
INSTITUTE FOR COMPUTER SCIENCE
KNOWLEDGE-BASED SYSTEMS GROUP

# Context-aware anchoring, semantic mapping and active perception for mobile robots

*Dissertation*

zur Erlangung des Doktorgrades (Dr. rer. nat.)
des Fachbereichs Mathematik/Informatik
der Universität Osnabrück

vorgelegt von

*Martin Günther*

April 2021

| | |
|---|---|
| First supervisor: | Prof. Dr. Joachim Hertzberg |
| Second supervisor: | Prof. Dr. Alessandro Saffiotti |

## Abstract

An autonomous robot that acts in a goal-directed fashion requires a world model of the elements that are relevant to the robot's task. In real-world, dynamic environments, the world model has to be created and continually updated from uncertain sensor data. The symbols used in plan-based robot control have to be anchored to detected objects. Furthermore, robot perception is not only a bottom-up and passive process: Knowledge about the composition of compound objects can be used to recognize larger-scale structures from their parts. Knowledge about the spatial context of an object and about common relations to other objects can be exploited to improve the quality of the world model and can inform an active search for objects that are missing from the world model.

This thesis makes several contributions to address these challenges: First, a model-based semantic mapping system is presented that recognizes larger-scale structures like furniture based on semantic descriptions in an ontology. Second, a context-aware anchoring process is presented that creates and maintains the links between object symbols and the sensor data corresponding to those objects while exploiting the geometric context of objects. Third, an active perception system is presented that actively searches for a required object while being guided by the robot's knowledge about the environment.



## Zusammenfassung

Autonome Roboter benötigen ein Weltmodell derjenigen Elemente in seiner Umgebung, die für ihre jeweilige Aufgabe relevant sind, um zielgerichtet handeln zu können. In dynamischen Umgebungen muss das Weltmodell aus unsicheren Sensordaten erzeugt und kontinuierlich aktuell gehalten werden. Symbole, die in der planbasierten Robotersteuerung verwendet werden, müssen in tatsächlich erkannten Objekten verankert werden. Zudem ist Roboterwahrnehmung nicht ein Prozess, der allein Bottom-Up und passiv geschieht: Wissen über die Zusammensetzung von Objekten kann genutzt werden, um größere Strukturen aus ihren Teilen zu erkennen. Das Wissen über den räumlichen Kontext eines Objektes kann dabei helfen, die Qualität des Weltmodells zu steigern, und kann eine aktive Suche nach fehlenden Objekten lenken.

Die vorliegende Arbeit leistet die folgenden Beträge zur Lösung dieser Probleme: Erstens wird ein Verfahren zur modellbasierten semantischen Kartierung vorgestellt, das größere Strukturen (wie Möbel) basierend auf semantischen Beschreibungen aus einer Ontologie erkennt. Zweitens wird ein kontextbasierter Anchoringprozess präsentiert, welcher die Verbindungen zwischen Objektsymbolen und den entsprechenden Sensordaten erzeugt und aktuell hält und

dabei den geometrischen Kontext der Objekte berücksichtigt. Drittens wird ein System zur aktiven Wahrnehmung vorgestellt, welches aktiv nach einem benötigten Objekt sucht und dabei das Wissen über die Umgebung des Roboters ausnutzt.

# Acknowledgments

First of all, I would like to thank my supervisor Prof. Dr. Joachim Hertzberg for his invaluable supervision, support and encouragement during the writing of this dissertation. I always knew that I could come to him for help and advice, no matter how busy he was at the time, and that he would always have my back.

I would like to express my sincere gratitude to Prof. Dr. Alessandro Saffiotti for agreeing to be my second supervisor and for warmly welcoming me to his research group during my research stay in Örebro. The long and intense discussions with him over the years not only provided much-needed clarity and insight but also always made me walk away with a smile.

I would like to offer my special thanks to my coauthors and colleagues from the University days Dr. Sven Albrecht, Dr. Thomas Schüler, Jochen Sprickerhof and Priv. Doz. Dr. Thomas Wiemann for the shared research and the stimulating conversations we had during our journey together.

I would also like to thank my current colleagues at DFKI, which are too numerous to name here, for providing a friendly and inspiring research environment.

I would like to extend my sincere thanks to the whole team of the RACE project for the fruitful collaboration and the sometimes exhausting, but always exhilarating code sprints, in particular Thorsten Gedicke, Štefan Konečný, Dr. Masoumeh Mansouri, Dr. Miguel Oliveira, Dr. Federico Pecora, and Dr. Sebastian Stock for the cherished time spent together in the lab, and in social settings.

I would like to thank my coauthor Dr. José Raúl Ruiz-Sarmiento for our very productive work together and all the nice moments during his stay in Osnabrück and afterward.

Lastly, I am deeply grateful to my wife, Dr. Vanessa Dizinger, and my parents Reinhard Günther and Brigitte Braun-Günther for their patience and their unwavering support and belief in me.

# Contents

# Chapter 1

# Introduction

Autonomous mobile robots have begun to make impressive inroads into both homes and factories in the last two decades. Professional service robots (mainly logistics robots, but also other fields such as defense, milking robots, public relations, exoskeletons, and medical robots) have sold 173,000 units in 2019, while service robots for domestic and personal use (vacuum cleaning, lawn mowing, window cleaning and other types) have sold more than 18.6 million units in the same year, both with a yearly growth of over 30 % (International Federation of Robotics, 2020).

At the moment, these robots are very specialized and focused on simple tasks which do not require object manipulation and only limited perception and semantics. For most of the tasks listed above, it suffices to recognize obstacles, and the semantic properties of the environment can be mostly ignored.

The most advanced robot prototypes, which include semantic environment perception, are found in the research community; the progress in this field can be observed for example in the annual RoboCup@Home competition (van der Zant and Wisspeintner, 2005; Wisspeintner et al., 2009). These academic developments are already approaching a point where they are ready to be employed in real-world scenarios, as is evidenced by the rise of a number of start-ups such as Fetch Robotics[1] or Magazino[2], who start to offer more advanced robots for tasks such as order picking and commissioning in commercial warehouses.

At the same time, a new generation of "safe" robots that a suitable for human-robot collaboration —also known as cobots— have sprung up in the industrial community (de Gea Fernández et al., 2017a,b; Villani et al., 2018). These robots are intended to work closely together

[1] `https://fetchrobotics.com/`
[2] `https://www.magazino.eu/`

with humans in dynamic environments, instead of being separated from human workers by a safety fence, and are no longer limited to repetitive tasks in the carefully controlled environment of a robot work cell. To be able to respond to mass customization and rapidly changing requirements, several carmakers are already replacing the traditional conveyor-belt-based manufacturing processes that have existed since Ford's days with a newer, more dynamic system, where robots transport the parts that are needed for a given assembly step between work stations that are manned by workers and collaborative robots. All of these developments are expected to continue to accelerate in the coming years.

One essential skill that all these advanced autonomous robots require is robot perception, i.e., a deeper semantic understanding of the dynamic environment they operate in. Robot perception goes beyond pure bottom-up object recognition: It has to take the robot's knowledge about its environment and the context of the scene into account, and it has to bridge the gap between raw sensor data and the high-level (usually plan-based) robot control. This is the core of this thesis.

## 1.1 Motivation

Consider the case of a robot working as a waiter in a restaurant (Figure 1.1). It has to fulfill a variety of tasks: setting a table for new guests, serving meals and beverages to a guest, cleaning a table, filling a dishwasher and so on. To fulfill these tasks, there are many things the robot needs to know about its environment:

- Where are the pieces of furniture that are relevant to the task (tables, bar, chairs, counter, dishwasher, fridge, ...)?
- Where are the relevant objects (plates, cutlery, glasses, mugs, salt and pepper mills, menus, table signs, ...)?
- How to search efficiently for a missing object? I.e., what are the most likely places to look for an object? For example, some restaurants have a shared pepper mill that is brought to a table with certain meals and retrieved by the waiter later. What if the guest has moved the pepper mill in the meantime?

When our exemplary robot was delivered from the factory, it was already equipped with certain general knowledge about generic restaurants: An ontology of types of furniture, CAD models of the furniture, some basic object recognition capabilities. However, when it is first

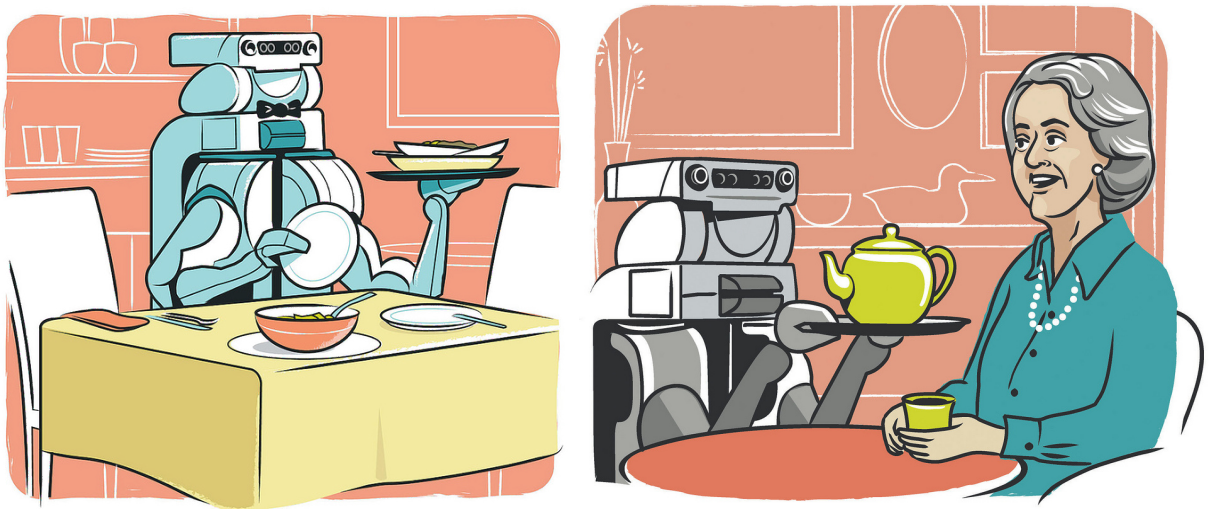


**Figure 1.1:** Artist's conception of the PR2 robot working as a waiter in a restaurant: setting a table and serving tea. Image source: `http://ellingson.cc/willow`, artwork by Joshua Ellingson

turned on, it still needs to adapt to the specific layout of the restaurant that it is operating in; for example, it needs to create a semantic map of all the furniture that is present in the restaurant. Moreover, it needs to continuously keep track of all the object locations and the changes in the environment while it is operating. Also, when the plan-based robot control requires an object that is not in the robot's current world model, the robot needs to actively search for this object while taking into account known objects and knowledge about probable locations of the target object.

