databases (Marton et al., 2011), e.g. KIT object Models Web Database[2], Household Objects Database from Willow Garage[3], and TUM Semantic Database of 3D Objects[4] (a compressed version can be seen in Table 5.10). As in our previous works, the categories are given by human intuition, but results using unsupervised clustering show that they make sense also based on the data.

We used the recently introduced Regularized Information Maximization (RIM) technique (Gomes et al., 2010) to find meaningful clusters of our training data (see Section 5.2) and assign testing instances to these clusters in the GRSD- and VFH feature spaces. Since there is no standard way for evaluating clustering without ground truth, as it depends on the application, we measured how well do the clusters overlap with

[2]http://i61p109.ira.uka.de/ObjectModelsWebUI/
[3]http://www.ros.org/wiki/household_objects_database
[4]http://ias.in.tum.de/software/semantic-3d



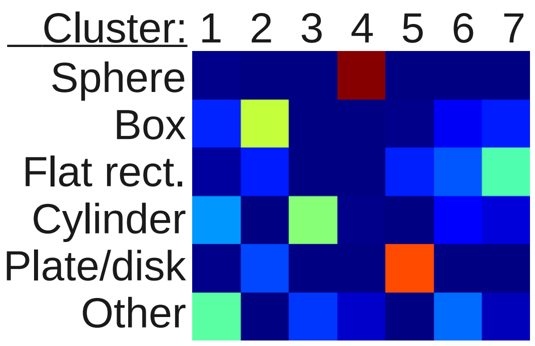


Figure 5.7: *Unsupervised clustering results using RIM compared to the manually defined geometric categories (left: GRSD-, right: VFH). Clusters overlap well with the used categories, with two geometrically similar pairs merged using GRSD-. However, in the higher dimensional VFH feature space these can be distinguished.*

the given categories by computing the Adjusted Rand Index (ARI), using different parameters.

For GRSD- the best ARI (0.36) is obtained using 6 clusters and $\lambda = 90$, with stable results around these values. As shown in Figure 5.7 (left), the clusters are quite clean, and also the categories are grouped nicely with clusters, but cylindrical objects were merged with boxes and flat ones with plates. This makes sense given that GRSD- encodes only relations between neighboring voxels, thus features like the contour are not captured. Additionally, small boxes and cylinders can look quite similarly in Kinect scans, especially after smoothing. However, we chose to keep these two pairs as separate categories as they are semantically different (using a different feature they still could be separated) and provide relevant information for model fitting and grasping applications.

Using VFH these clusters could be separated, thanks to the increased descriptiveness given by the higher dimensionality and viewpoint variance. Here the best ARI (0.42) is obtained using 7 or 8 clusters and λ between 75-80, but the results are not as stable as in the case of GRSD-, suggesting that the clustering depends very much on the random initialization.

In both cases, smaller clusters are created as well, into which parts of the object categories are separated, suggesting that some views of object instances from a category could be grouped together (e.g. side and front views of flat rectangular objects like cereal boxes). Such a strategy was used in (Marton et al., 2011) to increase the geometric categorization accuracy.

5.2 Large-scale Evaluation and Comparisons

Sphere	Box	Flat (rectangle)	Cylindrical	Plate (disk)	Other
bowl (6)	food box (4)	notebook (5)	coffee mug (8)	plate (7)	cap (4)
ball (6)	sponge (8)	cereal box (5)	food cup (3)		kleenex (5)
		food box (8)	soda can (4)		pitcher (3)
			food can (14)		
			food jar (6)		
			water bottle (6)		

Table 5.1: *Selected object categories from the RGBD Dataset with good 3D data, grouped into general geometric categories ("RGBD-Large" set). The number of objects in each category is given in parentheses, with the total number of scans being roughly 80000, of which every fifth is used.*

For our tests we used the large RGBD dataset from (Lai et al., 2011a), which contains a total of over 200,000 scans of 300 objects from 51 object categories. As in (Lai et al., 2011a), we use every fifth point cloud from the dataset in our experiments, because the similarity between consecutive point clouds is extremely high. Since in this work we focus on 3D classification, we selected those object categories that have good 3D data (and excluded very small, shiny or transparent objects) and grouped them into geometric categories as shown in Table 5.1. Please note that the "food box" category contained both regular boxes and large flat boxes, so we split them up accordingly.

Sphere	Box	Flat (rectangle)	Cylindrical	Plate (disk)	Other
ball (6)	food box (4)	food box (8)	coffee mug (8)	plate (7)	cap(4)
	sponge(8)		food cup (3)		pitcher(3)
			soda can(4)		

Table 5.2: *Reduced set of objects from the selected categories ("RGBD-Small" set). The number of objects in each category is given in parentheses, with the total number of scans being roughly 30000, of which every fifth is used.*

In order to be able to test and compare our method and features, for some of the more time-intensive tests we reduced the dataset from Table 5.1 to roughly 7000 scans of 57 objects from 9 object categories, presented in Table 5.2. For generating a large amount of labeled test scenes, we combined test scans from the dataset to generate realistic cluttered scenes. Additionally, we used the dataset from (Kanezaki et al., 2011c) for some of the tests to provide the classifier with knowledge about the objects in our environment. Real cluttered scenes were tested on several combinations of the training datasets, presented in more detail below.

5.2.1 Complete Cluttered Scenes

In this subsection we present categorization results of several cluttered table-top scenes containing different objects. Since labeling these scenes is a time-consuming process, we could evaluate only a couple of them. We present results on 3 frames in this subsection, and a sequence of 6 scans of a fixed scene will be used in the next section. Figure 5.8 show three tabletop scenes on which we tested our approach. The color red represents the *sphere* class, blue *cylinders*, yellow *boxes*, and cyan the *flat* class.

Testing on the cluttered scenes was run using different datasets (or combinations) as training data, as shown in Tables 5.3 and 5.4. As it is expected, result vary depending on the type of feature descriptor and on the training dataset.

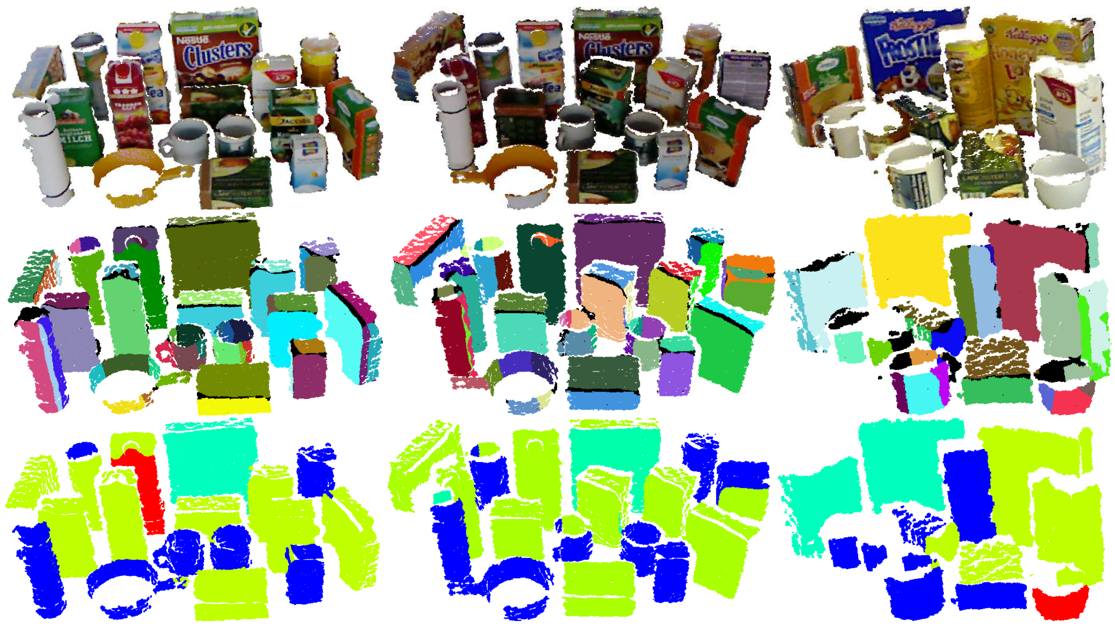


Figure 5.8: *Segmentation and geometric categorization on three cluttered scenes.*

In order to diversify our training data we combined the RGBD datasets with the "VOSCH" Kinect scan dataset (VDB) used in (Kanezaki et al., 2011c), consisting of 63 similar objects than in our scenes, captured from different viewpoints with an angular step of 15 degrees. Similarly to (Lai and Fox, 2010), we found that this "domain adaptation" improves results, as seen in Table 5.3. However, as the results on the larger RGBD dataset suggest, identifying the correct weighting of the two data sources is necessary, possibly based on an evaluation set. Apparently, as the number of objects increases, confusions get more frequent, therefore the weight of the domain specific objects need to be increased. In the case of the smaller dataset, the combination with the scans from VDB improved over the results on both separate training sets, highlighting the importance of mixing various sources of information while keeping specific specialties[5]. Related ideas are discussed by Horswill et.al. Horswill (1995) as well (task and environment adaptation improving perception capabilities).

Datasets:	RGBD-Small	RGBD-Large	VDB	Small+VDB	Large+VDB
Scene 1					
per point	73%	47%	76%	**84%**	54%
per segment	**83%**	49%	67%	**83%**	54%
Scene 2					
per point	76%	54%	70%	**80%**	58%
per segment	**76%**	41%	72%	74%	47%
Scene 3					
per point	70%	42%	78%	**88%**	70%
per segment	76%	45%	**84%**	80%	76%
AVERAGE					
per point	73%	48%	75%	**84%**	61%
per segment	78%	45%	74%	**79%**	59%

Table 5.3: *Results in clutter using different training datasets with GRSD-*

Lai *et al.* reported results on the comparison of visual and geometric features using the database presented in (Lai et al., 2011a). Their tests highlight the fact that geometric features are more suitable for categorization and visual ones for instance recognition, but they found that visual features outperformed geometric ones both at instance and category recognition, while a combination of both works best. Using our experiments

[5]Thanks to the hashing approach, handling large databases and dynamically adding new objects is alleviated, as only affected groups have to be re-trained.

Datasets:	RGBD-Small	RGBD-Large	VDB	Small+VDB	Large+VDB
Scene 1					
per point	29%	39%	**77%**	60%	55%
per segment	47%	43%	**75%**	64%	50%
Scene 2					
per point	26%	25%	**79%**	52%	40%
per segment	36%	27%	**72%**	50%	32%
Scene 3					
per point	73%	**81%**	45%	73%	**81%**
per segment	56%	**68%**	60%	56%	**68%**
AVERAGE					
per point	43%	48%	**67%**	62%	59%
per segment	46%	46%	**69%**	57%	50%

Table 5.4: *Results in clutter using different training datasets with VOSCH*

this was not the case, suggesting that their conclusion does not hold in every case. When using the color-dependent VOSCH feature, the fact that many of the test objects are from VDB becomes reflected in higher success rates, as shown in Table 5.4. However, these results are worse than the corresponding results using GRSD- and much worse than the best results obtained with the purely geometric feature (despite the large difference in dimensionality). We believe that the contradicting results are due to the fact that in (Lai et al., 2011a) some categories show little variation among the instances (at least with the employed features).

Run-times vary depending on the dimensionality of the extracted feature and the scale of the used dataset, with classification on the small VDB dataset using the only 20 dimensional GRSD- feature yielding the fastest results, due to the fact that the VDB contains only around 900 individual scans of objects. The classification times shown in Table 5.5 were obtained on a single core 2.4 GHz CPU.

For a more detailed evaluation, the next subsections will focus on large scale tests using the RGBD dataset, using separated objects as queries. The RGBD-Small set was split in a ratio of 2 : 1 into training and testing scans, except for the cross-validation test that was performed using the methodology from (Lai et al., 2011a). Given that the objects are already segmented, our approach can take advantage of the fact that only a single object needs to be categorized, and merge the results obtained for the

Runtimes using diff. datasets	RGBD-Small	RGBD-Large	VDB	Small+VDB	Large+VDB
GRSD- [20d]					
per point	2.4E-05	4.4E-0.5	0.41E-05	2.88E-05	4.76E-05
per segment	0.043	0.083	0.007	0.053	0.089
VOSCH [137d]					
per point	1.4E-04	2.3E-04	0.19E-04	1.6E-04	2.5E-04
per segment	0.27	0.43	0.03	0.30	0.47

Table 5.5: *Average classification times in seconds for the scenes from Figure 5.8*

different parts by weighting the label probabilities by the number of points in the part.

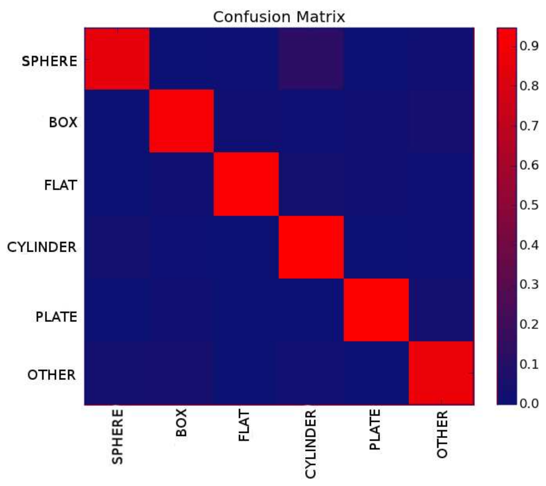
5.2.2 Evaluation of Features

GRSD- distance metric	Accuracy [%]
Manhattan	92.5
Euclidean	94.5
Jeffries-Matsuhita	95.5
Baseline (SVM – not part-based)	95.3

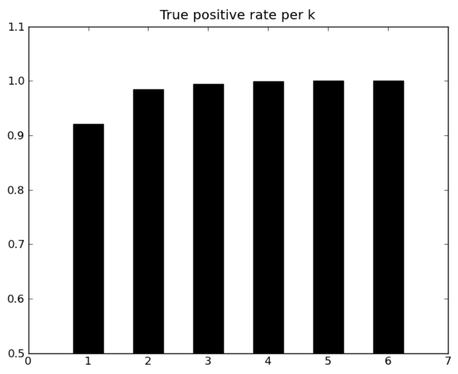
Table 5.6: *Classification results using different distance metrics as compared to the baseline obtained for GRSD-.*

We tested different distance metrics for nearest neighbors classification and found that the Jeffries-Matsuhita distance performs best, as shown in Table 5.6. Due to the hashing procedure, the separate classifiers for each hash key combination have an easier job in distinguishing parts coming from different categories. Thus, results are on par with that obtained with SVM with RBF kernel, but using a simple nearest neighbors approach, which has considerably shorter training time.