Concretely, the robot in our example performs an initial exploration of the environment while recording point clouds from its RGB-D cameras and/or 3D laser scanners. These point clouds are continually registered using 6DoF SLAM, and a mesh representation is created. The mesh representation is matched against an OWL-DL knowledge base of furniture, and hypotheses for possible pieces of furniture are generated. These hypotheses are then verified by matching CAD models of the furniture into the accumulated point cloud. This is further described in Chapter 3.

Given the semantic map of the environment and its furniture, the robot can start planning and executing tasks, such as completing a table setting, i.e., ensuring that there is a complete place setting consisting of a plate, a spoon and a fork in front of every chair. An important part of this task is to recognize which objects are already present on the table. Since the spatial context in which an object appears can help to disambiguate the object class in the face of noisy

object recognition, the robot uses a probabilistic graph representation of the scene to improve the results of a baseline object recognition system (see Figure 1.2). It also continually anchors symbols used by the planner to objects detected through its sensors. Details on the context-aware anchoring system can be found in Chapter 4.

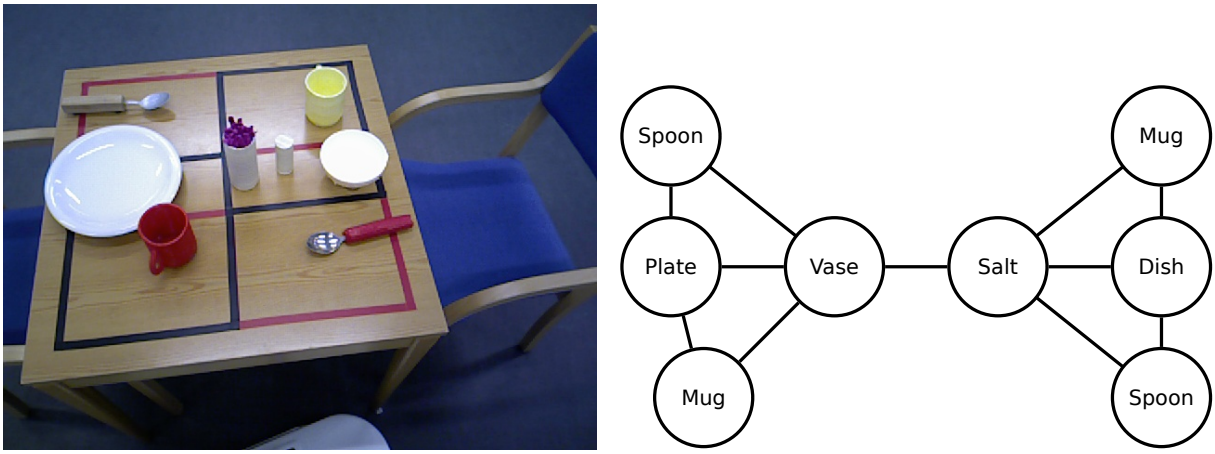



**Figure 1.2:** Left: A table setting from the point of view of the robot's RGB camera; right: a graph representation of the same scene.

Both modules described so far (semantic mapping and context-aware anchoring) are passive, i.e., they continually process data that is perceived through the robot's sensors. However, an autonomous robot also needs to actively navigate, re-position its sensors and manipulate the environment to gather the necessary information to fulfill tasks such as object search (to find an object that is required for a task) or exploration (for example, to explore all occluded regions of the table that have a certain chance of hiding an object in the case of our "complete a table setting" example). This is the role of the final module, the active perception system. It continually builds and updates an octomap of the environment (see Figure 1.3) to track known and unknown space. For each region of unknown space, it tracks the probability of containing a target object. This probability could be supplied, for example, by sampling the Probabilistic Graphical Model used by the context-aware anchoring system. The active perception system then plans and executes the sequence of actions (re-positioning the sensors by moving head and torso, navigating to a different pose, moving away occluding objects) that has the lowest expected time until a target object is found. Since the execution of those actions provides new viewpoints for the camera, the passive context-aware anchoring system (which is running in the background) also profits from this. The active perception module is the topic of Chapter 5.

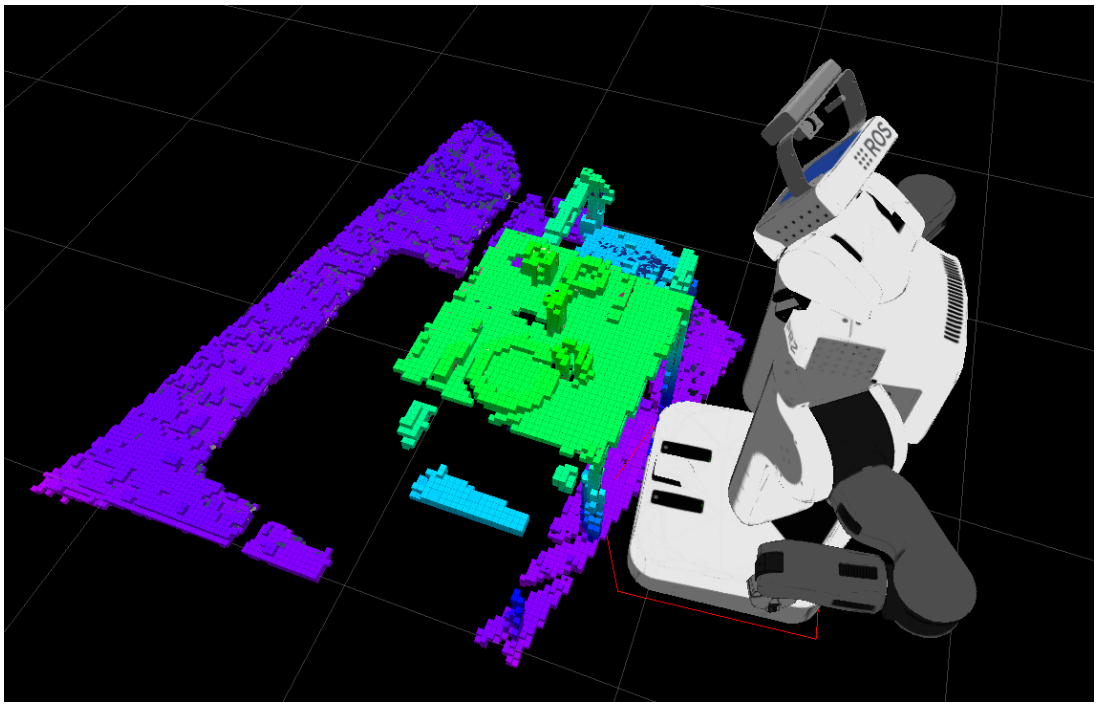



**Figure 1.3:** An octomap of the table scene from Figure 1.2.

While the thesis focuses on robot perception, it should be made clear that the focus is not on pure computer vision. Most computer vision research focuses on recognizing objects in a single RGB or RGB-D frame while being agnostic of the circumstances of where and when this RGB frame was taken. While this task is an essential building block of any robot perception system, robot perception should go beyond just narrow passive bottom-up perception. Robots usually have a wealth of information about the situation that an image was taken in: The location, previously detected (or manipulated) objects nearby, customary arrangements of certain objects, the part hierarchy of composite objects and so on. This top-down knowledge needs to be integrated with bottom-up object detection to create a full robot perception system. Moreover, robot perception is not necessarily passive: Usually, a robot actively points its sensors towards a location and runs object recognition to locate a certain object, and it needs to figure out how to do this most efficiently based on its knowledge of the situation.

Note that none of this should not be taken as a dismissal of common computer vision methods; rather, that this thesis tackles a different hard problem while integrating with and complementing computer vision.

## 1.2 Scientific Contribution

The scientific contribution of this thesis can be summarized in the following three points:

**Model-based Semantic Mapping** A model-based semantic mapping system is presented that recognizes larger-scale structures like furniture based on semantic descriptions in an OWL-DL ontology. The system works incrementally and in a closed loop, based on RGB-D images captured by a mobile robot. The result is a mesh representation of the static structures in the robot's environment, enriched by CAD models of the recognized pieces of furniture.

**Context-aware Anchoring** A context-aware anchoring process is presented that creates and maintains the links between object symbols and the sensor data corresponding to those objects while exploiting the geometric context of objects to improve the classification accuracy of a baseline local object recognition system. The system works online and incrementally aggregates information from RGB-D images captured by a mobile robot, which enables it to take into account contextual relations with objects that are outside the current sensor view.

**Active Perception** An active perception system is presented that explores the environment in a goal-directed fashion and actively searches for a required object while being guided by the robot's probabilistic knowledge about the environment (which could be supplied by the anchoring system, for example). New possible sensing actions are generated using a frontier-based 3D view sampling technique. The algorithm finds objects in less time (on average) than state-of-the-art greedy algorithms by continually planning multiple sensing actions ahead.

## 1.3 Thesis Outline

The remainder of this thesis is organized as follows:

**Chapter 2** contains the background and an introduction to the relevant parts of the RACE project, from which this thesis originates.

**Chapter 3** describes the model-based semantic mapping system for recognition and mapping of large-scale structures based on an OWL-DL ontology.

**Chapter 4** presents the context-aware anchoring system, which maintains the link between symbols used by the planner and objects recognized from the robot's sensors.

**Chapter 5** introduces the active perception system that actively plans and executes perception actions on the robot.

**Chapter 6** concludes this thesis and discusses possible avenues for future research based on this work.

## 1.4 Cooperations with Other Researchers

This thesis did not emerge from a vacuum; rather, it was written and developed in the context of different cooperations with several other researchers, mostly from the EU project RACE. These cooperations have resulted in a fruitful exchange of ideas and several joint publications. For this reason, a comprehensive account of the systems developed in this thesis necessarily includes descriptions of the parts developed by these other authors. Therefore, this section provides a differentiation of the author's work from that of his co-authors and colleagues.