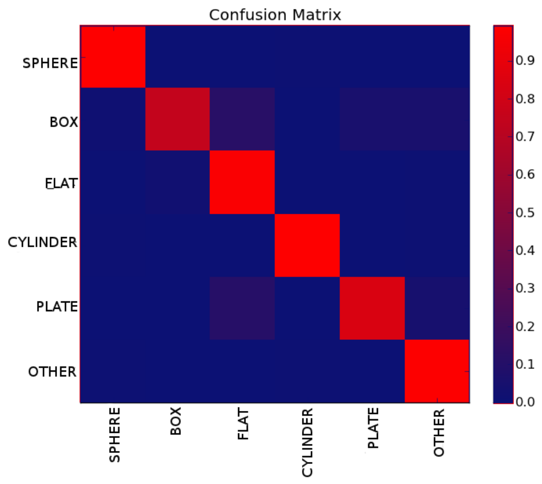
Here, the splits of the RGBD-Small dataset were used, to obtain an SVM classifier in practical time, meaning that the method was trained on different views of the objects encountered during testing. Additionally, the results were obtained by testing objects as separate clusters, thus giving considerably better performance than during testing on the cluttered scenes of unknown objects. This highlights the difference in difficulty between instance recognition and categorizing novel objects in difficult settings. For instance recognition RGB image-based information is more informative than 3D one, but this does not hold for categorization, not even for categorizing complete objects as we will see below.



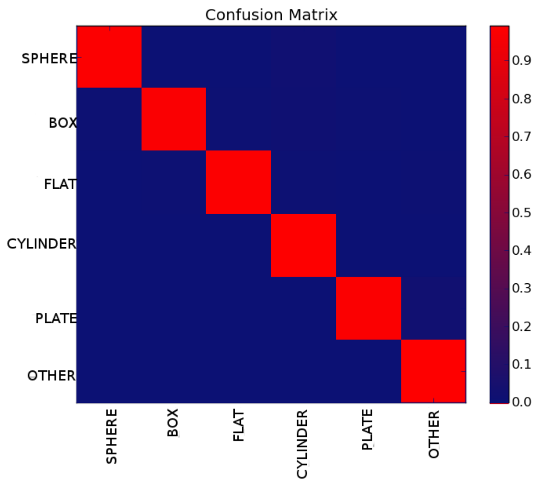
(a) GRSD-



(b) Cumulative result GRSD-



(c) VOSCH



(d) C^3-HLAC

Figure 5.9: *Confusion matrices and cumulative score on the RGBD-Large set. Please note the different scale of the confusion matrices used for better visualization.*

To show this, the results obtained by our method on the RGBD-Large dataset using GRSD-, were compared to the C^3-HLAC and VOSCH additive features. For this test,

	GRSD- [%]	C^3-HLAC [%]	VOSCH [%]
RGBD-Large	92.1	98.48	94.59

Table 5.7: *Categorization results using different features on the RGDB-Large set*

we left out every third scan of an object from the training set, and used it for testing. Results are shown in Figure 5.9 and Table 5.7, with an interesting observation relating to (b): the two most likely results are by 5% better than the ones reported as most likely. This suggests that in case we obtain similar top scores, re-segmenting the test scene (with different random seeds) could improve the labeling, by merging the votes from different segmentations. This approach was employed in the next section in the case of different views.

We also compared GRSD- to rotation-invariant C^3-HLAC (Kanezaki et al., 2011a), and their combination VOSCH (Kanezaki et al., 2011c) using the Linear Subspace Method. In the training process of LSM, we divided the whole voxel grid of each object segment into cubic subdivisions of a certain size (14 cm × 14 cm × 14 cm with 10 cm × 10 cm × 10 cm overlapping in our case) and then extracted feature vectors from all of the subdivisions to perform Principal Component Analysis. In the testing process, we extracted one feature vector from the whole voxel grid of each object segment. Note that we do not need to divide the voxel grid in the testing process, because we can calculate the similarity value of the query object to a reference object by simply projecting its feature vector to the subspace, owing to the additive property of the used features.

The reason why GRSD- performed poorly compared to the features that captured color information, is that in these tests scans in the training and testing set came from different views of the same object. Therefore, we also performed a cross validation experiment (following the methodology in (Lai et al., 2011a)) to test how well these additive features generalize to unknown objects. See Table 5.8 for results. As it was to be expected, the purely color based C^3-HLAC feature performs worst (except for the typically white plates), with an average success rate of 59.19%. The VOSCH feature is aided by its geometric part, and achieves 70.88%, while in this experiment GRSD- performed best, with an average of 72.06%. There are large variations in the success rates for certain classes, due to the fact that we used only the RGBD-Small set, and

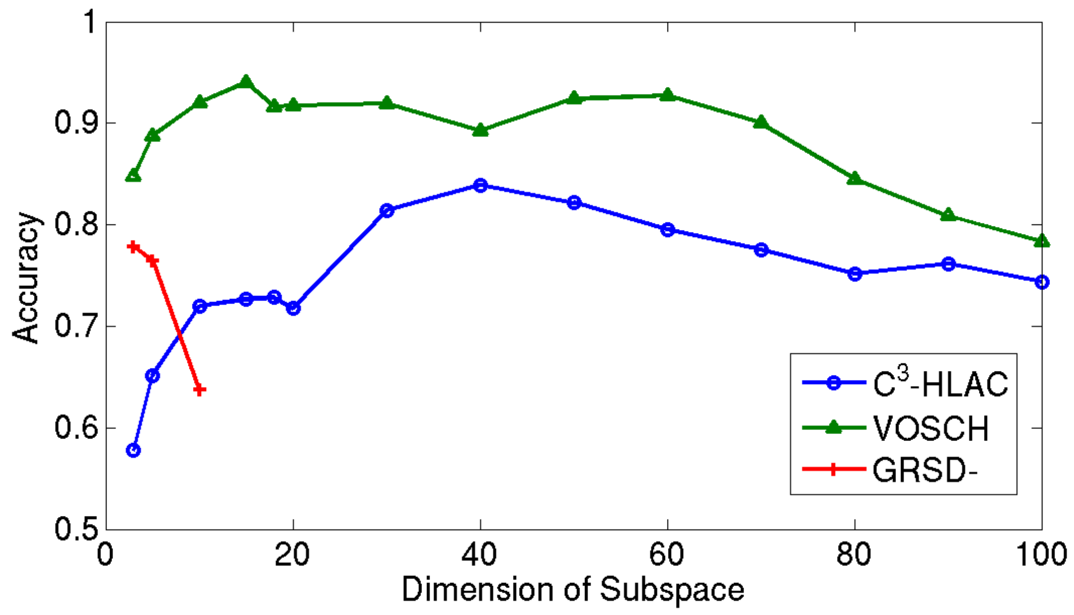


Figure 5.10: *Accuracy vs. number of subspace dimension using LSM. The red cross shows the result with GRSD-, blue circle with C^3-HLAC and green triangle with VOSCH.*

some classes are more varied than others. Still, the smallest variance is produced by the RGB-insensitive GRSD- feature descriptor.

	Sphere [%]	Box [%]	Flat [%]	Cylinder [%]	Plate [%]	Other [%]	**Total**
GRSD-	**67.5±26.2**	52.5±**15.2**	**95.8±2.7**	89.5±**2.6**	47.1±**22.5**	**79.9±14.2**	**72.06**
C^3-HLAC	53.0±28.8	32.1±17.6	80.2±8.6	77.4±10.7	**65.1**±32.2	47.3±22.7	59.19
VOSCH	63.2±27.2	**60.4**±25.2	90.6±7.2	**90.1**±9.5	50.9±27.7	70.1±24.5	70.88

Table 5.8: *Per category leave-one-out cross validation tests on the RGBD-Small set*

In conclusion, these tests underline the fact discussed earlier, that geometry is more stable between instances from the same category as color. This is also supported by findings suggesting that geometric knowledge (shape primitives and their spacial relations) have a modality-independent representation in the human brain (Yildirim and Jacobs, 2013).

5.2.3 Comparison to Previous Methods

In our previous work (Marton et al., 2012a) we performed a comparison to segmentation based categorization, by segmenting round and rectangular objects using the method from (Goron et al., 2012), and found a significant drop in accuracy due to segmentation mistakes. Since we consider multiple segmentation possibilities and the relations between parts, the results were more robust than for a single segmentation and global feature based approaches.

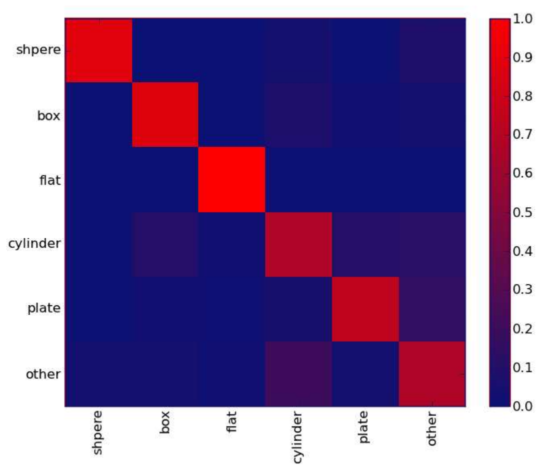
Here we compared our results to those obtained with the statistical features and method described in (Mozos et al., 2011), considering only the part voting step, without the geometric object (pose) identification, as CAD models and ground truth poses are not available for our objects. A vocabulary of size 400 was created out of the descriptors of the parts from the training dataset using K-Means, and used to assign class probabilities to parts in the testing dataset. These votes cast by the different parts are weighted by their similarity to the activated cluster, and the final class is assigned to the highest scoring one.

Both the statistical features used in the original publication and GRSD- were tested using this method, and we obtained a mean success rate of 80.45% for the former and 75.86% for the latter. As seen from the corresponding confusion matrices in Figure 5.11, the difference is due to the fact that the miscellaneous "other" class is handled considerably better by the statistical features – if this class is ignored, the two features give practically the same result. Since the original features are not additive, using them in the current method would require its repeated re-computation, something we would like to avoid. Moreover, some of the statistical features are orientation dependent, thus would require training objects in multiple poses.

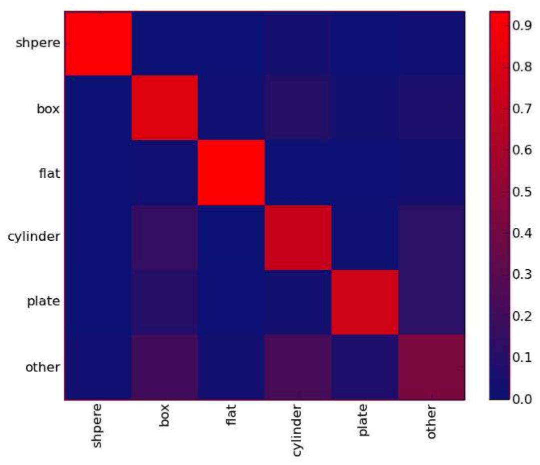
Our method and the sliding window based Linear Subspace method was also evaluated on the same data. Overall, the results indicate a clear advantage of the part-based categorization process, as shown in Table 5.9.

	Part-graph Hashing	Part Vocabulary (Mozos et al., 2011)	Linear Subspace Method (Kanezaki et al., 2010)
Success rate	95.5	75.9	77.8

Table 5.9: *Results using different methods on the RGDB-Small datasets*



(a) Statistical Feature



(b) GRSD-

Figure 5.11: *Confusion matrices of the vocabulary of parts method. Please note the different scale of the confusion matrices used for better visualization.*

5.2.4 Generated Scenes

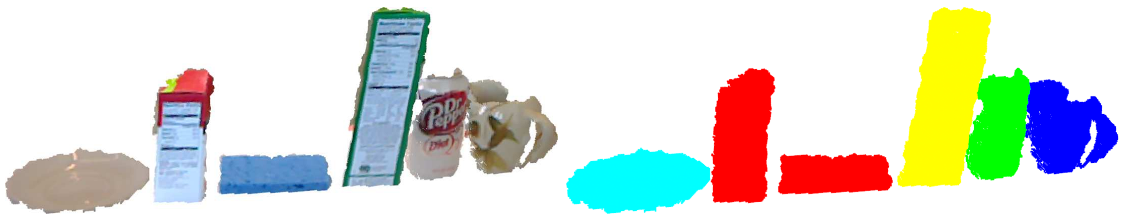
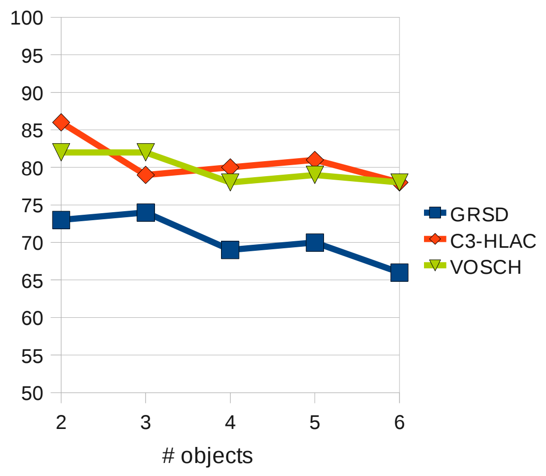


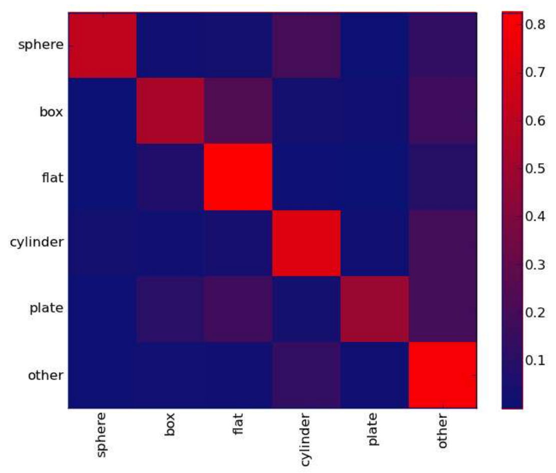
Figure 5.12: *Synthetic scene automatically created and labeled from individual scans.*

This subsection presents results on a large scale test on scenes containing touching objects (without occlusions). As ground truth data is difficult to obtain, we generated scenes containing from 2 to 6 object scans from the testing dataset (100 scenes from each type) and labeled them with the known object category, as shown in Figure 5.12. This way we can quantitatively evaluate the effect of scene complexity on the results, as shown in Figure 5.13. Since in this test unknown objects are not considered, the features incorporating color outperform GRSD-.

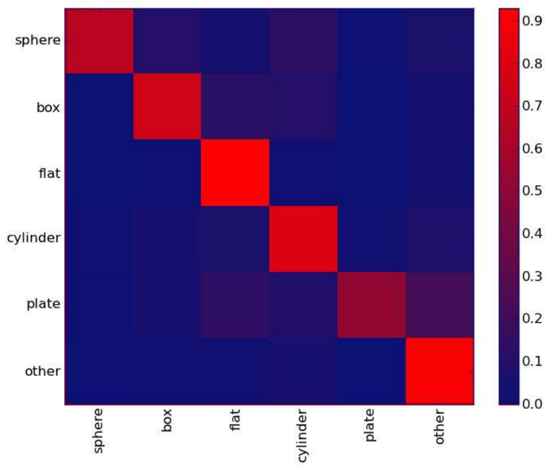
The generated scenes do not contain occlusions, but the results are indicative about the performance drop when more and more false groupings are considered by our method. Considering more than 6 touching objects should affect the results less and



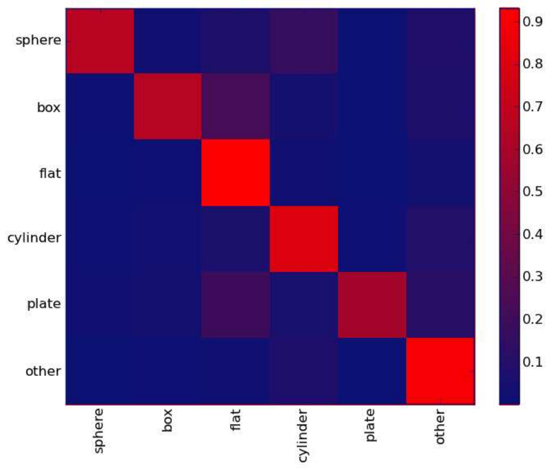
(a) Success Rates



(b) GRSD-



(c) C^3-HLAC



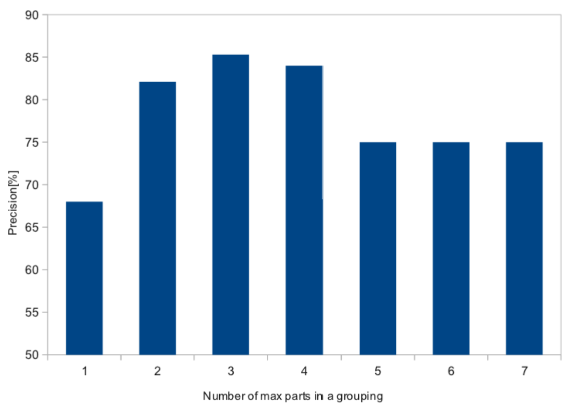
(d) VOSCH

Figure 5.13: *Per-segment results on the 600 generated scenes from scans in the testing split. Please note the different scale of the confusion matrices used for better visualization.*

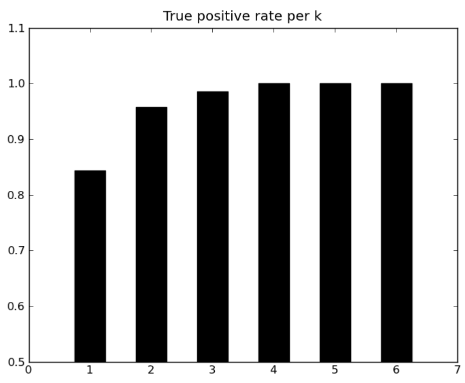
less, as the number of parts that are grouped is limited. Best results on the real scenes were obtained for 3-4 parts being considered, as detailed in the next section.

5.3 Accumulating Cues and Exploiting the Robot’s Embodiment

One of the important contributions of our work is the grouping of neighboring parts and taking into account the results of these groups for classifying a part. When grouping parts, there is an upper threshold on the maximum number of them that can form a grouping. During training, this threshold was set to 8. In the case of tabletop scenes we experimented with this threshold and found that the optimal maximum number of parts that can form a grouping is actually less then the one used for single objects. Results of this evaluation are shown in Figure 5.14 (a). It can be observed that grouping the segments greatly improves the classification process up to a given number of parts. However, if we choose the threshold to be too high in a cluttered scene, we risk grouping parts together that do not belong to the same object.



(a) effect of part sub-graph size



(b) percentage of correct top k votes

Figure 5.14: *Results for different maximum parts in a grouping, and different number of top votes considered averaged for the scenes 1 and 2.*

As a final experiment, we repeated the experiment from Figure 5.9 (b) for the combined (per-point) results of the first two scenes as well, see Figure 5.14 (b). Again, we found that the top votes are correct in the majority of cases, with the success rate increasing by nearly 12% to 95.8% with the first two votes being considered. These results suggest that the highest votes are close to each other, and additional information is needed for choosing the correct one. As we saw in the previous chapter, this extra information could come from a second scan of the scene from a new viewpoint, as shown in Figure 5.15.

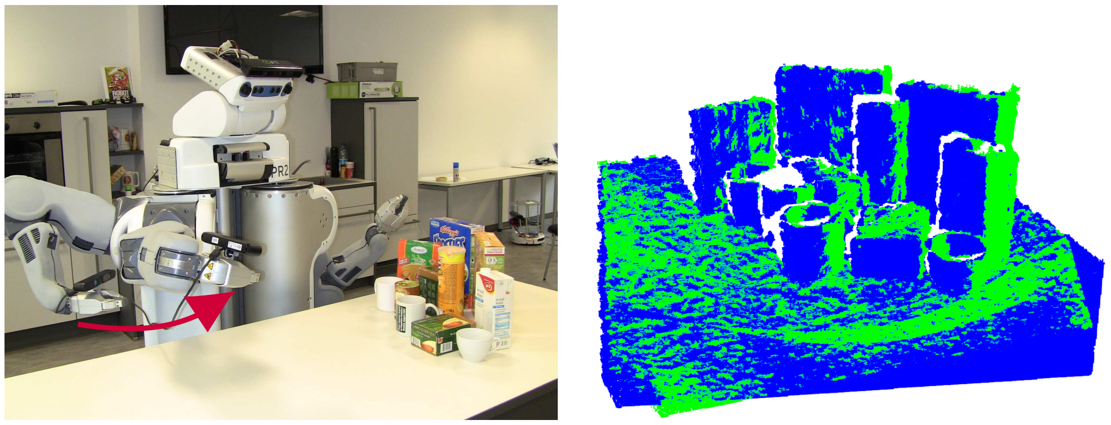


Figure 5.15: *As the camera is moved (left), multiple frames can be captured that cover different parts of the objects in the scene (right). Thanks to the calibration of the kinematic chain and localization, registered frames can be captured from a variety of viewpoints.*

The advantage of incorporating multiple views is evaluated on six views of a test scene. We used GRSD- as a feature for speed and the “Small+VDS” dataset combination for training to obtain a good balance between general objects and environment specific ones. As the robot is calibrated, all the scans can be placed into the same coordinate system, with only small misalignments (that could be fixed by an Iterative Closest Point algorithm). Then a 5 *mm* voxel grid was used to assign points from different frames to each other. The votes were accumulated for each voxel, and a per-point success rate is calculated both for the individual frames, their average and for the merged RGBD point cloud, presented in Figure 5.16.

The robot’s end-effector was moved such that to point the camera towards the scene while moving along a circle that respects the minimum range requirement from the scene. Still, some of the scenes were not captured fully, or not from an optimal angle, so large variations in accuracy can be observed. By incorporating multiple views however, the overall success rate could be improved by nearly 5%.

5.4 Model Reconstruction for Grasping

After a geometric labeling is obtained of the point cloud, the next logical step is to use this information to guide the segmentation and reconstruction of 3D models of the

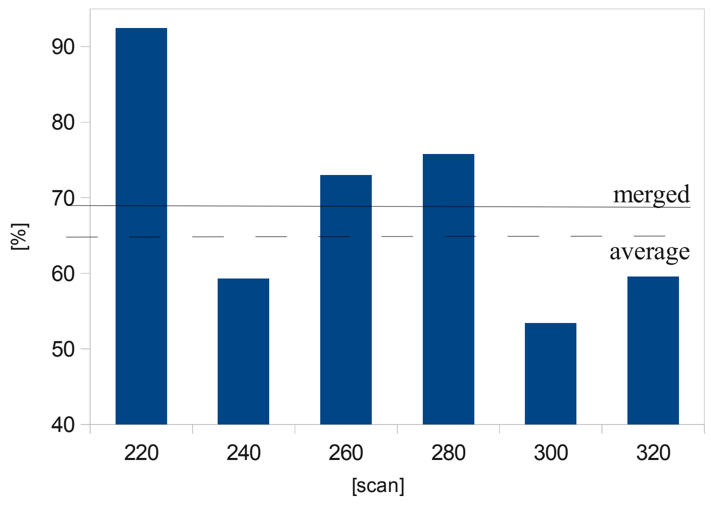


Figure 5.16: *Results for a cluttered scene with 7 frames from multiple viewpoints (denoted by angles around the table's normal).*

objects. In order for these to be useful for manipulation tasks, they should include also the occluded sides of objects, s.t. the grasp planning algorithm can reason about valid finger positioning, minimizing the risk of collisions (Marton et al., 2010a).

5.4.1 Shape-fitting and Verification

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 as well. However, the categorization method was already successfully employed to pre-segment scenes and to signal the presence of remaining under-segmented parts to an interactive segmentation system (Hausman et al., 2013).

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 5.17, the distribution of the number of different object parts can be modeled as a Poisson distribution, with an average error of 1.7% (and at most roughly 9%). Since some geometrical classes have similar looking distributions, we merge them together by considering everything spherical and cylindrical being *round* and disks/plates, flat rectangles and boxes as *flat* objects. With the category *other* we thus get three object types, whereas most household objects fall into the first two types, as summarized in Table 5.10.

	OPD	KIT	SDO	WG	TOTAL
Round	151	43	12	107	313
Flat	69	52	12	6	139
Other	99	15	11	19	144

Table 5.10: *Public object model databases and contained object types, from the following sources: TUM Organizational Principles Database (OPD), KIT Object Models Web Database (KIT), TUM Semantic Database of 3D Objects (SDO), and Household Objects DB from Willow Garage (WG), Based on (Marton et al., 2011) by merging the geometric categories appropriately.*

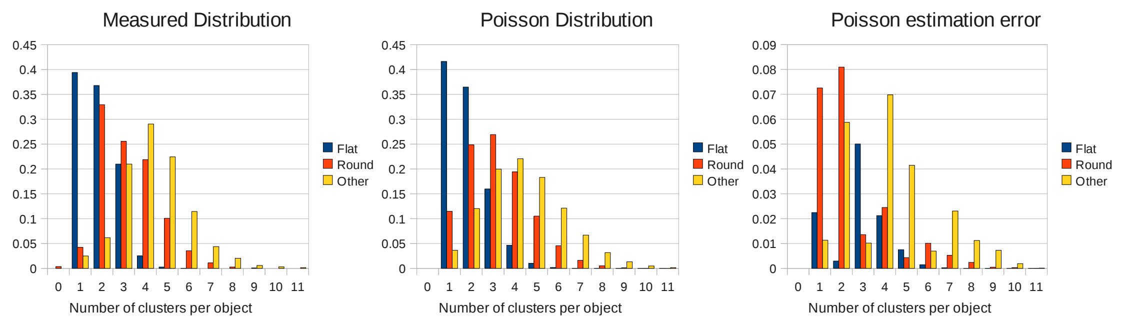


Figure 5.17: *Distribution of number of parts per object (for different object types) in the training dataset and their approximation with a Poisson distribution.*

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.

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

This probabilistic approximation can be used to judge if a group of parts of the same geometric category forms a single object or not.

However, before an interactive segmentation is performed, as in the case of doors and drawers (Blodow et al., 2011a) and household objects (Bersch et al., 2012; Hausman et al., 2013), fitting of object models should be attempted in order to disambiguate, verify and correct the segmentation. We described a method for voting for object centroids followed by a model fitting step in (Mozos et al., 2011), but that would require CAD models of the objects known a priori and the consideration of 6DOF poses as well.

Therefore, here we focus on obtaining a grouping of parts into objects by geometric fitting and grouping. We extend the model reconstruction method from (Goron et al., 2012) to use the obtained probability distributions over the geometric labels as priors when selecting the type of object. The work deals only with upright rectangular and cylindrical shapes for now, but this covers a large percentage of objects in our household table-top settings, as discussed earlier and in (Marton et al., 2011).

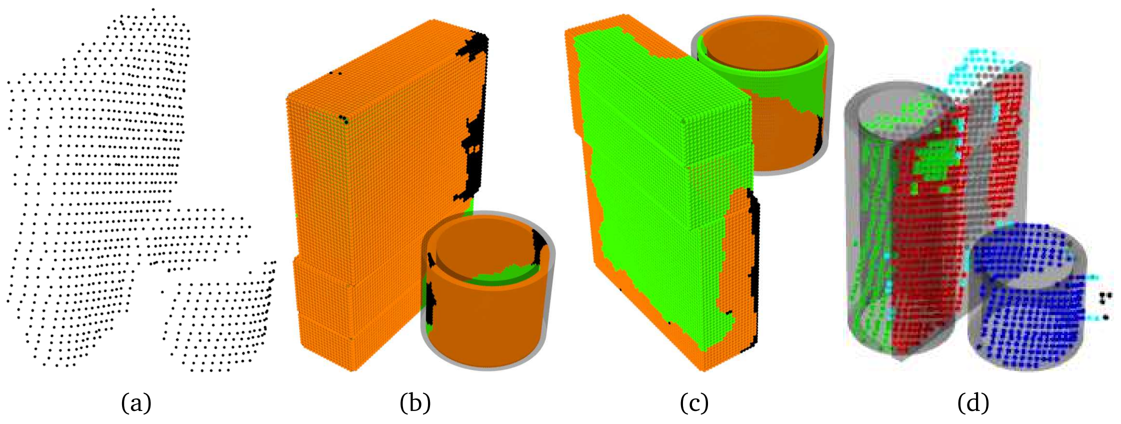


Figure 5.18: *Reconstruction results using the method from (Goron et al., 2012) for the two touching objects in (a). Regions of the final reconstructed surface are labeled: marking verified (orange), invalid (black) and occluded parts (green), as shown in (b) and in the back-side view (c). Even with these verifications, in some cases objects are incorrectly reconstructed, without violating the visibility checks: the cereal box is approximated by a flat box and a cylinder in (d).*

The method works by selecting the geometric model that best explains the data in a combination of RANSAC and Hough voting. There are several checks to filter out bad models, but as seen in Figure 5.18 these not always suffice. We observed these bad fits in out table-top scans as well, as shown in the top row of Figure 5.19. Thus we used the categorization results to better guide the model selection, by incorporating the shape probabilities as weights into the sampling (which used to be random) and the scoring as well. To match the methods capabilities, the categories were collapsed into rectangular and round shapes. As shown in the bottom row of Figure 5.19, this is able to correct some of the inaccuracies of the fitting.