- The model-based semantic mapping system presented in Chapter 3 was developed in close collaboration with my then-coworkers at Osnabrück University, Thomas Wiemann and Sven Albrecht. The meshing tools described in Section 3.2.1 were developed by Thomas Wiemann, and Sven Albrecht contributed the CAD model matching part of the system described in Section 3.2.3.

- José Raúl Ruiz-Sarmiento from the University of Málaga stayed at Osnabrück University as a guest researcher from February through May 2014, which sparked long-lasting cooperation with the author of this thesis. During his stay in Osnabrück, J.R. Ruiz-Sarmiento started to develop the Undirected Probabilistic Graphical Models in a C++ library (UPGMpp) in order to integrate it with the context-aware anchoring system described in Chapter 4 (Sections 4.2.3.1, 4.2.3.3 and 4.2.3.4).

- In the context of the RACE project, Miguel Oliveira, Gi Hyun Lim, S. Hamidreza Kasaei, Aneesh Chauhan and Luís Seabra Lopes from the Universidade de Aveiro developed the local object recognition system employed by the context-aware anchoring system (Section 4.2.1).

- Thorsten Gedicke from Osnabrück University worked as a student assistant in the RACE project and later completed his master's thesis (Gedicke, 2015), which was co-supervised by the author. During this time, he played a major part in the development of the forward-looking active perception system covered in Chapter 5. He also participated in the implementation of the anchoring system (Chapter 4).

- The RACE blackboard was mainly developed by Pascal Rost, Stephanie von Riegen and Lothar Hotz from the University of Hamburg / HITeC e.V. (Section 2.3).

## 1.5 Publications

The contributions described in this dissertation have in part first been published in conference or journal publications. The following list gives an overview. All publications predated the text of this thesis, with the exception of Günther et al. (2018), which is a revised version of Chapter 4.

**Chapter 2 (Background: The RACE project)**

- Joachim Hertzberg, Jianwei Zhang, Liwei Zhang, Sebastian Rockel, Bernd Neumann, Jos Lehmann, Krishna S. R. Dubba, Anthony G. Cohn, Alessandro Saffiotti, Federico Pecora, Masoumeh Mansouri, Štefan Konečný, Martin Günther, Sebastian Stock, Luís Seabra Lopes, Miguel Oliveira, Gi Hyun Lim, Hamidreza Kasaei, Vahid Mokhtari, Lothar Hotz, and Wilfried Bohlken. The RACE project. *KI - Künstliche Intelligenz*, 28(4):297–304, 2014. ISSN 0933-1875. doi: 10.1007/s13218-014-0327-y

- Sebastian Rockel, Bernd Neumann, Jianwei Zhang, Krishna S. R Dubba, Anthony G. Cohn, Štefan Konečný, Masoumeh Mansouri, Federico Pecora, Alessandro Saffiotti, Martin Günther, Sebastian Stock, Joachim Hertzberg, Ana Maria Tomé, Armando J. Pinho, Luís Seabra Lopes, Stephanie von Riegen, and Lothar Hotz. An ontology-based multi-level robot architecture for learning from experiences. In *Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, Stanford, USA, March 2013

- Sebastian Stock, Martin Günther, and Joachim Hertzberg. Generating and executing hierarchical mobile manipulation plans. In *Proceedings of ISR/Robotik 2014; 41$^{st}$ International Symposium on Robotics*, pages 1–6. VDE, 2014

**Chapter 3 (Model-Based Semantic Mapping)**

- Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Model-based furniture recognition for building semantic object maps. *Artif. Intell.*, 247:336–351, June 2017. doi: 10.1016/j.artint.2014.12.007. Available online: Jan 23, 2014
- Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Building semantic object maps from sparse and noisy 3D data. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 2228–2233. IEEE, 2013. doi: 10.1109/IROS.2013.6696668
- Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Model-based object recognition from 3D laser data. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings*, volume 7006 of *Lecture Notes in Computer Science*, pages 99–110. Springer, 2011. doi: 10.1007/978-3-642-24455-1_9
- Sven Albrecht, Thomas Wiemann, Martin Günther, and Joachim Hertzberg. Matching CAD object models in semantic mapping. In *Proc. ICRA 2011 workshop: Semantic Perception, Mapping and Exploration, SPME '11*, Shanghai, China, 2011

**Chapter 4 (Context-Aware Anchoring)**

- Martin Günther, José Raúl Ruiz-Sarmiento, Cipriano Galindo, Javier González-Jiménez, and Joachim Hertzberg. Context-aware 3D object anchoring for mobile robots. *Robot. Auton. Syst.*, 110:12–32, December 2018. doi: 10.1016/j.robot.2018.08.016
- José Raúl Ruiz-Sarmiento, Martin Günther, Cipriano Galindo, Javier González-Jiménez, and Joachim Hertzberg. Online context-based object recognition for mobile robots. In *17th International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, April 2017c

**Chapter 5 (Forward-Looking Active Perception)**

- Thorsten Gedicke, Martin Günther, and Joachim Hertzberg. FLAP for CAOS: Forward-looking active perception for clutter-aware object search. In *Proc. 9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, volume 49 (15) of *IFAC-PapersOnLine*, pages 114–119, Leipzig, Germany, June 2016. IFAC. doi: 10.1016/j.ifacol.2016.07.718

**Other activities**

As part of the 3rd Lucia Winter School on Artificial Intelligence and Robotics in Örebro, Sweden (December 11-16, 2016), the model-based semantic mapping system (Chapter 3) and the forward-looking active perception system (Chapter 5) were taught to a group of 30 graduate students and junior researchers as part of a full-day lecture and tutorial session.

# Chapter 2

# Background: The RACE Project

A major part of this thesis was developed within the RACE[1] (Robustness by Autonomous Competence Enhancement) project. RACE was a European FP7 project which ran from December 2011 to November 2014 and involved 6 partners: the University of Hamburg, the University of Leeds, Örebro University, Osnabrück University, Universidade de Aveiro and HITeC e.V.

## 2.1 RACE Project Description

According to the RACE project's Description of Work (2011), the overall aim of the project was (emphasis mine):

> ...to develop an artificial cognitive system, embodied by a service robot, able to build a high-level understanding of the world it inhabits by storing and exploiting appropriate memories of its experiences. Experiences will be recorded internally at multiple levels: **high-level descriptions** in terms of goals, tasks and behaviors, **connected to** constituting subtasks, and finally to **sensory and actuator skills at the lowest level**. In this way, experiences provide a detailed account of how the robot has achieved past goals or how it has failed, and what **sensory events** have accompanied the activities.
>
> The project aims to produce the following key results:
>
> (i) Robots capable of storing experiences in their memory in terms of multi-level

[1] `http://project-race.eu/`

representations **connecting actuator and sensory experiences with meaningful high-level structures**,

(ii) Methods for learning and generalizing from experiences obtained from behavior in realistically scaled real-world environments,

(iii) Robots demonstrating superior robustness and effectiveness in new situations and unknown environments using experience-based planning and behavior adaptation.

While the overarching aim of the project was on learning and generalizing from experiences, this was tackled by other project partners and is not the topic of this thesis. Instead, the modules presented in this thesis fit into the overall goal stated above by generating symbolic and metric "sensory events" from raw sensor data, and connecting them to high-level descriptions that are used both for analyzing and learning from experiences (offline) and for planning and execution monitoring (online).

For a more detailed overview of the RACE project, see Hertzberg et al. (2014); Rockel et al. (2013). Details on how the modules from this thesis fit into the overall RACE architecture will be discussed in the following section.

## 2.2 The RACE Architecture

The software architecture of RACE is centered around the so-called Blackboard: A knowledge base for symbolic temporally valid facts (so-called "fluents") with metric properties. Due to its central nature, Section 2.3 is dedicated to the Blackboard; this section describes the overall architecture.

Figure 2.1 gives an overview of the major software components and their interfaces in RACE. The figure also displays the two major feedback loops in the architecture: one fast, online feedback loop for plan-based robot control, and one slow, offline feedback loop that enables the robot to learn from experiences.

The online feedback loop (blue in Figure 2.1) starts with the **planner**, which builds its initial state representation from fluents on the blackboard that represent the robot's state (the robot position and the posture of the arms, torso and grippers), the position and state of objects and guests, and the goal given by the user. Furthermore, the knowledge on the blackboard is enriched by several spatial, temporal, resource and ontological **reasoners** as well as a high-level scene interpretation module, all of which can be used by the planner. Initially (Stock

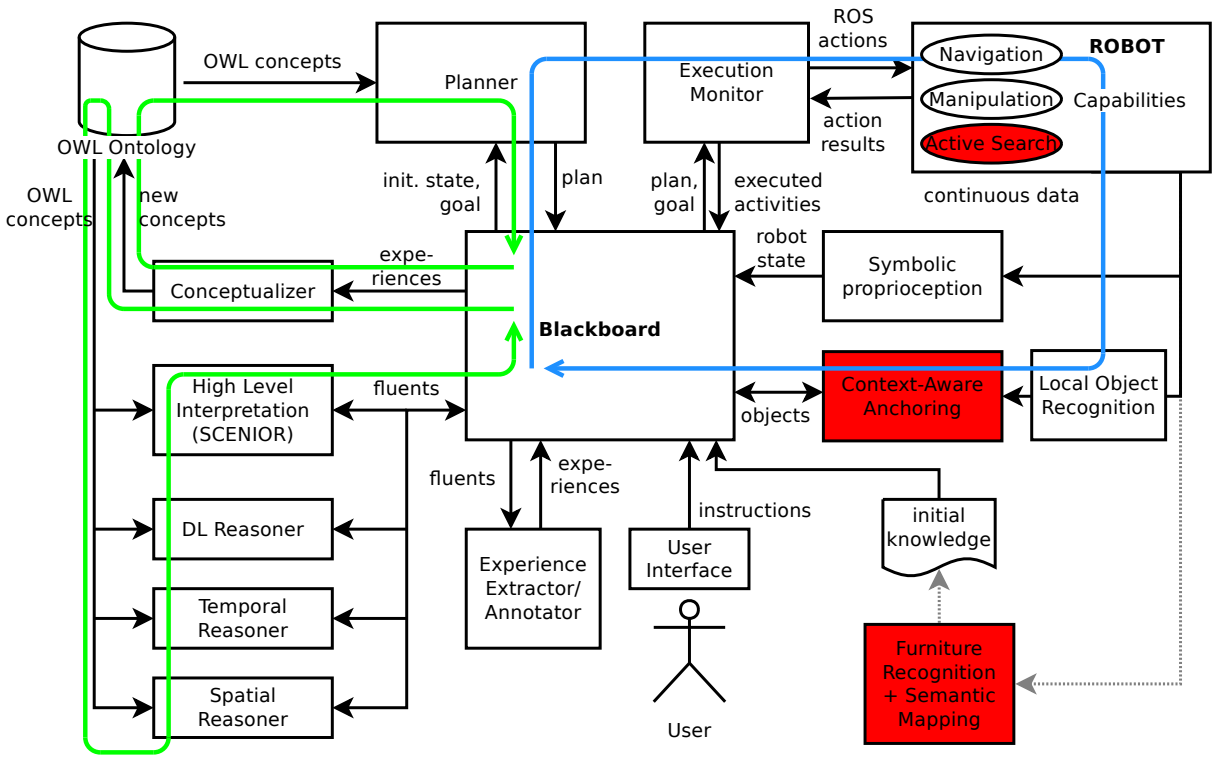



**Figure 2.1:** The RACE architecture. **Red**: the modules developed in this thesis; **blue**: the online feedback loop from planning via acting, sensing and anchoring back to planning; **green**: the offline feedback loops from experiences via planner and reasoners back to experiences. Modified from Rockel et al. (2013).

et al., 2014), RACE used the HTN planner Shop2 (Nau et al., 2001). Later, the HTN planner was replaced by the hierarchical hybrid planner CHIMP (Stock, 2017a,b).

The finished plan is passed (via the blackboard) to the **execution monitor**, which dispatches each planned action to the robot and monitors whether the actions are successfully executed. The initial implementation (Stock et al., 2014) was based on the Python-based state machine framework SMACH (Bohren et al., 2011), but was later replaced by Consistency-Based Execution Monitoring (CBEM; Konečný et al., 2014). CBEM not only monitors whether an already dispatched action aborts with an error but instead continually checks the planned and the actual world state for consistency. It does so by transforming the plan including all preconditions and effects into a temporal constraint network, which allows it to ensure temporal consistency (something that the Shop2 HTN planner does not account for), as well as to detect when it becomes impossible to fulfill the preconditions of an action that is scheduled to be executed later in the plan.

The base-level robot software provides the **capabilities** of the robot (navigation, manipulation, control of the head, torso and grippers, ...) in the form of ROS action interfaces. Most of these actions were already provided with the robot, and are implementations of generic interfaces that are provided by many other robots. This decouples the specific robot to be used from the specific control architecture (such as RACE), which promotes reuse and portability of both the base-level robot control software and the control architecture. One extra capability that was developed in the context of this thesis is **active search** (Chapter 5), which enables the robot to actively search for an object. Each ROS action provided by the robot corresponds to a (parametrizable) action that is available to the planner.

The continuous data from the robot's sensors is provided in the form of ROS topics to various modules that process it into a form suitable for storage on the blackboard. The **symbolic proprioception** simply checks the continuous state of the gripper, torso and arm joints and discretizes it into symbols for certain predefined states of the robot, such as "gripper open" or "left arm tucked". The RGB-D data from the robot's head camera is processed by the **local object recognition** system, which is based on spin images and was developed within the RACE project (Kasaei et al., 2015; Oliveira et al., 2014). The recognized objects and their confidence values are passed to the **context-aware anchoring** module (Chapter 4), which performs the final joint object recognition and anchoring to object fluents on the blackboard. The fluents representing the robot's state and objects are consumed by the planner and the execution monitor, which completes the first loop.

The second feedback loop (green in Figure 2.1) enables the robot to learn from experiences. While the robot is executing tasks, the **experience extractor/annotator** extracts a subset of the fluents on the blackboard and labels them as a new experience. Between episodes, the **conceptualizer** processes the collected experiences and updates the concepts in the OWL ontology. The OWL ontology is used as a common representation format, from which the planning domain and the knowledge used by the other reasoners are generated. In the next episode, the planner and reasoners can make use of the learned concepts, which completes the second loop. In RACE, two separate approaches for learning from experiences were developed, called Example-Based Compositional Learning (EBCL; Neumann et al., 2014) and Experience-Based Planning Domains (EBPDs; Mokhtari et al., 2016). For details on learning from experiences, we refer the reader to those publications.

Mapping (semantic or otherwise) was not the topic of the RACE project; in RACE, the positions of all task-relevant furniture objects were part of the initial knowledge that was loaded at system start, and it was assumed that a suitable semantic mapping system could produce an

identical output. The **furniture recognition and semantic mapping** system presented in this thesis (Chapter 3) fills that gap. It could be integrated into the RACE architecture by performing an initial mapping session and writing all recognized furniture instances into the Blackboard's initial knowledge, as indicated in Figure 2.1. Since both the presented furniture recognition system and the Blackboard use OWL-DL as a knowledge representation and reasoning system, they could be integrated seamlessly, giving the furniture recognition system access to all knowledge about furniture stored in the Blackboard's ontology.

## 2.3 The RACE Blackboard

The Blackboard is the central component of the RACE architecture. Most of the communication between RACE components does not happen directly, but instead by reading and writing fluents to the Blackboard. The Blackboard is implemented on top of the Sesame RDF triple store. All concepts (plans, planned and executed actions, furniture, movable objects, named areas of interest, spatial relations, the robot's state and so on) are represented in an OWL-DL ontology. For example, the concept `MoveObjectFromTo` represents a specific type of activity that was executed by the robot. Its purpose is to record the robot's activities including their related objects, areas etc., to facilitate learning from experiences. In general, each activity corresponds to a task that was emitted by the planner. Since the planning domain is hierarchical, the activities are also hierarchical (i.e., an activity can have sub-activities). `MoveObjectFromTo` is defined as the OWL class in Listing 2.1. As we can see, `MoveObjectFromTo` is derived from `MoveObjectActivity` and inherits some more properties from that superclass.

Taken together, these properties encode the following information about the activity:

**hasPassiveObject:** The object that was moved.

**hasAreaA, hasAreaB:** The named areas between which the object was moved (from A to B).

**hasGetObjects, hasPutObject:** The sub-activities of getting and putting down the object.

**hasTask:** The planned task that triggered this activity.

Of course, there are plenty of other properties inherited from superclasses further up the OWL class hierarchy, but these are omitted here for brevity. The activities tend to be one of the more complex parts of the ontology (which is why they were chosen here for demonstration purposes). Many other concepts are simpler (such as the concepts representing areas or physical objects).

**Listing 2.1** The `MoveObjectFromTo` OWL class and its superclass `MoveObjectActivity` (in Manchester OWL syntax)

```
Class:
  MoveObjectFromTo
EquivalentTo:
  MoveObjectActivity AND
  hasAreaA EXACTLY 1 Area AND
  hasPutObject EXACTLY 1 PutObjectActivity AND
  hasGetObjects EXACTLY 1 GetObjectsActivity OR
                          GetObjectsDriveRobotGraspObjectsActivity OR
                          GetObjectsFromDifferentActivity OR
                          GetObjectsGraspObjectsActivity
```

```
Class:
  MoveObjectActivity
EquivalentTo:
  RobotActivity AND
  hasTask EXACTLY 1 move_object_Task AND
  hasPassiveObject EXACTLY 1 PassiveObject AND
  hasAreaB EXACTLY 1 Area
```

Each piece of information stored on the blackboard is an instance (or, in OWL terminology, an *individual*) of a concept defined in the ontology. More precisely, in RACE each instance is a so-called *fluent*: a temporally valid predicate (i.e., a fact that holds only within a specified time interval). This modeling choice was made because facts that have become invalid cannot be simply deleted from the blackboard, but have to be retained in order to provide the full execution trace of an episode to the modules for learning from experiences. On the RACE blackboard, fluents are never removed; instead, they are "ended" by updating the finish time to a time point prior to the current time.

As an example, three fluents are shown in Listing 2.2: `pickUpObjectActivity_0X0`, `on4` and `holding0`. The example shows another peculiarity in how the validity interval of fluents are modeled: Both the start and finish times are not definite time points, but instead they are intervals themselves (from earliest possible start/finish time to latest possible start/finish time). This affords a limited modeling of uncertainty: In many cases, the exact start or finish times of a fluent are not known but can be narrowed down to an interval. In the example, the activity `pickUpObjectActivity_0X0` started at 119.621 s (when the action was dispatched) and ended at 161.817 s (when the action returned successfully). These two time points are

**Listing 2.2** Fluents representing a finished `PickUpObjectActivity` on the blackboard (in YAML syntax)

```
!Fluent
Class_Instance: [PickUpObjectActivity, pickUpObjectActivity_0X0]
StartTime: [119.621, 119.621]
FinishTime: [161.817, 161.817]
Properties:
   - [hasArm, Arm, rightArm1]
   - [hasPassiveObject, Mug, mug1]
   - [hasResult, xsd:string, succeeded]
   - [hasOn, On, on4]
   - [hasHolding, Holding, holding0]
---
!Fluent
Class_Instance: [On, on4]
StartTime: [0.0, 0.0]
FinishTime: [119.621, 161.817]
Properties:
   - [hasArea, PlacingAreaEastRight, placingAreaEastRightCounter1]
   - [hasPhysicalEntity, Mug, mug1]
   - [isInitialKnowledge, xsd:boolean, True]
---
!Fluent
Class_Instance: [Holding, holding0]
StartTime: [119.621, 161.817]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPassiveObject, Mug, mug1]
   - [hasGripper, Gripper, rightGripper1]
```

known exactly, so the earliest possible and latest possible start (resp. finish) times of that fluent are identical. At some point during the execution of this activity, `mug1` ceased to be `on placingAreaEastRightCounter1` and was instead held by `rightGripper1`. Since the dispatched action is opaque to the dispatcher, the exact time point of this transition is not known; it could have happened any time during the execution of the activity. Therefore, the finish time of `on4` and the start time of `holding0` is set to the interval `[119.621, 161.817]`. The start time of `on4` is unknown (since it was part of the robot's initial knowledge in this scenario). Likewise, the end time of `holding0` is not yet known, since it still holds (but can be updated later). To signify this, both of these are set to the special value `[0.0, 0.0]`.