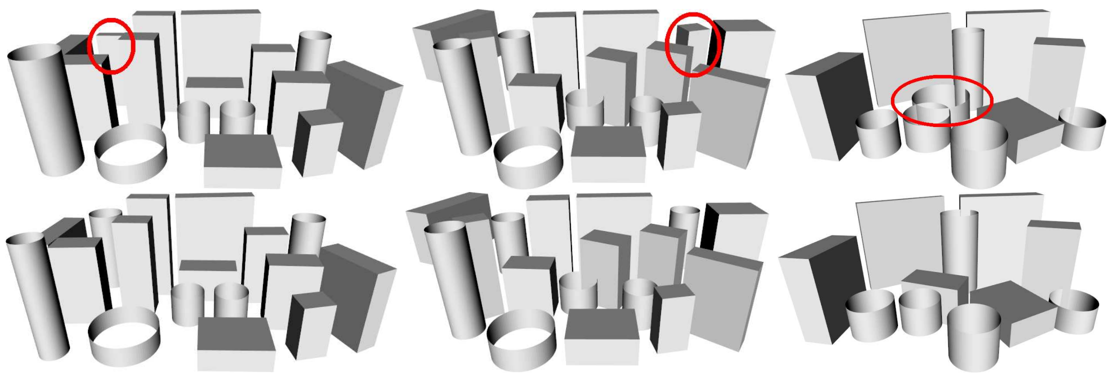


Figure 5.19: *Geometric modeling of the three test scenes. Top: original method (Goron et al., 2012) with fitting errors marked with red. Bottom: correct reconstruction using the priors from the categorization.*

By reconstructing completed 3D models that include the back side as well, the robot is equipped with the necessary information to grasp the objects (reactively, and avoiding the regions labeled as invalid) instead of pushing them for individuation. This way the segmentation and the models could be checked and improved. That constitutes a completely new research direction, however, so we will focus here on the passive observation of the scene.

In the following subsection we will evaluate and give more detail on the reconstruction methods that lay the foundation for the line of research (Marton et al., 2009a; Goron et al., 2010) that lead to (Goron et al., 2012) and the integrated reconstruction approach used here.

5.4.2 Creation of Complete 3D Models from a Single View

In this subsection, we will lay out the 3D reconstruction methods from (Marton et al., 2010a) we integrated in order to create complete object model hypotheses that can be used in grasping scenarios. The focus here does not lie on millimeter-accurate modeling of clusters, which is realistically not possible in a lot of cases due to sensor noise and large variability in the objects we encounter. Instead, we strive to improve the models generated from point cloud data in two aspects. First, we want the resulting reconstruction to be smooth, especially when compared to a simple meshing approach, where sensor noise causes problems in the grasp planner, e.g. when computing contact points and forces. Second, we want to generate a working hypothesis concerning the backside of an object perceived only partially. Otherwise, grasp analysis might suggest a pinch grasp on the edge of the cluster, even though the object continues smoothly towards the unobserved back.

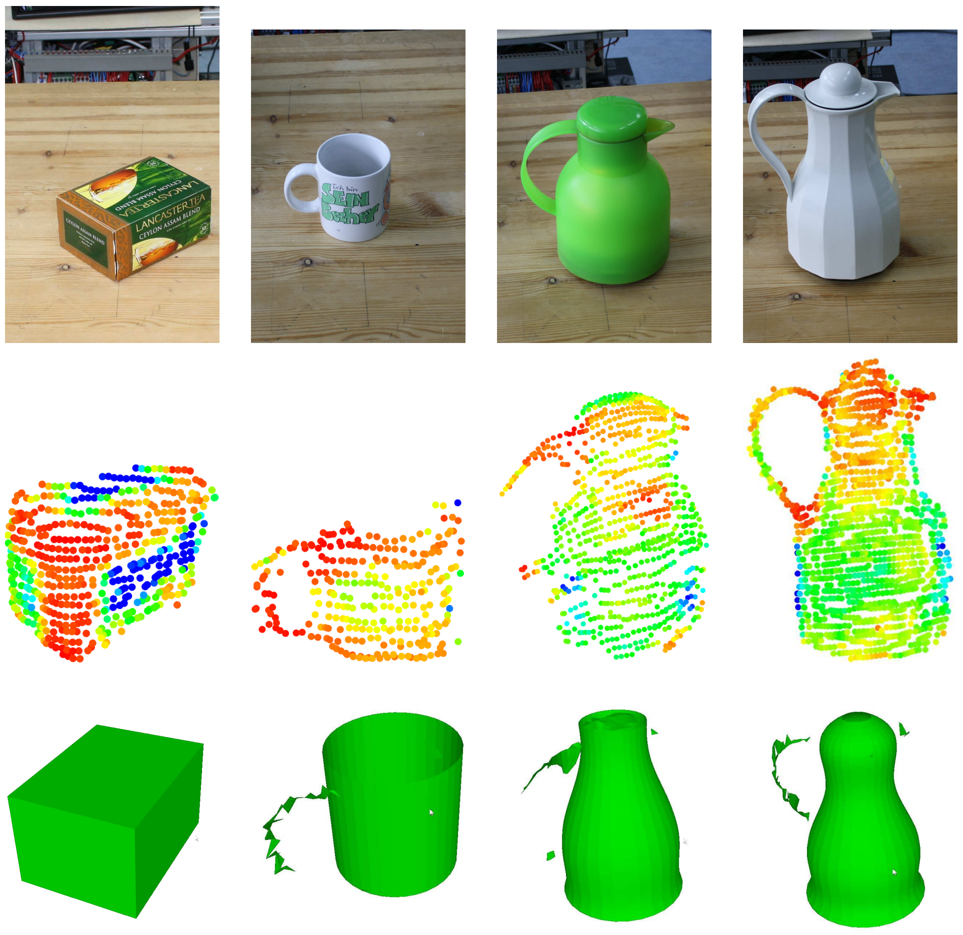


Figure 5.20: *Automatic reconstruction of objects of different types. Top row shows the estimated r_{min} values color-coded by the legend of Figure 2.23 (red* 0*, blue* 0.2 *m or higher).*

We have developed reconstruction algorithms for boxes, cylinders and rotational objects (Marton et al., 2010a), as shown in Figure 5.20. We chose these models because they are inherently symmetrical and occur frequently in a large number of objects (as shown in Table 5.10). In the case of cylinders and rotational objects, a rotation axis is determined, which allows for simple completion of the object model, and the hidden surfaces can be retrieved for boxes as well.

5.4.2.1 Box and Cylinder Fitting

The selection of the appropriate fitting model is based on our previous work on footprint analysis (Marton et al., 2009a), but in addition, we rely on the 3D normals to robustly detect a circular or a rectangular footprint, also during the 3D model fitting and validation step. This enables our method to make the best choice regarding what model it should choose.

If a sufficient number of points have a minimum radius greater than $0.1m$ they highly likely lie on planar surfaces and we set out to find the best fitting box to the cluster. Unlike in (Marton et al., 2009a), we fit a rectangular model directly to the points having normals perpendicular to the “up” axis (as we assume boxes to be statically stable, i.e. standing on one of their sides) and maximize the number of inliers in a RANSAC loop. This direct approach estimates the box orientation and outperforms the line detection and merging and/or box detection based on PCA. Since we assume a correct segmentation and noise is removed in pre-processing, the box dimensions are computed from the oriented bounding box, and a final inlier count is performed.

For cylinders, we use a RANSAC approach which is based on the observation that on a cylinder surface, all normals are 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.

The quality of box and cylinder fittings for two objects is presented in Figure 5.21. Further tests on a total of 267 measurements of 5 objects show an average fitting error in 5.31 mm with a standard deviation (STD) of 2.69 mm for the extensions of boxes, and slightly less for cylinder radii, an average of 2.85 mm and STD of 1.23 mm. Both are well below the original sensor noise of roughly 1 cm. The average error for the axis orientation for cylinders was 1.11 degrees with a STD of 0.34 degrees. By enforcing an upright position for the axes results are more robust, but it is not mandatory.

Compared to the original approach presented in (Marton et al., 2009a) we can see a clear increase in accuracy. On the same scans, the obtained average errors for box extents was 10.6 mm with STD 6.05 mm. Results did not improve (nor worsen) for cylinders as the methods have the same basis, but instead we are now reconstructing them in arbitrary poses, while the original approach assumed all objects to be upright (and would consider lying cylinders to be boxes).

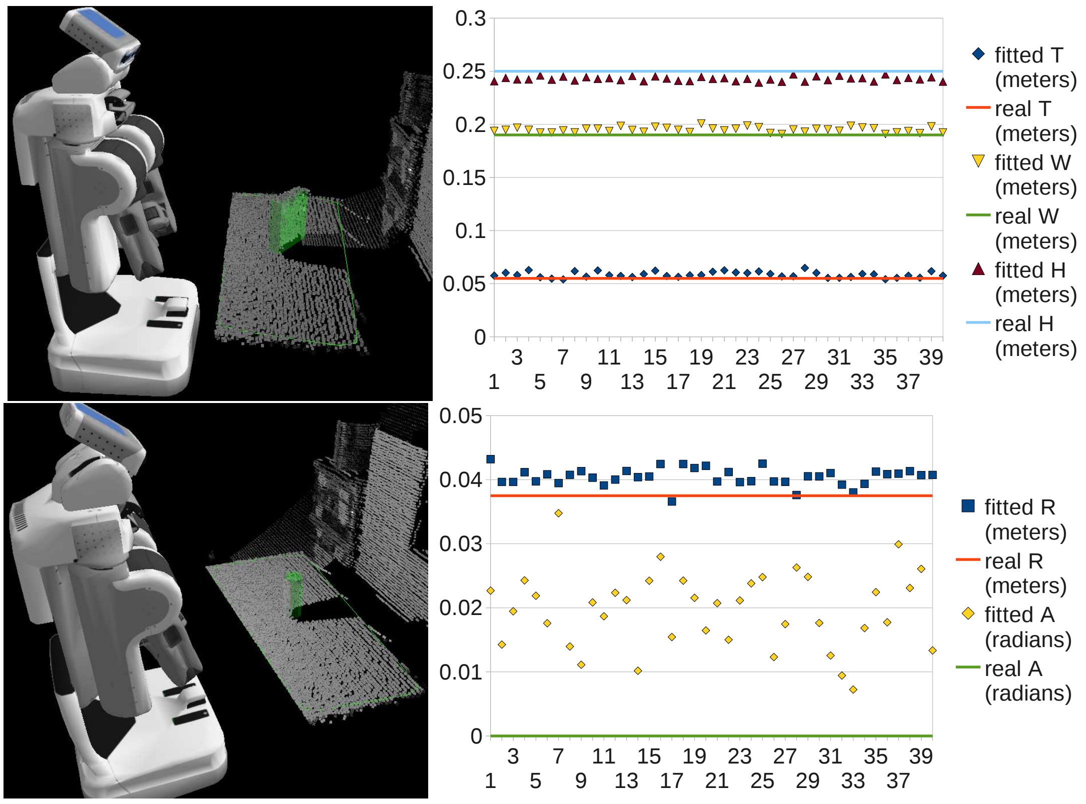


Figure 5.21: *Left: visualization of the setup used for measuring fitting errors. The model and the table are marked with green. Right: errors in estimated box width (W), height (H), thickness (T) – and in estimated cylinder radius (R) and angle to the vertical (A). Graph shows these errors for subsequent scanning and fitting (X axis shows the scan number).*

5.4.2.2 Estimation of Rotational Surfaces

To reconstruct surfaces of revolution, we employ a RANSAC-based two-step approach as described in (Blodow et al., 2009). In the first step, a rotation axis is estimated from sample points by minimizing a function over the line-to-line distances between the axis and the lines defined by each sample point and its corresponding normal. This is based on the observation that for a rotational object, a line constructed from a point and corresponding normal intersects the symmetry axis, similar to the cylinder case. The contour line is then estimated in the second step, which is explained and evaluated in greater detail in (Marton et al., 2011).

Although the rotational estimation is less accurate and computationally more intensive than fitting of simple models, it is an important addition in order to be able to deal with more types of objects. More accurate results can be obtained by increasing the expected probability of a successful fit for RANSAC (and thus the maximum number of iteration), but in order to obtain results under 2 seconds, axis estimation errors around 10 degrees were obtained.

In the case of box-like and cylindrical models, the fact that the dimensions are computed accurately from partial views already validates the use of this approach for grasp planning. As presented in subsection 6.1.2 having a hypothesized back side makes a significant difference when the goal is to pre-compute realistic stable grasps. Similarly, this is an important consideration for rotational objects as well, as considering only the scanned points does not give any estimate on how large the object is, and what the back side looks like.

5.4.2.3 Mesh Representation

If a fitted model does not have at least 75% of the points as inliers, we assume that it has a more complex shape, and a simple triangulation is performed as a fall back, as described in 2.3.2. Also in the other cases, the outliers of the geometric models are triangulated and added to the final model as a mesh, e.g. to model additional geometric features of an object, such as handles. Figure 5.22 shows such an example. As the majority of grasp planning methods use triangular meshes, the geometric shapes are decomposed in triangles as well, but we are exploring the usage of the shapes directly for more optimized grasping.

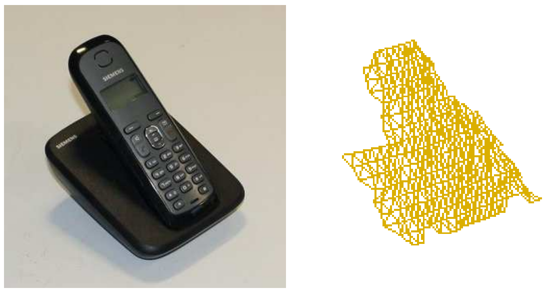


Figure 5.22: *Triangulation of an object where no good model could be fitted.*

5.5 Discussion

In this chapter we have shown the capability of our perception system for exploiting multiple frames and part-graph descriptors to deal with object categorization in clutter. The system was evaluated on a large RGBD dataset, and on Kinect scans of cluttered tabletop scenes, and showed promising results when compared to alternative approaches. The advantage of geometric features was shown for the cases when testing objects that are very different from the trained ones needed to be categorized. Thanks to the efficient hashing-based categorization, it is possible to obtain classification results fast for complete scenes. This allows the merging of results coming from multiple views of the same scene in order to improve detection. As argued by Fergus et al. (2003) as well, partial occlusion and shape variability is handled well by object part relations. As we can produce multiple segmentations by choosing different (random) seed points, different part decompositions can be used for training, which improved classification rates by 5% in our first experiments (Balint-Benczedi et al., 2012). Analogously, multiple segmentations of a frame, or segmentations of consecutive frames could be merged, using the presented voting method fusing different viewpoints.

The presented large scale evaluations validate the development of the presented multi-cue perception system, and the following chapter presents some additional application scenarios and the most important demonstrations that were performed using it.

Chapter 6

Applications and Discussions

Humans are really good at being able to take a bit of knowledge and use it to great advantage. [...] It's important not to wait until we understand everything, because that's going to be a long time away.

BERT VOGELSTEIN

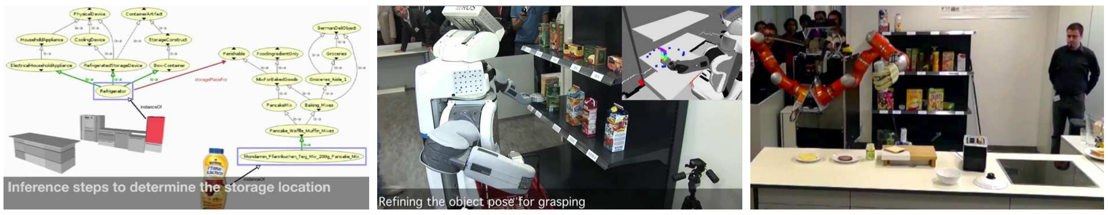
As in biological systems, robotic systems are composed of several layers of complexly interacting subsystems. Current robots are far away from the flexibility and robustness of living things, but that does not mean that there are no good solutions for certain subproblems already. In this chapter, I will present some of the higher level applications of the work described so far, to highlight the uses of a multi-cue perception system for robotic object manipulation tasks.

6.1 Demonstration Scenarios

The proposed system was validated in the context of the execution of a high-level task like preparing a meal and going shopping, as shown in Figure 6.1.

Demonstrations of the presented approach were made for example during the 2nd BRICS Research Camp "From 3D sensing to 3D models" (www.best-of-robotics.org/2nd_researchcamp/MainPage) and various workshops of the Cognition for Technical Systems (CoTeSys) cluster of excellence. Among others, these included the pancake[1] and shopping[2] (Pangercic et al., 2011b) demos, taking advantage of the built environ-

[1]www.youtube.com/watch?v=4usoE981e7I&feature=channel_video_title
[2]www.youtube.com/watch?v=x0Ybod_6ADA


Inference steps to determine the storage location
Refining the object pose for grasping

(a) Pancake making (b) Going shopping (c) Sandwich preparation

Figure 6.1: *High-level demonstration scenarios involving the developed multi-cue perception system, on public CoTeSys Fall Workshops involving the TUM-Rosie and PR2 robots.*

ment map, and employing the perception system for detecting the shopping basket and other objects that were manipulated.

During the last CoTeSys Fall Workshop sandwich preparation was demonstrated using the TUM-Rosie robot[3]. A region of interest was provided by the task executive using the known environment model (see Chapter 4), along with the list of possible objects to be detected. Then, the different detection, classification and model fitting methods decide for each request to activate or not. They do this based on the objects to be detected and if they have models for those, as they are specialized for different detection tasks (see Section 3.1). Different 2D and 3D methods are chained in order to produce the final result, i.e. list of object locations and poses or reconstructed models. The task executive then interprets the results, decides on the next action to be taken (which could be repeating a failed procedure) and triggers a new task if necessary.

Specifically, this meant locating the cutting board, toaster plates, etc. using the appropriate methods, given the location of the table from the semantic environment map. Having obtained these regions of interests (ROIs), the toast bread and other ingredients were located, either geometrically or image-based. After each manipulation tasks as well, detection tasks are assigned to the multi-cue perception system in order to verify the success of the results (as in the case of the salami for example).

Thus, the presented system was instrumental in achieving high-level functionality through providing accurate object reconstruction, categorization and classification routines to the planning system. Additional use cases are described below, despite these not being the main focus of the thesis.

[3]www.youtube.com/watch?v=DTaeWITW1kI

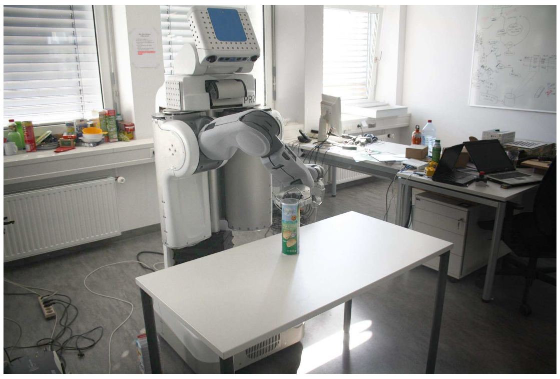


Figure 6.2: *Grasping of the localized object by the PR2 robot. The fitted geometric model was triangulated and fed into the grasp planning software to obtain grasping points.*

6.1.1 Interactive Segmentation of Textureless Objects

Similarly to view planning, interacting with the scene is an important strategy for understanding a difficult scene or to gather more information about it, as we discussed in Section 4.2.1. Inspired by the correction of door segmentation through opening it and observing the difference (Blodow et al., 2011a), we developed a system for introducing a change in a cluttered setting in order to segment the objects.

The approach, that was a finalist at for the Best Service Robotics Paper Award at ICRA 2013 (Hausman et al., 2013), used the presented system for analyzing the scene and to create the final object segmentation that considers both the part structure and the additional movement trajectory information. The fact that different parts of the (rigid) objects move together in a geometrically consistent way was included in the part grouping strategy, enabling the robot to deal with scenes that are difficult to segment on visual information alone.

6.1.2 Grasping of Objects using Reconstructed Object Models

The reconstructed geometric models are of utmost importance for accurate grasping. In our grasping scenarios the triangulated meshes are compared with stored models which have already grasp points computed, and if no good enough match was found they are uploaded into the GraspIt simulator (Goldfeder et al., 2009) where they get annotated with large sets of stable, pre-computed grasp points. Computed grasps can then be performed by the robot, as shown in Figure 6.2

Grasps for novel objects (e.g. Figure 5.22) are computed for the specific robot (PR2 and TUM-Rosie) at run-time from 3D sensor data, using heuristics based on both the overall shape of the object and its local features. We kindly refer the reader to the papers by (Hsiao et al., 2010) and (Marton et al., 2010a) to obtain more information for PR2 and TUM-Rosie calculation of grasps. We make use of the grasping pipeline developed for the PR2 robot, adapted to our application, which is thoroughly described in (Ciocarlie et al., 2010).

Each of the resulting grasps is then tested for feasibility in the current environment; this includes collision checks for both the gripper and the arm against potential obstacles, as well as generation of a collision-free arm motion plan for placing the gripper in the desired pose.

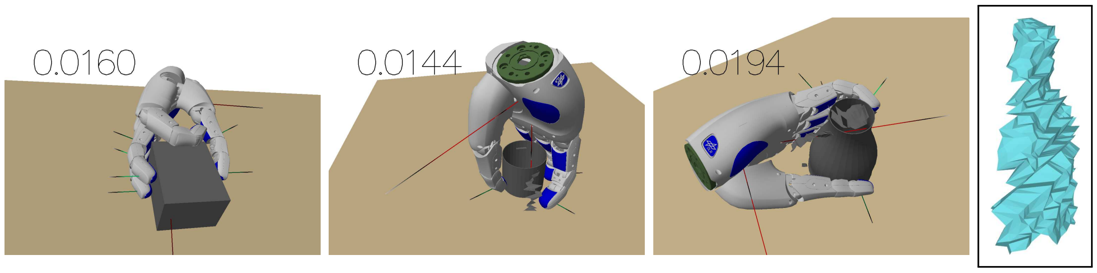


Figure 6.3: *Grasp planning application for TUM-Rosie. Note that we used incomplete scans of the objects, like the one depicted in the last image (side view of triangulated mesh) and were still able to generate good grasps for the back sides.*

The major benefit of the reconstruction routine is that grasps are calculated by also considering the backsides of the objects as depicted in Figure 6.3. From the contacts, the grasp quality is statically estimated using the algorithm presented in (Ferrari and Canny, 1992).

6.1.3 CAD Model-based Object Detection

In case 3D data is unavailable or of too low quality to perform pose estimation as in (Aldoma et al., 2012), it is possible to perform 6 DOF position retrieval of objects on camera images using state-of-the-art 3D shape model matching technique that simulates the 2D appearance of the objects in a shape model generation phase (Ulrich et al., 2009a). The routine gets object candidates (clusters of points) projected onto the corresponding 2D image from point cloud data and then uses a priori built CAD model of objects to perform the actual detection. The strength of this approach is in that the CAD models do not need to be modeled manually but can rather be generated by the robot itself in the desired complexity (see Figure 6.4 for the matching results and Section 5.4.2 for the generation of CAD models).

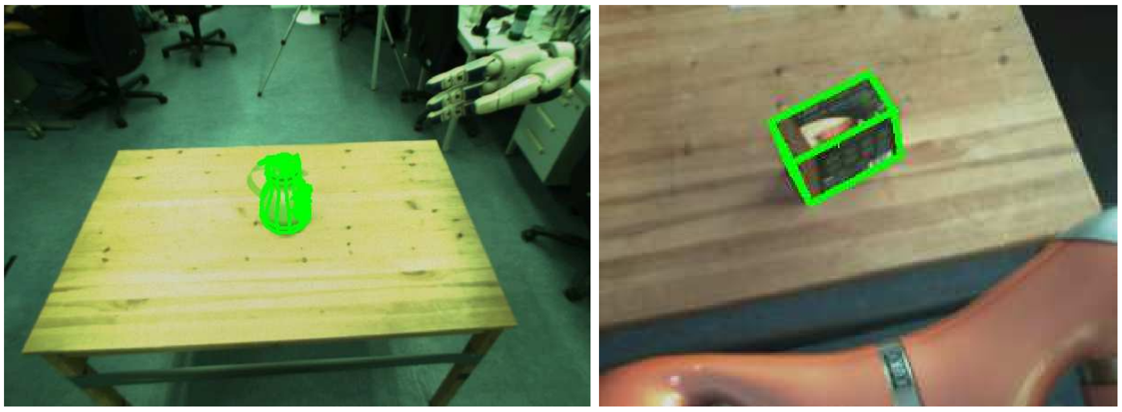


Figure 6.4: *TUM-Rosie locating the resultant geometric model in an image.*

These meshes can be used by other systems as well, even if the sensors and their setup is different. If, however, the model is built by the agent performing the CAD model-based detection, the known pose of the model can be used to avoid unnecessary matching of very different poses in the image.

6.1.4 Lifelong Dynamic World State Mapping

The multi-modal nature of our perception system lends itself to applications for entity resolution (Blodow et al., 2010), where we proposed to use Markov Logic Networks to probabilistically track trajectories of objects over time using reconstructions and refinement algorithms as described in in this thesis. The approach computes similarity measures between different object cluster observations from compatible object descriptions and uses the full power of first-order logic with the probabilistic semantics of graphical models in order to deal with difficult conditions such as partial observability of the robot's environment, incomplete object descriptions, and a dynamic environment in which we can not observe the human and his actions directly.

6.1.5 Platform independent Perception System

Our perception system has been validated on 2 state-of-the-art service robots, the TUM-Rosie and the PR2 thanks to the general infrastructure provided by ROS. Since the only prerequisite is to have the kinematic chains between the 2D cameras and the 3D tilting laser sensor calibrated with the precision under 5 mm (which still assures correct region-of-interest extraction for the purpose of tabletop manipulation), our system can thus be ported to any arbitrary robot platform that meets the above requirement. As most of the testing was done on the PR2 robot, Figures 6.3 and 6.4 verify our claims for the TUM-Rosie robot by showing the execution of the shape-based matching algorithm and grasp planning respectively.

6.2 Conclusions

Scientists and others tend to be quite fond of neat, clear-cut patterns. Nature is not. [...] Why, then, should the outcome correspond to our logical ideas about how to build a living system that succeeds?

DENIS NOBLE (THE MUSIC OF LIFE)

So that the possible may arise, the impossible must always be attempted.

HERMANN HESSE

The presented work was put to use in real-world scenarios, demonstrating its uses and potential to scale, and be integrated into a complete robotic system. While connecting the perception system to planning and knowledge representation modules is clearly mutually beneficial, there are inherent limitations in perception that remain difficult to solve. This is due to difficult, ambiguous situations, like the one exemplified in Figure 6.5, and limited understanding of how humans deal with such cases. Nonetheless, the system is capable to deliver useful results, and is a step in the right direction, as suggested by theories of human cognition.