In OWL, each property is a binary relation, i.e., it can only link an individual to a value

(such as an integer, a string etc.) or another individual. To represent arbitrary n-ary relations, a technique called *reification* is employed, whereby the n-ary relation is represented by an auxiliary individual. As a side effect, reification also makes it possible to specify durations for the validity of a relation, as seen previously. The two instances of the classes `On` and `Holding` in Listing 2.2 are examples of reified relations. In OWL, it is impossible to directly specify `pickUpObjectActivity_0X0.hasHolding(rightGripper1, mug1)`, since this would be a ternary relation between the three individuals `pickUpObjectActivity_0X0`, `rightGripper1` and `mug1`. Therefore, the ternary relation is split up into one individual `holding0` and three binary relations:

- `pickUpObjectActivity_0X0.hasHolding(holding0)`,
- `holding0.hasGripper(rightGripper1)` and
- `holding0.hasPassiveObject(mug1)`.

The blackboard can be queried by other modules using the SPARQL query language. For example, the following query returns all floor areas, i.e., all individuals of class `upper:Area` (or its transitive subclasses) whose bounding box pose has the $z$ coordinate 0:

```
SELECT DISTINCT ?area WHERE
{
  ?area rdf:type ?type .
  ?type rdfs:subClassOf* upper:Area .
  ?area upper:hasBoundingBox ?bbox .
  ?bbox upper:hasPose ?pose .
  ?pose upper:hasZ "0.0"^^xsd:float .
}
```

SPARQL queries can also be registered with the blackboard. When a fluent that matches a registered query is added or modified, the blackboard publishes the fluent on a specified ROS topic. This has the advantage that other modules do not have to poll the blackboard continuously, but are instead actively notified.

## 2.4 Demonstration Environment

The environment in which the demonstrations of the RACE project took place is a stylized restaurant with two tables, four chairs and a counter (see Figure 2.2). There are also varying amounts of tabletop objects, such as mugs, plates, various cutlery, pepper mills etc., depending on the concrete scenario. In some scenarios, a human guest is involved, sitting at one of the tables or blocking the path of the robot. Typical tasks of the robot were serving a mug of coffee or clearing a table.


**Figure 2.2:** The RACE demonstration environment at the University of Hamburg

In the RACE ontology, various types of areas in the restaurant that are significant for the robot's actions are represented (see Figure 2.3):

**ManipulationAreas** are areas on the floor from which the robot can reach objects in some nearby PlacingAreas.

**PreManipulationAreas** are areas on the floor that are adjacent to a ManipulationArea, but far enough from any obstacles so that they can be reached using the standard navigation components.

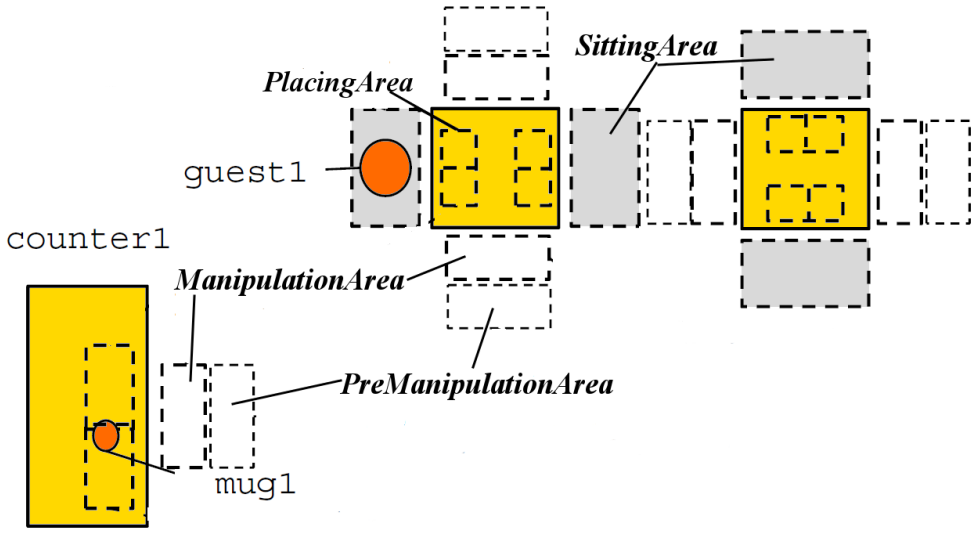



**Figure 2.3:** Schema of the named areas and scenario-relevant objects in one of the RACE demo scenarios. Figure reproduced from Hertzberg et al. (2014).

**SittingAreas** are areas on the floor which can be occupied by chairs.

**PlacingAreas** are areas on the table where objects can be placed.

The symbolic names of these areas are used by many modules in the RACE system, such as the planner, executor and anchoring. For example, `placingAreaEastRightCounter1` (from Listing 2.2 on page 17) is a fluent that is used by the planner to designate the area in which a mug is to be placed. Likewise, the executor monitors the successful placement of the mug on this area, and the anchoring module uses it to update the locations of objects that are detected on the table.

Since mapping was not in the scope of the RACE project, all areas and poses of the pieces of furniture were specified manually as part of the robot's initial knowledge. However, a semantic mapping system such as the one presented in Chapter 3 could be used to autonomously acquire this information.

## 2.5 Software Framework: ROS

The robotics software framework that is used in this thesis is ROS, the Robot Operating System (Quigley et al., 2009). Since the work started on this thesis, ROS has become the de-facto

standard robotics middleware in autonomous robotics, both in academia and in many industrial applications. Despite its name, ROS is not an operating system, but a collection of tools, libraries and conventions used to write robot software. It mainly consists of the following components:

1. A *communication middleware*. The middleware is the "plumbing" that connects different processes ("ROS nodes") and is based on a publish-subscribe pattern, which enables loose coupling between nodes. As a result, nodes can be written in many different languages, can be distributed over several computers and robots in the networks, and the whole system has a certain degree of resilience against the failure of single nodes.

2. A set of *capabilities*: Ready-to-use software packages for many common robot capabilities (navigation, manipulation, sensor data processing, ...). Most packages are contributed by the community, i.e., a large number of research labs and companies worldwide.

3. *Standards and conventions*. A large set of commonly-used data types have been standardized in the message formats that are sent over the ROS network. Also, there is a set of conventions (for example for coordinate systems or units of measurement), which are formalized in so-called ROS Enhancement Proposals (REPs). These two points are often overlooked in discussions about ROS, but are perhaps the most valuable contribution of ROS: They provide hardware independence by decoupling algorithms from specific hardware and specific implementations of a capability. For example, the Active Perception algorithm FLAP4CAOS (Chapter 5) requires a navigation capability, and all ROS-enabled mobile ground robots provide a navigation capability with the same message interface. This makes it relatively easy to adapt FLAP4CAOS to a new robot. In fact, FLAP4CAOS has been demonstrated on the PR2 robot, the Calvin robot and a group of three modified Turtlebots.

## 2.6 Robots and Simulator used in the Experiments

The main demonstration platform of the RACE project was the PR2 robot. Some experiments in this thesis have additionally been performed on Osnabrück University's Calvin and Kurtana robots and in a Gazebo simulation of the PR2 robot in the RACE demonstration environment (Section 2.4). This section gives a brief overview of these robots and the simulator. The reason why the Gazebo simulator is included in this section is that Gazebo provides the same ROS

interface as the physical PR2 robot, so from the point of view of higher-level software, the simulated robot can be treated equally to a physical robot.

Since the PR2 robot was developed and sold by Willow Garage, the original creators of ROS, it already came with very good ROS support. In the course of this thesis, the author was responsible for the full ROS integration of the Calvin and Kurtana robots and a large part of the integration of the RACE software with the PR2 robot and Gazebo simulator.

### 2.6.1 The PR2 Robot and Gazebo Simulator

The PR2 (Figure 2.4) is a mobile manipulator robot that has an omnidirectional base, a liftable torso, a pan-tilt head and two 7DoF arms with parallel grippers. The sensor suite consists of a 2D Hokuyo laser scanner in the base, a 3D laser scanner (consisting of a Hokuyo 2D laser scanner on a tilt unit), and three 3D camera systems in the head (a narrow-angle stereo pair, a wide-angle stereo pair and a custom structured light camera). The University of Hamburg's PR2 was also fitted with an ASUS Xtion Pro Live RGB-D camera, which was used instead of the built-in 3D cameras in the RACE project. The PR2 was used in the experiments in Chapters 4 and 5.

The RACE demonstration environment (Section 2.4), including the PR2 robot, the furniture and all task-relevant objects, was modeled in the Gazebo simulator (Figure 2.5). This eased the software development at RACE partners without easy access to the physical PR2 robot. It also made it possible to perform a large number of experiments in a reproducible way. The Gazebo simulator was used in the experiments in Chapter 5.

### 2.6.2 The Calvin Robot

Osnabrück University's Calvin robot is equipped with an ASUS Xtion Pro Live RGB-D camera (top) that was removed from its casing, a web camera below (which was not used in the experiments) and a pair of laser scanners for localization, navigation and obstacle avoidance (see Figure 2.6). It has a differential drive base and a Katana 5DoF arm with an angular gripper. Calvin was used in the experiments in Chapters 4 and 5.

### 2.6.3 The Kurtana Robot

Osnabrück University's Kurtana robot (Figure 2.7) is a differential drive Kurt robot with a Microsoft Kinect mounted on a pole. It was used in the experiments in Chapter 3.


**Figure 2.4:** University of Hamburg's PR2 robot "Trixi" during one of the demonstration scenarios. Figure reproduced from RACE Deliverable D5.3 (2013).