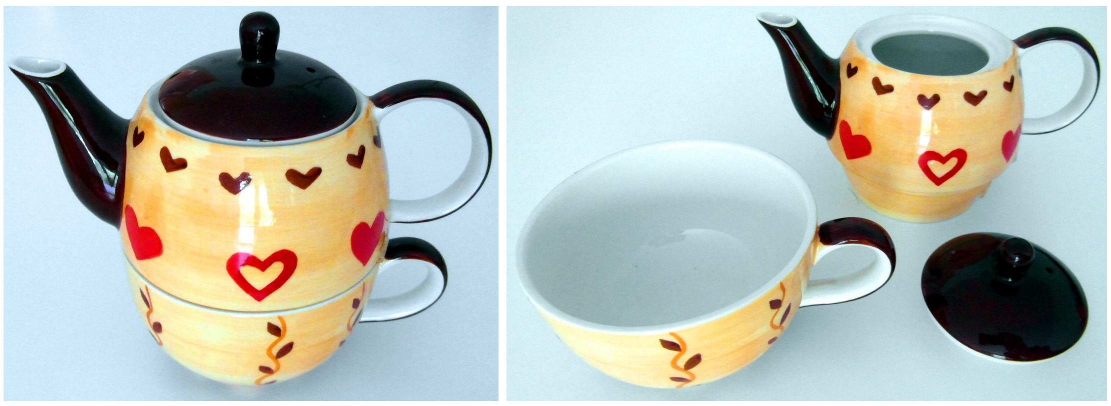


Figure 6.5: *One object or two: difficulties in definition are reflected by difficulties in object segmentation, recognition and reconstruction. While the presented combination of a tea cup and tea pot respects many principles defining a single object, it is not physically connected and is separated during manipulation. The color-, texture- and 3D-based boundaries in some cases do and in others don't present real boundaries. My wife and I also tend to call and treat it as one object or two, depending on the situation.*

While there is no generally accepted theory explaining human brain function, there are large scale projects aiming to elucidate it in the near future (Alivisatos et al., 2012). There are several theories on how the brain perceives and reacts to the world, and these provide promising research directions also for robotics. In line with the integration approach presented here, the Bayesian brain principle (Doya et al., 2007) assumes a continuous update of hypotheses about the world, and correcting them through sensory information.

In the machine learning field, deep learning using artificial neural networks is becoming an increasing popular concept also in computer vision (Le et al., 2012). It is being deployed already by companies like Google, Microsoft and Apple's Siri virtual personal assistant for speech recognition tasks, but also has uses in address identification from images and drug design (Markoff, 2012). Friston and Stephan (2007) details how ideas related to artificial neural networks might explain the real network of the brain's neurons and its capability for inference, adaptation and plasticity, using Bayesian filtering in a neurobiologically plausible way.

This framework is closely related to the common coding theory (Sperry, 1952), arguing that perception and action are treated together in the brain, one generating the other such that to achieve a desired configuration. This calls for a closer integration for perception and action than the classic perception-cognition-action loops that is also followed in this thesis. This would mean extending the described multi-cue perception system with a perception generating action module. First attempts in this direction have been the next best view planning and *interactive segmentation* approaches described earlier, and this research direction is starting to gain momentum both in the robotic vision and action planning domains (early examples of this being systems inspired by mirror neurons).

All in all, the thesis presents theoretical and practical solutions for the development of cognitive robotic systems, that are supported by theories of the brain and perception, and advance the field by creating reference implementations for them. Two main ideas span multiple chapters and methods and serve as a condensed lesson from the contributions of this work. These general principles are based on psychological findings and practical experience, which were applied and evaluated throughout the thesis, confirming their validity:

1. the fusion of segmentation (bottom up) and model fitting (top down) approaches for pose estimation and reconstruction;

2. the merging of cues/information for object recognition, on the levels of sensor data, feature, object part, and classification.

The following subsections summarize in the light of these principles the specific problems that were addressed (listing our publications that are most relevant for the topic), and present the open source dissemination of the results.

6.2.1 Detection of Objects in Complex Scenes

Shape-fitting and Verification. Categorization and model reconstruction of previously unknown objects, based on geometric properties, as published in (Marton et al., 2009a; Rusu et al., 2009a; Goron et al., 2010; Blodow et al., 2009; Marton et al., 2010a, 2011; Goron et al., 2012).

Geometric Over-segmentation. Detection/segmentation of furniture pieces and container doors in realistic cluttered scenes of complete rooms, and similarly detected objects based on over-segmentation and sub-graph classification (Marton et al., 2009c; Blodow et al., 2011a; Mozos et al., 2011; Marton et al., 2012a; Usenko et al., 2012; Marton et al., 2013, 2014).

Detection Based on Environment Models. The use of the acquired environment model by the robot to improve and guide object detection. Methods like incremental triangulation, moving least squares, and segmentation methods are presented in (Rusu et al., 2007, 2008c, 2009d; Marton et al., 2009b), and their use in object detection (e.g. detecting fixtures, moving objects). These models can also be used for improved human-robot-interaction (Blodow et al., 2011b).

6.2.2 Accumulating Cues from Multiple Sources for Object Classification

Informative Local Features for 3D Data. Efficient computation of informative features from 3D data used for local shape classification, registration and model fitting (Rusu et al., 2008a,b; Marton et al., 2010a, 2011).

Part-based Multi-modal Classification. The advantages of the deep combination of 3D and color/texture features for object detection clutter shown based on TOF, laser, mono and stereo cameras, and on the Kinect sensor (Marton et al., 2009c, 2010b; Kanezaki et al., 2011c) and also for improving the detection of furniture pieces (Blodow et al., 2011a).

Ensemble Learning for Multi-cue Classification. Combinations of classifiers that are trained on different features (potentially coming from different sensors) are shown to be able to outperform individual classifiers and classifiers trained on a simple concatenation of features (Marton et al., 2012c,b). This idea can be applied for merging results obtained from different viewpoints or segmentation results (Mozos et al., 2011; Balint-Benczedi et al., 2012; Marton et al., 2013, 2014). Similarly, fusing multi-modal correspondences also improves local and global registration (Stanimirovic et al., 2013).

6.2.3 Integration of Algorithms in Large Open Source Projects

My work formed a part of the Cognitive Perception (CoP) project in the CoTeSys cluster of excellence, that resulted in the creation of the Point Cloud Library (PCL) and its predecessors (Rusu, 2009), which constitutes the 3D perception backbone of the Robot Operating System (ROS). I integrated the methods which are mostly related to 3D data processing (like surface resampling, triangulation, segmentation, 3D feature descriptors, etc.) into PCL[4], while methods that integrate robotics related information (e.g. trajectories, or are meant to run on a distributed system) got integrated into ROS packages for simple reusability[5].

The PCL project started at the IAS group by Radu B. Rusu, Nico Blodow and myself got extremely popular, attracting many contributors and user, becoming a self-supporting community whose activities are handled by Open Perception[6]. It is used and supported by several companies and became the 3D equivalent of OpenCV. On the 22nd of November 2011, PCL was awarded the Grand Prize of the Open Source Software World Challenge 2011 by the Ministry of Knowledge and Economy of Korea[7].

[4]http://pointclouds.com
[5]http://code.in.tum.de
[6]http://openperception.org
[7]http://ossaward.org/

List of Prior Publications

Part of the work presented here was described earlier in various peer-reviewed publications I worked on during my stay at the Intelligent Autonomous Systems group. Below the complete list of my publications is given, which were cited in the appropriate places of the thesis. In cases where multiple authors contributed an equal amount, they were marked with an asterisk (*).

Journal Articles

Zoltan Csaba Marton, Ferenc Balint-Benczedi, Oscar Martinez Mozos, Nico Blodow, Asako Kanezaki, Dejan Pangercic, Lucian Cosmin Goron and Michael Beetz. Part-Based Geometric Categorization and Object Reconstruction in Cluttered Table-Top Scenes. *Journal of Intelligent & Robotic Systems*, 76(1):35–56, January 14, 2014.

Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Tutorial: Point Cloud Library – Three-Dimensional Object Recognition and 6 DoF Pose Estimation. *Robotics & Automation Magazine*, 19(3):80–91, September 11, 2012.

Zoltan Csaba Marton, Florian Seidel, Ferenc Balint-Benczedi, and Michael Beetz. Ensembles of Strong Learners for Multi-cue Classification. *Pattern Recognition Letters (Special Issue on Scene Understanding and Behavior Analysis)*, 34(7):754–761, July 27, 2012.

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, June 16, 2011.

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 (Special Issue on Semantic Perception for Robots in Indoor Environments, Part II), 30(11):1378–1402, August 1, 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, November 30, 2008.

Conference Papers

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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2013. **Best Service Robotics Paper Award Finalist**.

Zoltan-Csaba Marton, Florian Seidel, and Michael Beetz. Towards Modular Spatio-temporal Perception for Task-adapting Robots. In *Postgraduate Conference on Robotics and Development of Cognition (RobotDoC-PhD), a satellite event of the 22nd International Conference on Artificial Neural Networks (ICANN)*, 2012.

Lucian Cosmin Goron, Zoltan Csaba Marton, Gheorghe Lazea, and Michael Beetz. Segmenting Cylindrical and Box-like Objects in Cluttered 3D Scenes. In *7th German Conference on Robotics (ROBOTIK)*, 2012.

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)*, Abbey Kloster Seeon, Germany, 2012.

Ferenc Balint-Benczedi, Zoltan-Csaba Marton, and Michael Beetz. Efficient part-graph hashes for object categorization. In *5th International Conference on Cognitive Systems (CogSys)*, 2012.

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 *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September 25-30, 2011.

Zoltan-Csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3D Modelling of Novel Objects from a Single View. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 18-22, 2010.

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 2010 IEEE-RAS International Conference on Humanoid Robots*, Nashville, TN, USA, December 6-8, 2010.

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, Nashville, TN, USA, December 6-8, 2010.

Lucian Cosmin Goron, Zoltan Csaba Marton, Gheorghe Lazea, and Michael Beetz. Automatic Layered 3D Reconstruction of Simplified Object Models for Grasping. In *Joint 41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*, 2010.

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)*, St. Louis, MO, USA, October 11-15, 2009.

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 11-15, 2009.

Zoltan Csaba Marton, Radu Bogdan Rusu, Dominik Jain, Ulrich Klank, and Michael Beetz. Probabilistic Categorization of Kitchen Objects in Table Settings with a Composite Sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4777–4784, St. Louis, MO, USA, October 11-15, 2009.

Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

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)*, Paris, France, December 7-10, 2009.

Radu Bogdan Rusu, Jan Bandouch, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Action Recognition in Intelligent Environments using Point Cloud Features Extracted from Silhouette Sequences. In *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, 2008.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Emanuel Dolha, and Michael Beetz. Functional Object Mapping of Kitchen Environments. In *Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, September 22-26, 2008.

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning Point Cloud Views using Persistent Feature Histograms. In *Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, September 22-26, 2008.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning Informative Point Classes for the Acquisition of Object Model Maps. In *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Hanoi, Vietnam, December 17-20, 2008.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent Point Feature Histograms for 3D Point Clouds. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10)*, Baden-Baden, Germany, 2008.

Radu Bogdan Rusu, Nico Blodow, Zoltan-Csaba Marton, Alina Soos, and Michael Beetz. Towards 3d object maps for autonomous household robots. In *Proceedings of the 20th IEEE International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

Workshop and Symposium Papers

Zoltan-Csaba Marton, Ferenc Balint-Benczedi, Oscar Martinez Mozos, Dejan Pangercic, and Michael Beetz. Cumulative Object Categorization in Clutter. In *Robotics: Science and Systems Conference (RSS), 2nd Workshop on Robotics in Clutter*, 2013.

Darko Stanimirovic, Zoltan-Csaba Marton, and Michael Suppa. On Texture and Geometric Feature Based RGB-D Registration. In *3rd Semantic Mapping, Perception and Exploration Workshop, in conjunction with ICRA*, 2013.

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 *Robotics: Science and Systems Conference (RSS), Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012.

Karol Hausman, Christian Bersch, Dejan Pangercic, Sarah Osentoski, Zoltan-Csaba Marton, and Michael Beetz. Segmentation of cluttered scenes through interactive perception. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Semantic Perception and Mapping for Knowledge-enabled Service Robotics*, St. Paul, MN, USA, May 14-18, 2012.

Vladyslav Usenko, Florian Seidel, Zoltan-Csaba Marton, and Dejan Pangercic Michael Beetz. Furniture Classification using WWW CAD Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception (ASP'12)*, 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*, Västerås, Sweden, April 8, 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*, San Francisco, CA, USA, September, 25-30, 2011.

Nico Blodow*, Zoltan-Csaba Marton*, Dejan Pangercic*, Thomas Rühr*, Moritz Tenorth*, and Michael Beetz. Inferring generalized pick-and-place tasks from pointing gestures. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Semantic Perception, Mapping and Exploration*, May, 9-13, 2011.

Zoltan-Csaba Marton, Nico Blodow, and Michael Beetz. Advantages of spatial-temporal object maps for service robotics. In *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, Half-Moon Bay, CA, USA, October 2-4, 2011.

Nico Blodow, Zoltan-Csaba Marton, Dejan Pangercic, and Michael Beetz. Making Sense of 3D Data. In *Robotics: Science and Systems (RSS), Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

Zoltan Csaba Marton, Lucian Cosmin Goron, Radu Bogdan Rusu, and Michael Beetz. Reconstruction and Verification of 3D Object Models for Grasping. In *Proceedings of the 14th International Symposium on Robotics Research (ISRR)*, Lucerne, Switzerland, August 31 - September 3, 2009.

Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, Moritz Tenorth, Radu Bogdan Rusu, and Michael Beetz. Autonomous Mapping of Kitchen Environments and Applications. In *Proceedings of the 1st International Workshop on Cognition for Technical Systems (CoTeSys)*, Munich, Germany, 6-8 October, 2008.

Other References

Dejan Pangercic, Koppany Mathe, Zoltan-Csaba Marton, Lucian Cosmin Goron, Monica-Simona Opris, Martin Schuster, Moritz Tenorth, Dominik Jain, Thomas Rühr, and Michael Beetz. A Robot that Shops for and Stores Groceries. *AAAI Video Competition (AIVC 2011)*, 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*, St. Louis, MO, USA, October 11-15 2009. Invited paper.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Interpretation of Urban Scenes based on Geometric Features. In *Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop on 3D Mapping, Nice, France, September 26*, 2008. Invited paper.

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.

Bibliography

Aitor Aldoma, Nico Blodow, David Gossow, Suat Gedikli, Radu Rusu, Markus Vincze, and Gary Bradski. CAD-Model Recognition and 6 DOF Pose Estimation Using 3D Cues. In *ICCV Workshop on 3D Representation and Recognition (3dRR11)*, 2011.

Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Tutorial: Point Cloud Library – Three-Dimensional Object Recognition and 6 DoF Pose Estimation. *Robotics & Automation Magazine*, 19(3):80–91, 2012.

M. Alexa, S. Rusinkiewicz, Marc Alexa, and Anders Adamson. On normals and projection operators for surfaces defined by point sets. In *Proceedings of Eurographics Symposium on Point-based Graphics (Eurographics)*, 2004.

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and Rendering Point Set Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

A. Paul Alivisatos, Miyoung Chun, George M. Church, Ralph J. Greenspan, Michael L. Roukes, and Rafael Yuste. The Brain Activity Map Project and the Challenge of Functional Connectomics. *Neuron*, 74(6):970–974, 2012.

Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the 6th ACM Symposium on Solid Modeling and Applications (SMA)*, pages 249–266. ACM Press, 2001.

Dragomir Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Ng. Discriminative Learning of Markov random fields for Segmentation of 3D Scan Data. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 169–176, 2005.

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

S. Arya and D. M. Mount. Approximate Nearest Neighbor Searching. In *Proceedings of 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 271–280, 1993.

A. Aydemir, K. Sjöö, J. Folkesson, A. Pronobis, and P. Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2818–2824. IEEE, 2011.

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

Ferenc Balint-Benczedi, Zoltan-Csaba Marton, and Michael Beetz. Efficient Part-Graph Hashes for Object Categorization. In *5th International Conference on Cognitive Systems (CogSys)*, 2012.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up Robust Features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

Niklas Bergström, Mårten Björkman, and Danica Kragic. Generating Object Hypotheses in Natural Scenes through Human-Robot Interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 827–833, 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 *Robotics: Science and Systems Conference (RSS), Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012.

F. Beske, E. Becker, A. Katalinic, C. Krauss, and R. Pritzkuleit. Gesundheitsversorgung 2050 - Prognose für Deutschland und Schleswig-Holstein. Technical report, Fritz Beske Institut für Gesundheits-System-Forschung, Kiel, Germany, 2007. ISBN: 978-3-88312-335-6 (146 pages).

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

Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

Georg Biegelbauer and Markus Vincze. Efficient 3D Object Detection by Fitting Superquadrics to Range Image Data for Robot's Object Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy*, 2007.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

F. Bley, V. Schmirgel, and K.-F. Kraiss. Mobile Manipulation Based on Generic Object Knowledge. In *The 15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, 2006.

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)*, 2011a.

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 (Humanoids)*, pages 160–166, 2010.

Nico Blodow, Zoltan-Csaba Marton, Dejan Pangercic, Thomas Rühr, Moritz Tenorth, and Michael Beetz. Inferring Generalized Pick-and-Place Tasks from Pointing Gestures. In *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Semantic Perception, Mapping and Exploration*, 2011b.

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.

Margaret A. Boden. *Mind as Machine: A History of Cognitive Science*. Oxford University Press, Oxford, England, 2006.

Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mosenlechner, Wim Meeussen, and Stefan Holzer. Towards Autonomous Robotic Butlers: Lessons Learned with the PR2. In *ICRA*, 2011.

Gary M. Bone, Andrew Lambert, and Mark Edwards. Automated Modeling and Robotic Grasping of Unknown Three-Dimensional Objects. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

Alfons Botthof, Wolfgang Domröse, Wolfram Groß, Peter Gabriel, Katrin Gaßner, Rainer Heinstein, Monika Huber, Andreas Kerzmann, Thorsten Knape, Helko Kögel, Jörg Maas, Uwe Seidel, Hartmut Strese, Christine Weiß, Dana Mietzner, and Martin Kamprath. *Technologische und wirtschaftliche Perspektiven Deutschlands durch die Konvergenz der elektronischen Medien*. VDI/VDE Innovation + Technik GmbH and Institut für Gründung und Innovation der Universität Potsdam, 2011. http://www.autonomik.de/documents/20110630_Konvergenzstudie_Studienband.pdf.

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

Gary R. Bradski and Vadim Pisarevsky. Intel's Computer Vision Library: Applications in Calibration, Stereo, Segmentation, Tracking, Gesture, Face and Object Recognition. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2000.

Rodney Brooks. *Cambrian Intelligence: The Early History of the New AI*. The MIT Press, Cambridge, MA, 1999.

M. Callieri, A. Fasano, G. Impoco, P. Cignoni, R. Scopigno, G. Parrini, and G. Biagini. RoboScan: An Automatic System for Accurate and Unattended 3D Scanning. In *IEEE 3DPVT*, pages 805–812, 2004.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3D modeling. *ACM Transactions on Graphics*, 30: 35:1–35:10, 2011.

Yu Chen, Tae-Kyun Kim, and Roberto Cipolla. Inferring 3D shapes and deformations from single views. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, ECCV'10, pages 300–313. Springer-Verlag, 2010.

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.

Adam Coates, Paul Baumstarck, Quoc V. Le, and Andrew Y. Ng. Scalable learning for object detection with GPU hardware. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4287–4293, 2009.

Adam Coates, Honglak Lee, and Andrew Y. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24: 603–619, 2002.

Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cedric Bray. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

Kan Deng. OMEGA: On-line Memory-based General Purpose System Classifier. Technical report, PhD Thesis, Carnegie Mellon University, 1997.

Tamal K. Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 127–134, 2003.

Sven J. Dickinson. The Evolution of Object Categorization and the Challenge of Image Abstraction. In Sven J. Dickinson, Aleš Leonardis, Bernt Schiele, and Michael J. Tarr, editors, *Object Categorization: Computer and Human Vision Perspectives*, pages pp 1–37. Cambridge University Press, 2009.

J. Diebel and S. Thrun. An Application of Markov Random Fields to Range Sensing. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*. MIT Press, 2005.

K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao. *Bayesian brain: Probabilistic approaches to neural coding*. The MIT Press, 2007.

Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model Globally, Match Locally: Efficient and Robust 3D Object Recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 264–271, 2003.

Carlos Ferrari and John Canny. Planning Optimal Grasps. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, 1992.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79, 2010.

Charlotte Fiell. *1000 Chairs*. Taschen; 25th edition, 2005.

M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24, 1981.

Shachar Fleishman, Daniel Cohen-Or, and Cl&#225;udio T. Silva. Robust moving least-squares fitting with sharp features. In *ACM SIGGRAPH 2005 Papers*, pages 544–552, 2005.

Charless C. Fowlkes, David R. Martin, and Jitendra Malik. Local figure–ground cues are valid for natural images. *Journal of Vision*, 7(8), 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 28:2000, 1998.

John P. Frisby and James V. Stone. *Seeing: The Computational Approach to Biological Vision*, chapter 8: Seeing Objects, pages 178–179. MIT Press, 2010.

K.J. Friston and K.E. Stephan. Free Energy and the Brain. *Synthese*, 159(3):417–458, 2007.

Mario Fritz, Trevor Darrell, Michael Black, Gary Bradski, and Sergey Karayev. An Additive Latent Feature Model for Transparent Object Recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Giorgio Fumera and Fabio Roli. A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:942–956, 2005.

Markus Vincze Georg Biegelbauer. Efficient 3D object detection by fitting superquadrics to range image data for robot's object manipulation. *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K. Allen. The Columbia Grasp Database. In *International Conference on Robotics and Automation (ICRA)*, pages 1710–1716, 2009.

Ryan Gomes, Andreas Krause, and Pietro Perona. Discriminative Clustering by Regularized Information Maximization. *Advances in Neural Information Processing Systems 23*, pages 1–9, 2010.

M. Gopi, S. Krishnan, and C. T. Silva. Surface Reconstruction Based on Lower Dimensional Localized Delaunay Triangulation. In *Computer Graphics Forum (Eurographics)*, volume 19(3), 2000.

M. Gopi and Shankar Krishnan. A Fast and Efficient Projection-Based Approach for Surface Reconstruction. In *Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 179–186, 2002.

Lucian Cosmin Goron, Zoltan Csaba Marton, Gheorghe Lazea, and Michael Beetz. Automatic Layered 3D Reconstruction of Simplified Object Models for Grasping. In *Joint 41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*, 2010.

Lucian Cosmin Goron, Zoltan Csaba Marton, Gheorghe Lazea, and Michael Beetz. Segmenting Cylindrical and Box-like Objects in Cluttered 3D Scenes. In *7th German Conference on Robotics (ROBOTIK)*, 2012.

Stephen Gould, Olga Russakovsky, Ian Goodfellow, Paul Baumstarck, Andrew Y. Ng, and Daphne Koller. The STAIR Vision Library (v2.4), 2010. http://ai.stanford.edu/ sgould/svl.

S. Griffith, J. Sinapov, M. Miller, and A. Stoytchev. Toward interactive learning of object categories by a robot: A case study with container and non-container objects. In *IEEE 8th International Conference on Development and Learning (ICDL)*, 2009.

Gregory D. Hager and Ben Wegbreit. Scene Parsing Using a Prior World Model. *International Journal of Robotics Research (IJRR)*, 30(12):1477–1507, 2011.

Uwe Hahne and Marc Alexa. Combining Time-Of-Flight depth and stereo images without accurate extrinsic calibration. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4):325–333, 2008.

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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2013. Best

Service Robotics Paper Award Finalist.

Ulrich Hillenbrand. Pose clustering from stereo data. In *Proceedings VISAPP International Workshop on Robotic Perception – RoboPerc*, 2008.

Ulrich Hillenbrand. Non-parametric 3D shape warping. In *Proceedings of the International Conference on Pattern Recognition*, 2010.

Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Andreas Hofhauser, Carsten Steger, and Nassir Navab. Harmonic Deformation Model for Edge Based Template Matching. In *Third International Conference on Computer Vision Theory and Applications*, volume 2, pages 75–82, 2008.

Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 71–78, 1992.

Ian Horswill. Integrating Vision and Natural Language Without Central Models. In *Proceedings of the AAAI Fall Symposium on Embodied Language and Action*, 1995.

Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E. Gil Jones. Contact-Reactive Grasping of Objects with Partial Shape Information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

Daniel Huber, Anuj Kapuria, Raghavendra Rao Donamukkala, and Martial Hebert. Parts-based 3D object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

David W. Jacobs. Perceptual Organization As Generic Object Recognition. In T.F. Shipley and P.J. Kellman, editors, *From Fragments to Objects - Segmentation and Grouping in Vision*, chapter IV. Models Of Segmentation And Grouping, pages 295–329. Elsevier Science B.V., 2001.

Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

Andrew Edie Johnson. Spin-Images: A Representation for 3-D Surface Matching. Technical Report tech. report CMU-RI-TR-97-47, Robotics Institute, Carnegie Mellon University, 1997.

Doctoral dissertation.

A. Kanezaki, T. Suzuki, T. Harada, and Y. Kuniyoshi. Fast Object Detection for Robots in a Cluttered Indoor Environment Using Integral 3D Feature Table. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011a.

Asako Kanezaki, Tatsuya Harada, and Yasuo Kuniyoshi. Scale and Rotation Invariant Color Features for Weakly-Supervised Object Learning in 3D Space. In *3rd International IEEE Workshop on 3D Representation and Recognition (3dRR-11), in conjunction with ICCV*, 2011b.

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*, 2011c.

Asako Kanezaki, Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. High-speed 3D object recognition using additive features in a linear subspace. In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 3128–3134, 2010.

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, 2009a.

Ulrich Klank, Muhammad Zeeshan Zia, and Michael Beetz. 3D Model Selection from an Internet Database for Robotic Vision. In *International Conference on Robotics and Automation (ICRA)*, pages 2406–2411, 2009b.

Klaas Klasing. *Aspects of 3D Perception, Abstraction, and Interpretation in Autonomous Mobile Robotics*. PhD thesis, Technische Universität München, 2010.

J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough Transform and 3D SURF for robust three dimensional classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

Kurt Konolige, Motilal Agrawal, Robert C. Bolles, Cregg Cowan, Martin Fischler, and Brian P. Gerkey. Outdoor Mapping and Navigation using Stereo Vision. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2006.

Danica Kragic and Markus Vincze. Vision for Robotics. *Found. Trends Robot*, 1(1):1–78, 2009.

Simon Kriegel, Manuel Brucker, Zoltan-Csaba Marton, Tim Bodenmüller, and Michael Suppa. Combining Object Modeling and Recognition for Active Scene Exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

Simon Kriegel, Christian Rink, Tim Bodenmüller, Alexander Narr, Michael Suppa, and Gerd Hirzinger. Next-Best-Scan Planning for Autonomous 3D Modeling. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2850–2856, 2012.

Matthias Künzel, editor. *AUTONOMIK Begleitforschung, Multimodale Sensorik – Konzepte zur Umwelterkennung und -modellierung*. VDI/VDE Innovation + Technik GmbH, 2012. http://www.autonomik.de/documents/Autonomik_Weissbuch.pdf.

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 *Proc. of 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 *Proc. of International Conference on Robotics and Automation (ICRA)*, 2011b.

Kevin Lai and Dieter Fox. 3D Laser Scan Classification Using Web Data and Domain Adaptation. In *Robotics: Science and Systems*, 2009.

Louisa Lam and Ching Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27: 1265–1278, 2005.

Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeffrey Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML)*, 2012.

B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289, 2008.

Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 131–144. ACM Press/Addison-Wesley Publishing Co., 2000.

Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. GlobFit: Consistently Fitting Primitives by Discovering Global Relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12, 2011.

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

Dermot Lynott and Louise Connell. Modality Exclusivity Norms for 423 Object Properties. *Behavior Research Methods*, 41(2):558–564, 2009.

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

Marianna Madry, Dan Song, and Danica Kragic. 2D/3D Object Categorization for Task Based Grasping. In *RGB-D Workshop on 3D Perception in Robotics at the European Robotics (euRobotics) Forum*, 2011.

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

John Markoff. The Learning Curve: It May Be a Human Trait That Can Be Mimicked. *The New York Times International Weekly*, 2012.

David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982.

Zoltan-Csaba Marton. Object Reconstruction and Semantic Mapping from Point-Set Surfaces. Master's thesis, Technical University of Cluj Napoca, 2007. Thesis written at the Technical University of Munich.

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)*, 2012a.

Zoltan-Csaba Marton, Ferenc Balint-Benczedi, Oscar Martinez Mozos, Nico Blodow, Asako Kanezaki, Lucian Cosmin Goron, Dejan Pangercic, and Michael Beetz. Part-Based Geometric Categorization and Object Reconstruction in Cluttered Table-Top Scenes. *Journal of Intelligent & Robotic Systems*, pages 1–22, 2014.

Zoltan-Csaba Marton, Ferenc Balint-Benczedi, Oscar Martinez Mozos, Dejan Pangercic, and Michael Beetz. Cumulative Object Categorization in Clutter. In *Robotics: Science and Systems Conference (RSS), 2nd Workshop on Robotics in Clutter*, 2013.

Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, Moritz Tenorth, Radu Bogdan Rusu, and Michael Beetz. Autonomous Mapping of Kitchen Environments and Applications. In *Proceedings of the 1st International Workshop on Cognition for Technical Systems, Munich, Germany, 6-8 October*, 2008.