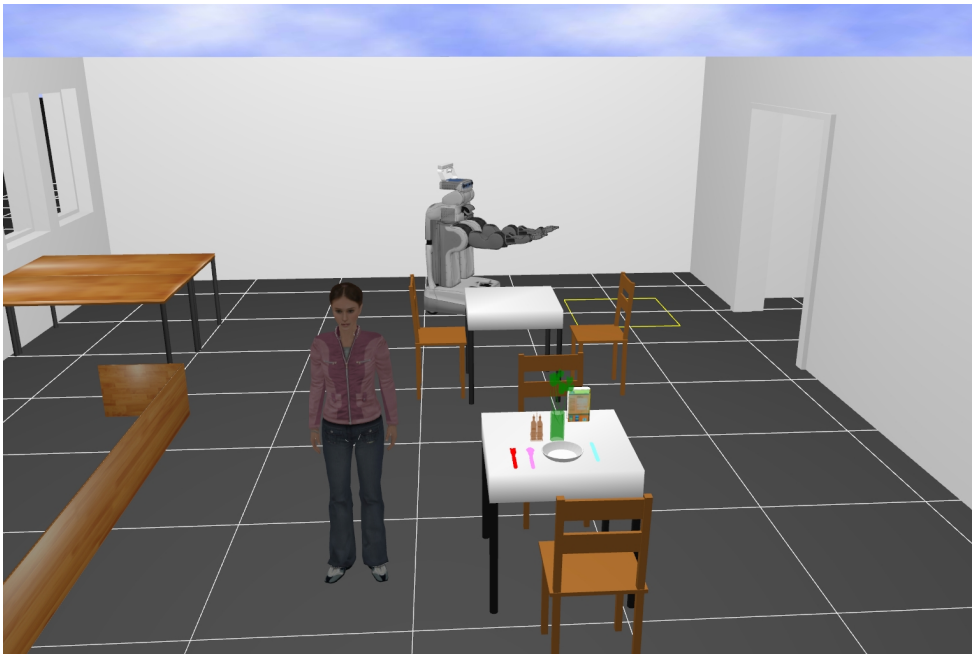


**Figure 2.5:** The demonstration environment in the Gazebo simulator. Figure reproduced from RACE Deliverable D5.3 (2013).


**Figure 2.6:** Osnabrück University's Calvin robot

**Figure 2.7:** Osnabrück University's Kurtana robot. Figure reproduced from Günther et al. (2013).

# Chapter 3

# Model-Based Semantic Mapping

Many robot applications require a *semantic* map of the environment, i.e., a map that not only captures the geometric properties of the environment but which is also annotated with semantic labels, such as the location and type of doors, elevators, furniture, workbenches, machinery or other task-relevant objects. For example, in the RACE project, the locations of the chairs, tables, and counter played a crucial role in the robot's activities (such as clearing a table or bringing a mug of coffee from the counter to a specific area on a given table). As explained in Section 2.4, in RACE, these areas were specified manually, since the topic of mapping was outside the scope of the project. However, a robot that adapts autonomously to new environments and changes to a known environment needs the capability to create such a semantic map autonomously. This chapter presents such a semantic mapping system with the ability to recognize larger-scale, quasi-static objects like furniture. Smaller, more dynamic objects like table-top objects are treated in Chapter 4.

In recent years, a great number of approaches that generate geometric maps from RGB-D data have been developed. These either register the continuous RGB-D frames into a consistent full-scene point cloud (Dryanovski et al., 2013; Endres et al., 2012; Henry et al., 2012; Kerl et al., 2015; Labbé and Michaud, 2018) or a triangle mesh (Newcombe et al., 2011). However, creating a *semantic* map of the environment is still an open research issue. Labeling environment features with meaningful semantic information is required for a diverse range of tasks, such as task planning, human-robot interaction, place recognition and object search. Moreover, recognizing semantic information in the mapped environment can be beneficial for the map-

The contributions described in this chapter have first been published in the following publications: Albrecht et al. (2011); Günther et al. (2011, 2013, 2017)

ping process itself by aiding loop closure or generating hypotheses about occluded parts of the environment.

According to the seminal paper by Kuipers and Byun (1991), a semantic map is *hybrid*: It contains geometric and high-level, qualitative semantic features (Nüchter and Hertzberg, 2008). In our system, the geometric map is a 3D triangle mesh of the environment, enriched with semantic information about the pieces of furniture in the form of CAD models with their 6DoF poses. The system works in a closed loop, i.e., semantic information is not only extracted from the sensor data (bottom-up), but knowledge and reasoning are also used in the object classification (top-down). The system is also capable of building its map incrementally, which has the advantage that one does not have to wait for the completion of the mapping process before obtaining initial results. This opens up the possibility of influencing the mapping process based on the information perceived so far.

The semantic mapping system presented in this chapter consists of the following three components:

**Surface reconstruction:** Based on the incoming RGB-D data, a 3D triangle mesh of the environment is constructed, and planar patches are extracted.

**Ontology-based object classification:** An ontology reasoner classifies the planar patches and generates object hypotheses.

**CAD model matching:** The object hypotheses are verified by matching a CAD model of the object to the underlying point cloud data.

The system presented in this chapter has certain commonalities with the context-aware anchoring system presented in Chapter 4: Both systems recognize objects in the environment based on RGB-D data and store their poses relative to a global reference frame. However, the objects that are handled are fundamentally different regarding their size, appearance and dynamics, which is why they are best tackled by different approaches. Furniture objects (which are dealt with in this chapter) are relatively large-scale, quasi-static objects that often feature dominant planes, whereas tabletop objects (Chapter 4) in comparison are small-scale, dynamic objects which mostly lack dominant planes and are instead often characterized by more complex shapes and textures. This leads to the following differences in the approaches:

**Size:** The large scale of furniture objects often causes the piece of furniture to extend beyond a 3D camera's sensor aperture. For this reason, the system presented in this chapter can not

only run in single-frame mode (when high interactivity is required), but can also register multiple point clouds into a consistent scene, where the full pieces of furniture are visible.

**Appearance:** The classification approach used by the system from this chapter is based on dominant planes, which is appropriate for most pieces of furniture. Also, furniture usually occurs in an "upright" orientation. To properly recognize various tabletop objects in all orientations, the system from Chapter 4 integrates an arbitrary 3D object recognition method that takes the full texture and shape of the objects into account.

**Dynamics:** Many pieces of furniture are quasi-static and are rarely moved around. For this reason, it makes sense to include them in a semantic map of the environment that is updated in larger intervals. On the other hand, tabletop objects are moved often, which is why they need to be tracked and continually re-anchored.

For these reasons, the two systems are complementary: The system presented in this chapter would typically be used in a first pass to create a semantic map, i.e., a hybrid map consisting of a geometric mesh-based 3D map overlaid with semantic information about the pieces of furniture present. The system from Chapter 4 is then run continually during task execution to anchor the tabletop objects with reference to the semantic map created earlier.

## 3.1 State of the Art

The field of semantic mapping has attracted continuing interest in the past decades (see the survey by Kostavelis and Gasteratos (2015) for an overview). Here, we focus on large-scale approaches that label point clouds with semantic information in an indoor environment, especially those which explicitly represent knowledge using some form of KR&R system.

Galindo et al. (2005) use a semantic map consisting of two separate hierarchies, spatial and semantic (using a DL ontology). The link between them is maintained through anchoring. They demonstrate that their approach can be used to communicate with humans symbolically and disambiguate incomplete information using a planner. Nüchter and Hertzberg (2008) combine a semantic network for classifying large-scale building structures such as walls, floor, ceiling and doors with a trained classifier for recognizing smaller objects. Rusu et al. (2008) recognize kitchen furniture objects in a point cloud and approximate them as cuboids, then employ a hierarchical object model to distinguish the furniture class based on smaller-scale features such as knobs and handles. Pangercic et al. (2012) detect furniture parts in point clouds and feed them

into an ontological knowledge base. In contrast to the system presented here, their approach works only in a bottom-up fashion; information from the knowledge base is not used in the recognition loop. Mason and Marthi (2012) build and update a semantic map over a long time span by autonomously navigating a robot repeatedly through the same environment over the course of several weeks. Their approach focuses on smaller-scale objects (not furniture) and does not employ an explicit knowledge representation system. In the same vein, Ambrus et al. (2014) build an environment model from repeated observations over a long time span, and use the difference between observations to extract both static environment features (rooms) and dynamic ones (objects), however without assigning them a semantic label. Alonso-Ramirez et al. (2018) present a system that recognizes furniture and builds an environment map that includes the furniture instances. Like our system, each piece of furniture is characterized by its constituent parts (like horizontal and vertical planes) and their features (like height and area). In contrast to our system, they use a relational graph instead of a knowledge representation and reasoning system to generate object hypotheses.

With regards to CAD-model-based object recognition, early approaches already appeared in the late nineties (Brenner et al., 1998; Majumdar and Seethalakshmy, 1997), but could only recognize objects at an approximately known position. In contrast, our approach employs general domain knowledge to generate object hypotheses before refining them using CAD models. In this regard, our approach is more similar to those by Klank et al. (2009); Mozos et al. (2011); Usenko et al. (2012); Wohlkinger et al. (2012). Klank et al. (2009) performs CAD matching on 2D image data, whereas we use 3D point clouds. Mozos et al. (2011) and Usenko et al. (2012) employ probabilistic Hough voting to generate object hypotheses and apply RANSAC for pose refinement and hypothesis verification. Wohlkinger et al. (2012) do not use the CAD model directly; instead, the RGB-D image is compared to synthetically rendered views of the CAD model to determine the best match.

The SLAM++ system (Salas-Moreno et al., 2013) performs real-time recognition of furniture in RGB-D data and embeds this process into a SLAM system, demonstrating the point made earlier that semantic interpretation of the data can be beneficial for the mapping process itself. More recently, McCormac et al. (2018) presented the Fusion++ system that finds object instances (including smaller-scale furniture) in RGB-D data using the Mask R-CNN classifier, then performs surface reconstruction on each object (with incremental refinement) and builds a persistent 6DoF pose graph, which is optimized using a SLAM approach.

In recent years, a number of *dense 3D semantic mapping* approaches (Hermans et al., 2014; Ma et al., 2017; McCormac et al., 2017; Stückler et al., 2015) have been developed, which incre-

mentally register RGB-D frames into an internal (usually point-cloud-based) representation, where the points of the point cloud are densely labeled with semantic categories. Since these approaches are methodically closer to the context-aware anchoring system (Chapter 4), they will be reviewed in Section 4.1.3.

## 3.2 Algorithm and Implementation

In recent years, CAD models of many kinds of objects have become widely available. One resource of CAD models is Google's 3D Warehouse, which allows querying and retrieving CAD models of virtually any kind of object via the web. In the domain of furniture recognition, CAD models are often available directly from the manufacturer or from companies specialized in creating CAD models for interior designers. We use a database of CAD models supplied by our university's furniture manufacturer.

In this work, we focus on the domain of furniture recognition for several reasons: First, due to the widespread use of CAD models in interior design, the availability of CAD models in this domain is especially strong. Second, most kinds of furniture feature a set of planar surfaces which can be robustly recognized in 3D point clouds. Third, due to the rigidness of furniture, these planar surfaces are in a clearly defined relation to each other.

Figure 3.1 shows the embedding of our system in a general semantic mapping framework. We see our model-based object recognition method as complementary to appearance-based methods based on 2D image features or 3D shape features.

Using the information contained in CAD models for object recognition has several advantages. Instead of having one classifier for each kind of object, only the geometric primitives have to be detected. Based on these, objects are reconstructed. Also, no classifier training and no labeled training sets are required; to add a new object class, only the CAD model is required. In the future, it would even be conceivable that such a CAD model could be retrieved on-line from the web. Another advantage is that once an object is recognized, the corresponding part of the sensor data can be replaced by the CAD model, thus filling up occlusions in the sensor data.

On the other hand, appearance-based methods have an advantage where the object that is to be recognized is non-rigid, does not consist of clearly identifiable geometric primitives of a certain minimum size or where labeled training data, but no CAD model is available.

We see our model-based object recognition method as an instance of a more general system architecture (Fig. 3.1), consisting of three steps: (1) surface reconstruction (Sec. 3.2.1), as an in-

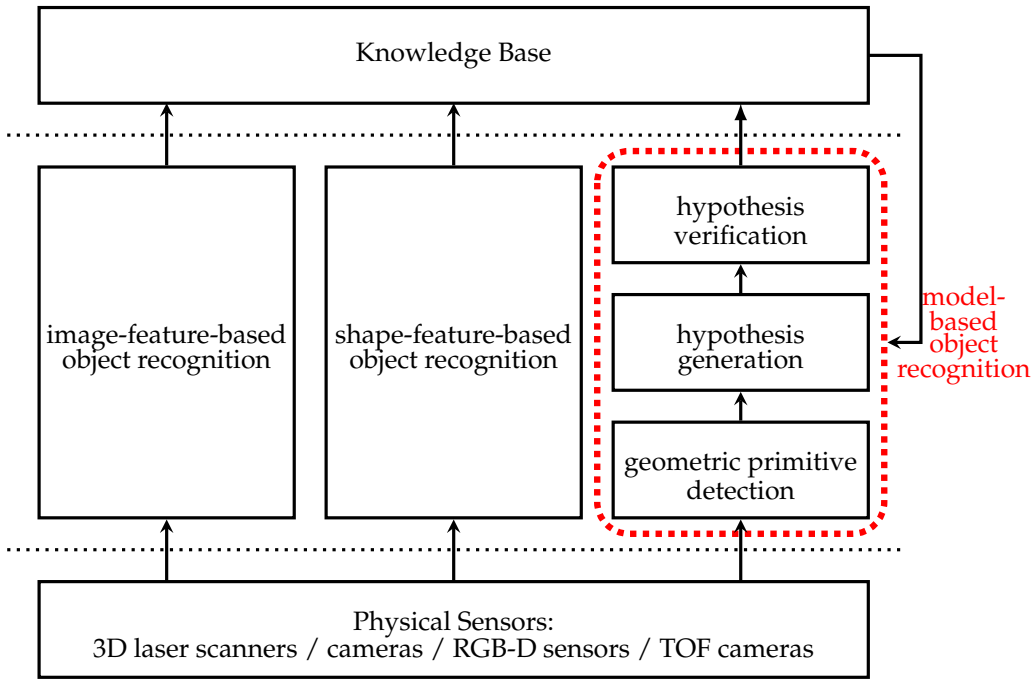



**Figure 3.1:** System overview. While the system presented here is focused on model-based object recognition, we consider this method as yielding complementary information to standard recognition methods. So in a more general system architecture, they may well co-exist. Figure reproduced from Günther et al. (2011).

stance of geometric primitive detection, transforms the input point cloud into a triangle mesh and extracts planar regions; (2) planar region classification (Sec. 3.2.2), as an instance of hypothesis generation, classifies the planar regions, detects furniture objects, and calculates initial pose estimates based on the planar regions; (3) final pose adjustment (Sec. 3.2.3), as an instance of hypothesis verification, computes the final pose using ICP, and places the corresponding CAD model in the scene.

### 3.2.1 Surface Reconstruction

Surface Reconstruction is our implementation of the geometric primitive detection step in Fig. 3.1[1]. The input to Surface Reconstruction is a point cloud (either a single point cloud captured by the robot's Kinect RGB-D camera, or a fully registered point cloud of the whole scene). The point cloud is first transformed into a mesh using the Marching Cubes implementation of the Las Vegas Surface Reconstruction Toolkit, LVR (Wiemann et al., 2012). Next,

[1]The part of the system described in Section 3.2.1 was developed by Thomas Wiemann and is only shortly outlined in order to describe the part it plays in the overall system. Details can be found in Günther et al. (2017).

the initial mesh is run through LVR's post-processing pipeline of several mesh optimization and segmentation algorithms. Afterward, planes in the mesh are segmented using a region-growing approach. The output of the planar segmentation algorithm is a set of planar regions represented by contour polygons along with their estimated size and pose.

### 3.2.2 Planar Region Classification

Planar region classification is our implementation of the hypothesis generation step in Figure 3.1. Once all planar regions have been extracted in the previous step, those regions corresponding to pieces of furniture have to be classified. Here, we make use of the fact that most pieces of furniture are comprised of planar structures that have a certain size, orientation, height above ground and spatial relation to each other. These features (and combinations of features) are expressed in an OWL-DL ontology in combination with SWRL rules.

The Web Ontology Language (OWL) is the standard proposed by the W3C consortium as the knowledge representation formalism for the Semantic Web. One of its three sub-languages, OWL-DL, corresponds to a Description Logic (Baader et al., 2003; Dean et al., 2004), a subset of First-Order Logic that provides many expressive language features while guaranteeing decidability. It has been extended by SWRL, the Semantic Web Rule Language (Horrocks et al., 2004), which allows writing Horn-like rules in combination with an OWL-DL knowledge base and includes so-called built-ins for arithmetic comparisons and calculations. We decided to use OWL-DL as the knowledge representation format for this work for several reasons: OWL-DL ontologies can be easily re-used and linked with other sources of domain knowledge from the Semantic Web, they easily scale to arbitrarily large knowledge bases, and fast reasoning support is available. In our implementation, we use the open-source OWL-DL reasoner Pellet (Sirin et al., 2007), which provides full support for OWL-DL ontologies using SWRL rules.

The class hierarchy of the ontology we use here is shown in Figure 3.2. The basic classes are Furniture (the parent class of all recognized furniture objects) and Plane (the planar regions of which Furniture objects are comprised). A set of SWRL rules is applied to the extracted planar regions to assign them classes in the Plane sub-hierarchy; for example, the lower plane of a shelf can be characterized by the following SWRL rule:

```
LowShelfPlane(?p) ← HorizontalPlane(?p)
   ∧ hasSize(?p, ?s) ∧ swrlb:greaterThan(?s, 0.01) ∧ swrlb:lessThan(?s, 0.5)
   ∧ hasPosY(?p, ?h) ∧ swrlb:greaterThan(?h, 0.08) ∧ swrlb:lessThan(?h, 0.18)
```

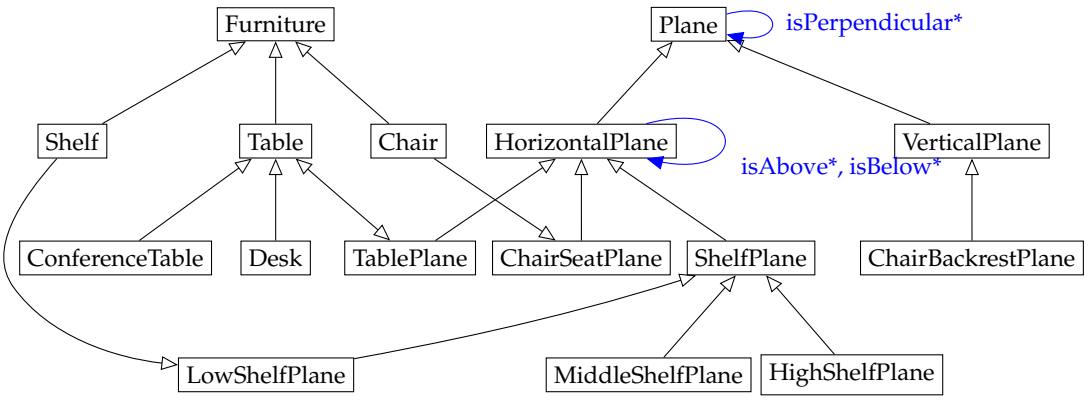



**Figure 3.2:** Parts of the OWL-DL ontology used for classification: classes (□), properties (—▶) and is-a relationships (—▷). Figure reproduced from Günther et al. (2013), adapted.

These rules are not exclusive, so one planar region can receive multiple labels (e.g., MiddleShelfPlane and ChairSeatPlane). The definitions of the classes in the Furniture sub-hierarchy refer to these labels; e.g., the fact that a Shelf consists of three planes on top of each other can be stated as:

```
Shelf ≡ LowShelfPlane
        ∧ (isBelow some (MiddleShelfPlane ∧ (isBelow some HighShelfPlane)))
```

Likewise, chairs are defined by a seat and a backrest, both with certain sizes, orientations, heights above ground and which are perpendicular to each other. In this work, these rules (which encode the structural model of the object) were constructed manually. The parameters can be measured directly from the CAD model – e.g., in the example above, the lower shelf plane has a height above ground of 0.13 m; adding a margin of 0.05 m to compensate for errors in the estimated height, one arrives at the specified range (0.08 m; 0.18 m). We hope to automate this process in future work.