Zoltan Csaba Marton, Lucian Cosmin Goron, Radu Bogdan Rusu, and Michael Beetz. Reconstruction and Verification of 3D Object Models for Grasping. In *Proceedings of the 14th International Symposium on Robotics Research (ISRR09)*, 2009a.

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 (IJRR), Special Issue on Semantic Perception for Robots in Indoor Environments, Part II*, 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)*, 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.

Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009b.

Zoltan Csaba Marton, Radu Bogdan Rusu, Dominik Jain, Ulrich Klank, and Michael Beetz. Probabilistic Categorization of Kitchen Objects in Table Settings with a Composite Sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4777–4784, 2009c.

Zoltan-Csaba Marton, Florian Seidel, Ferenc Balint-Benczedi, and Michael Beetz. Ensembles of Strong Learners for Multi-cue Classification. *Pattern Recognition Letters (PRL), Special Issue on Scene Understandings and Behaviours Analysis*, 2012b.

Zoltan-Csaba Marton, Florian Seidel, and Michael Beetz. Towards Modular Spatio-temporal Perception for Task-adapting Robots. In *Postgraduate Conference on Robotics and Development of Cognition (RobotDoC-PhD), a satellite event of the 22nd International Conference on Artificial Neural Networks (ICANN)*, 2012c.

Stefan May, David Droeschel, Dirk Holz, Christoph Wiesen, and Stefan Fuchs. 3D Pose Estimation and Mapping with Time-of-Flight Cameras. In *International Conference on Intelligent Robots and Systems (IROS), 3D Mapping workshop, Nice, France, September 22-26*, 2008.

Robert Mencl and Heinrich Müller. Interpolation and Approximation of Surfaces from Three-dimensional Scattered Data Points. In *State of the Art Reports, Eurographics '98*, pages 51–67, 1998.

Sibylle Meyer. *Mein Freund der Roboter*. BMBF/VDE Innovationspartnerschaft AAL (Ambient Assisted Living), VDE-Verlag, 2011. http://www.vde.com/de/Verband/Pressecenter/Pressemeldungen/Fach-und-Wirtschaftspresse/2011/Seiten/2011-15.aspx.

Ajmal S. Mian, Mohammed Bennamoun, and Robyn Owens. Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1584–1601, 2006.

Andrew Miller and Peter K. Allen. Graspit!: A Versatile Simulator for Robotic Grasping. *IEEE Robotics and Automation Magazine*, 11(4):110–122, 2004.

G. B. Mirisola, Jorge Lobo, and Jorge Dias. 3D map registration using vision/laser and inertial sensing. In *European Conference on Mobile Robots (ECMR2007), Freiburg, Germany, Sep.*, 2007.

Ajay K Mishra and Yiannis Aloimonos. Visual Segmentation of "Simple" Objects for Robots. In *Robotics: Science and Systems (RSS)*, 2011.

Benoit Morisset, Radu Bogdan Rusu, Aravind Sundaresan, Kris Hauser, Motilal Agrawal, Jean-Claude Latombe, and Michael Beetz. Leaving Flatland: Toward Real-Time 3D Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, May 12-17*, 2009.

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.

Marius Muja and David G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340. INSTICC Press, 2009.

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

Matthias Nieuwenhuisen, David Droeschel, Dirk Holz, Joerg Stückler, Alexander Berner, Jun Li, Reinhard Klein, and Sven Behnke. Mobile Bin Picking with an Anthropomorphic Service Robot. *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Denis Noble. *The Music of Life: Biology Beyond Genes*. Oxford University Press, 2006.

Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. Accurate Object Localization in 3D Laser Range Scans. In *International Conference on Advanced Robotics*, pages 665–672, 2005.

K. Okada, M. Kojima, S. Tokutsu, T. Maki, Y. Mori, and M. Inaba. Multi-cue 3D object recognition in knowledge-based vision-guided humanoid robot system. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, pages 3217–3222, 2007.

Stephen E. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, Cambridge, Massachusetts, 1999.

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, 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.

Mark Pauly, Markus Gross, and Leif Kobbelt. Efficient Simplification of Point-Sampled Surfaces. In *Proceedings of IEEE Visualization*, 2002.

Nadia Payet and Sinisa Todorovic. From Contours to 3D Object Detection and Pose Estimation. In *13th International Conference on Computer Vision (ICCV)*, 2011.

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

M.-T. Pham, O. J. Woodford, F. Perbet, A. Maki, B. Stenger, and R. Cipolla. A New Distance for Scale-Invariant 3D Shape Recognition and Registration. In *13th International Conference on Computer Vision (ICCV)*, 2011.

Axel Pinz. Object categorization. *Found. Trends. Comput. Graph. Vis.*, 1:255–353, 2005.

Richard Pito. A Solution to the Next Best View Problem for Automated Surface Acquisition. *IEEE PAMI*, 21(10):1016–1030, 1999.

Ingmar Posner, Mark Cummins, and Paul Newman. Fast Probabilistic Labeling of City Maps. In *Proceedings of Robotics: Science and Systems*. MIT Press, 2008.

Christian Potthast and Gaurav S. Sukhatme. Next Best View Estimation With Eye In Hand Camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): The PR2 Workshop*, 2011.

Morgan Quigley, Siddharth Batra, Stephen Gould, Ellen Klingbeil, Quoc V. Le, Ashley Wellman, and Andrew Y. Ng. High-Accuracy 3D Sensing for Mobile Manipulation: Improving Object Detection and Door Opening. In *ICRA*, 2009.

Michael K. Reed and Peter K. Allen. Constraint-Based Sensor Planning for Scene Modeling. *IEEE PAMI*, 22(12):1460–1467, 2000.

A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4791–4796, 2012.

Mario Richtsfeld and Markus Vincze. Grasping of Unknown Objects from a Table Top. In *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments*, 2008.

Alvaro Collet Romea, Dmitry Berenson, Siddhartha Srinivasa, , and David Ferguson. Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA '09)*, 2009.

S. Ruiz-Correa, L. G. Shapiro, and M. Meila. A new paradigm for recognizing 3-D object shapes from range data. In *International Conference on Computer Vision*, 2003.

Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universität München, Germany, 2009. Advisor: Univ.-Prof. Michael Beetz (TUM) Ph.D.; Committee: Univ.-Prof. Dr. Nassir Navab (TUM), Univ.-Prof. Michael Beetz (TUM) Ph.D., Prof. Kurt Konolige (Stanford) Ph.D., Prof. Gary Bradski (Stanford) Ph.D.; summa cum laude.

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009a.

Radu Bogdan Rusu, Nico Blodow, Zoltan-Csaba Marton, Alina Soos, and Michael Beetz. Towards 3D Object Maps for Autonomous Household Robots. In *Proceedings of the 20th IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

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, Andreas Holzbach, Nico Blodow, and Michael Beetz. Fast Geometric Point Labeling using Conditional Random Fields. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009b.

Radu Bogdan Rusu, Andreas Holzbach, Gary Bradski, and Michael Beetz. Detecting and Segmenting Objects for Mobile Manipulation. In *Proceedings of IEEE Workshop on Search in 3D and Video (S3DV), held in conjunction with the 12th IEEE International Conference on Computer Vision (ICCV)*, 2009c.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Learning Informative Point Classes for the Acquisition of Object Model Maps. In *Proceedings of the*

10th International Conference on Control, Automation, Robotics and Vision (ICARCV), Hanoi, Vietnam, December 17-20, 2008a.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent Point Feature Histograms for 3D Point Clouds. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10), Baden-Baden, Germany*, 2008b.

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, 2008c.

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)*, 2009d.

S. Savarese and L. Fei-Fei. Multi-view Object Categorization and Pose Estimation. *Studies in Computational Intelligence- Computer Vision*, pages 205–231, 2010.

Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic Grasping of Novel Objects using Vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

Carlos E. Scheidegger, Shachar Fleishman, and Cláudio T. Silva. Triangulating point set surfaces with bounded error. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 63, 2005.

Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

M.J. Schuster, J. Okerman, H. Nguyen, J.M. Rehg, and C.C. Kemp. Perceiving clutter and surfaces for object placement in indoor environments. In *Proceedings of the 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 152 –159, 2010.

Martin Sewell. Ensemble learning. Published online: http://machine-learning.martinsewell.com/ensembles/ensemble-learning.pdf, 2007.

Jamie Shotton, Andrew Blake, and Roberto Cipolla. Multiscale Categorical Object Recognition Using Contour Fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270–1281, 2008.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision*, 2007.

Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. Feature-Weighted Linear Stacking. *CoRR*, abs/0911.0460, 2009.

Jivko Sinapov and Alexander Stoytchev. Object Category Recognition by a Humanoid Robot Using Behavior-Grounded Relational Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

Manish Singh and Donald D. Hoffman. Part-Based Representations Of Visual Shape And Implications For Visual Cognition. In T.F. Shipley and P.J. Kellman, editors, *From Fragments to Objects - Segmentation and Grouping in Vision*, chapter IV. Models Of Segmentation And Grouping, pages 401–459. Elsevier Science B.V., 2001.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *28th International Conference on Machine Learning (ICML)*, pages 129–136, 2011.

G. Somanath, M.V. Rohith, D. Metaxas, and C. Kambhamettu. D - Clutter: Building object model library from unsupervised segmentation of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2783 –2789, 2009.

Roger W. Sperry. Neurology and the mind-body problem. *American Scientist*, 40(2):291–312, 1952.

Darko Stanimirovic, Zoltan-Csaba Marton, and Michael Suppa. On Texture and Geometric Feature Based RGB-D Registration. In *3rd Semantic Mapping, Perception and Exploration Workshop, in conjunction with ICRA*, 2013.

B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. NARF: 3D Range Image Features for Object Recognition. In *IROS Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics*, 2010.

Bastian Steder, Giorgio Grisetti, Mark Van Loock, and Wolfram Burgard. Robust On-line Model-based Object Detection from Range Images. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

L. Steels and R. A. Brooks, editors. *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1995.

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, W. Burgard, and D. Cremers. Evaluating Egomotion and Structure-from-Motion Approaches Using the TUM RGB-D Benchmark. In *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, 2012.

Min Sun, Gary Bradski, Bing-Xin Xu, and Silvio Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 658–671. Springer-Verlag, 2010.

Michael Suppa. *Autonomous Robot Work Cell Exploration using Multisensory Eye-in-Hand Systems*. PhD thesis, Leibniz Universität Hannover, 2008.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer-Verlag New York, Inc., New York, NY, USA, 2010.

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

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.

Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Winning the DARPA Grand Challenge. *Journal of Field Robotics*, 2006.

Sebastian Thrun and Ben. Wegbreit. Shape From Symmetry. In *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2005.

Kai Ming Ting and Ian H. Witten. Stacked Generalization: When Does it Work? In *in International Joint Conference on Artificial Intelligence (IJCAI)*, pages 866–871, 1997.

Manuel Martinez Torres, Alvaro Collet Romea, and Siddhartha Srinivasa. MOPED: A Scalable and Low Latency Object Recognition and Pose Estimation System. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

Rudolph Triebel, Kristian Kersting, and Wolfram Burgard. Robust 3D Scan Point Classification Using Associative Markov Networks. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2603–2608, 2006.

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

Alan M. Turing. Computing Machinery and Intelligence. *Mind*, LIX:433–460, 1950.

Alan M. Turing. Intelligent Machinery. *Machine Intelligence*, 5, 1970. Different sources cite 1947 and 1948 as the time of writing.

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, 2009a.

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

Vladyslav Usenko, Florian Seidel, Zoltan-Csaba Marton, and Dejan Pangercic Michael Beetz. Furniture Classification using WWW CAD Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception (ASP'12)*, 2012.

Koen E. A. van de Sande, Theo Gevers, and Cees G. M. Snoek. Color Descriptors for Object Category Recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 378–381, 2008.

David Vernon. Cognitive vision: The case for embodied perception. In *Image and Vision Computing*. Elsevier, 2005.

R. Vincent, F. Dieter, J. Ko, K. Konolige, B. Limketkai, B. Morisset, C. Ortiz, D. Schulz, and B. Stewart. Distributed Multirobot Exploration, Mapping, and Task Allocation. In *Special Issue on Multi-Robot Coverage, Search, and Exploration*. Annals of Math and Artificial Intelligence journal (AMAI), 2008.

Kiri Wagstaff. Machine Learning that Matters. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML)*, 2012.

Markus Waibel, Michael Beetz, Raffaello D'Andrea, Rob Janssen, Moritz Tenorth, Javier Civera, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, J.M.M. Montiel, Alexander Perzylo, Björn Schießle, Oliver Zweigle, and René van de Molengraft. RoboEarth - A World Wide Web for Robots. *Robotics & Automation Magazine*, 18(2):69–82, 2011.

Jianning Wang and Manuel M. Oliveira. A Hole-Filling Strategy for Reconstruction of Smooth Surfaces in Range Images. *XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, 00:11, 2003.

S. Watanabe and N. Pakvasa. Subspace method in pattern recognition. In *Proceedings of 1st International Joint Conference on Pattern Recognition*, 1973.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

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/.

K. Wyrobek, E. Berger, H.F.M. Van der Loos, and K. Salisbury. Towards a Personal Robotics Development Platform: Rationale and Design of an Intrinsically Safe Personal Robot. In *Proc. International Conference on Robotics and Automation (ICRA)*, 2008.

Zhixing Xue, Alexander Kasper, Marius J. Zoellner, and Rüdiger Dillmann. An automatic grasp planning system for service robots. In *International Conference on Advanced Robotics (ICAR)*, 2009.

Ilker Yildirim and Robert A. Jacobs. Transfer of Object Category Knowledge Across Visual and Haptic Modalities: Experimental and Computational Studies. *Cognition*, 126(2):135–148, 2013.

Yan Zhang, Andreas Koschan, and Mongi A. Abidi. Superquadric Representation of Automotive Parts Applying Part Decomposition. *Journal of Electronic Imaging, Special Issue on Quality Control by Artificial Vision, Vol. 13, No. 3*, pages 411–417, 2004.

Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

Jiejie Zhu, Liang Wang, Ruigang Yang, and James Davis. Fusion of Time-of-Flight Depth and Stereo for High Accuracy Depth Maps. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

Muhammad Zeeshan Zia, Ulrich Klank, and Michael Beetz. Acquisition of a Dense 3D Model Database for Robotic Vision. In *International Conference on Advanced Robotics (ICAR)*, 2009.