The classification of planar patches into furniture objects is performed by the Pellet reasoner. Initially, each planar patch that was extracted during surface reconstruction, along with its geometric features (size, position, orientation, bounding box) and relations to other planar patches is added as an individual to the ontology's ABox. Next, the reasoner jointly classifies all planar patches, using the SWRL rules and class definitions outlined above.

For each detected object, the initial position and orientation are estimated. The position is always the centroid of their main plane. Since chairs have two perpendicular planes (the backrest and the seat), a unique orientation can be calculated by the vertical component of

the difference vector between the centroids of those planes (assuming an upright position of the chair). For tables and shelves, the PCA of the points corresponding to their main plane is calculated to define the orientation. Note that the PCA of a rotationally symmetric object, such as a round table, is not well-defined. This does not impact the recognition rate of our system, only the initial orientation estimate; if the object has distinct non-planar features (such as table legs), this orientation error can be corrected by the next processing step.

### 3.2.3 Final Pose Adjustment and Model Replacement

Final Pose Adjustment is our implementation of the hypothesis verification step in Fig. 3.1[2]. The input to Final Pose Adjustment is the initial pose estimate of an object from the previous step along with the original point cloud. Additionally, a reference point cloud of the object is required. This reference point cloud is computed by sampling a CAD model of the object. To refine the object pose estimate, the reference point cloud placed at the initial pose estimate is matched to the original point cloud using the ICP algorithm (Besl and McKay, 1992).

## 3.3 Results

The experiments related to the work by Thomas Wiemann and Sven Albrecht (Sections 3.2.1 and 3.2.3) have been omitted here and can be found in Günther et al. (2017). Only the results related to the complete system are presented here.

### 3.3.1 Evaluation of the Complete System

To evaluate our recognition system, we captured two series of point clouds from a Kinect camera mounted on a mobile robot (the "Kurtana" robot, see Figure 2.7 on page 24). In the first scenario, the robot was teleoperated around an office while continuously capturing point cloud data at 2.2 Hz, resulting in a total of 431 point clouds. The office contained 13 recognizable objects from 5 classes (1 desk, 1 conference table, 1 office chair, 5 conference chairs and 5 bookshelves). For the second dataset, the robot was driven through a seminar room, capturing a total of 379 point clouds. The objects present in this dataset are 12 seminar tables and 20 chairs. One challenging aspect of this dataset is that there is a high level of occlusion. Both datasets are available at `http://kos.informatik.uni-osnabrueck.de/furniture_recognition/`.

[2]The part of the system described in Section 3.2.1 was developed by Sven Albrecht and is only shortly outlined in order to describe the part it plays in the overall system. Details can be found in Günther et al. (2017).

For both datasets, we registered the point clouds into a consistent full-scene point cloud, using SLAM6D from 3DTK (Nüchter et al., 2007). The full-scene point clouds were used to generate the ground truth poses for each piece of furniture by hand. These poses are used to evaluate the results of our classification and the ICP refinement step.

Ground truth data for each frame was generated by manually labeling each frame with the information which of the objects occur in that frame. The ground truth poses of each object were estimated by manually placing each CAD model into a scene consisting of the fully registered datasets. In combination with the camera trajectory obtained from the registration process, the ground truth object poses in each frame can be computed. A detection is considered "true positive" if its distance to the nearest true object pose of the same class was below a threshold, depending on the CAD model's size (15 cm for chairs, 25 cm for shelves, 45 cm for tables). If multiple detections fell into that range, only the nearest was counted as a true positive, and the others as false positives.

These high thresholds were chosen based on experience and reflect the diminishing depth resolution of the Kinect sensor (at 5 m the depth resolution is approx. 5 cm – even without noise depth measurements can differ substantially from the real distance) accompanied by additional random noise, as well as the size of the discriminating planar surfaces of the object classes. Furthermore, if all individual point clouds are registered perfectly, there will still be noisy "shadow points" around each object. Our ground-truth poses are located in the middle of the noisy point cloud resembling the objects in question.

Note that the recognition system itself does not require prior registration; it can work both directly on unregistered single-frame point clouds or a registered full-scene point cloud. Both approaches have their advantages; directly processing each frame eliminates the computational cost for registration, removes the risk of registration failures, and avoids artifacts arising from combining many point clouds from a noisy sensor. Single frame processing is therefore better-suited for online operation in an incremental semantic mapping framework. On the other hand, the narrow field of view and occlusions – that may be recovered by viewing an object from different angles – make it more likely that a piece of furniture is not fully visible.

The detection results both for single frames and the full scene on both datasets are shown in Table 3.1. In addition to the detection results, the translation and rotation error of the initial guess (based on PCA for tables and shelves, and based on the vector from backrest to seat for chairs) and the translation and rotation error after ICP pose correction are shown. Figure 3.3 on page 39 depicts some exemplary object detections, Figure 3.4 on page 40 shows the final results of the system running in full-scene mode.

**Table 3.1:** Detection rates of furniture recognition. Table reproduced from Günther et al. (2017).

| | true pos. | false pos. | false neg. | initial transl. error | initial rot. error | final transl. error | final rot. error | precis. | recall | $F_1$ score |
|---|---|---|---|---|---|---|---|---|---|---|
| **(a) office dataset (single point clouds):** | | | | | | | | | | |
| **Shelf** | 82 | 53 | 237 | 16.2 cm | 5.1° | 60.1 cm | 27.2° | 60.7 % | 25.7 % | 36.1 % |
| **OfficeChair** | 8 | 19 | 6 | 5.8 cm | 61.0° | 6.0 cm | 10.8° | 29.6 % | 57.1 % | 39.0 % |
| **ConfChair** | 100 | 190 | 114 | 9.0 cm | 51.9° | 11.0 cm | 56.8° | 34.5 % | 46.7 % | 39.7 % |
| **Desk** | 10 | 11 | 29 | 31.6 cm | 125.6° | 53.7 cm | 118.8° | 47.6 % | 25.6 % | 33.3 % |
| **ConfTable** | 81 | 0 | 0 | 8.5 cm | 8.8° | 9.4 cm | 6.2° | 100.0 % | 100.0 % | 100.0 % |
| **total** | 281 | 273 | 386 | 11.7 cm | 28.7° | 26.2 cm | 34.5° | 50.7 % | 42.1 % | 46.0 % |
| **(b) office dataset (full registered scene):** | | | | | | | | | | |
| **Shelf** | 1 | 0 | 4 | 24.7 cm | 1.0° | 131.6 cm | 4.1° | 100.0 % | 20.0 % | 33.3 % |
| **OfficeChair** | 1 | 0 | 0 | 5.2 cm | 68.1° | 4.7 cm | 14.2° | 100.0 % | 100.0 % | 100.0 % |
| **ConfChair** | 3 | 2 | 2 | 10.9 cm | 55.9° | 10.1 cm | 49.1° | 60.0 % | 60.0 % | 60.0 % |
| **Desk** | 1 | 0 | 0 | 41.8 cm | 21.2° | 12.8 cm | 6.7° | 100.0 % | 100.0 % | 100.0 % |
| **ConfTable** | 1 | 0 | 0 | 2.5 cm | 4.3° | 6.3 cm | 0.6° | 100.0 % | 100.0 % | 100.0 % |
| **total** | 7 | 2 | 6 | 15.3 cm | 37.5° | 26.5 cm | 24.7° | 77.8 % | 53.8 % | 63.6 % |
| **(c) seminar room dataset (single point clouds):** | | | | | | | | | | |
| **Chair** | 358 | 26 | 257 | 7.3 cm | 20.3° | 8.3 cm | 10.0° | 93.2 % | 58.2 % | 71.7 % |
| **SemTable** | 522 | 153 | 21 | 9.3 cm | 2.5° | 9.2 cm | 2.4° | 77.3 % | 96.1 % | 85.7 % |
| **total** | 880 | 179 | 278 | 8.5 cm | 9.7° | 8.8 cm | 5.5° | 83.1 % | 76.0 % | 79.4 % |
| **(d) seminar room dataset (full registered scene):** | | | | | | | | | | |
| **Chair** | 6 | 2 | 14 | 5.7 cm | 16.3° | 6.0 cm | 9.8° | 75.0 % | 30.0 % | 42.9 % |
| **SeminarTable** | 11 | 0 | 1 | 4.2 cm | 1.2° | 3.3 cm | 1.2° | 100.0 % | 91.7 % | 95.7 % |
| **total** | 17 | 4 | 15 | 4.7 cm | 6.537° | 4.3 cm | 4.252° | 81.0 % | 53.1 % | 64.2 % |

For the single point clouds, our approach achieves detection rates of 46.0 % and 79.4 % on the two data sets. We expect that these results can be improved significantly in the future by integrating information over several frames instead of treating each frame independently. The results show that our approach is not only robust enough to deal with the noise present in low-cost 3D sensors but also copes with occlusion and partial visibility, typical for sensors with a small opening angle. Unexpectedly, the final ICP pose correction did not improve the average initial guess on single frames for the first dataset. We attribute this to the fact that we matched a complete CAD model to a partially occluded object view; possible solutions are outlined in the next section. In the second dataset, however, ICP clearly improved the rotation of the chairs compared to the initial guess from the SWRL rules.

The results also show varying detection success for the different classes. Shelves have one of the lowest detection rates and highest final pose errors in our experiments. This is not surprising: All shelves in our data set were completely filled with books, so only a small portion of the actual shelf was visible. This also explains why the final pose correction performed worse on shelves compared to the other object classes. In addition, we currently do not handle aggregates of objects: If two shelf segments or two tables are positioned with no gap between them, the planar classification will combine them into one potential object, with the possible location at the center of the combined plane. This usually leads to one false positive (for the non-existent object at the center location) and several false negatives for actual objects creating the aggregated plane. Another problematic object is the big L-shaped desk: The desk is so big that only a small part of it is visible in most single frames.

A comparison between single-frame and full-scene mode reveals that both approaches have their complementary strengths and weaknesses. Both classification accuracy and final pose error for most objects (especially big ones, like tables) are better in full-scene mode since there are fewer problems with partial visibility due to occlusion or limited aperture. On the other hand, the detection rate for chairs in the seminar table dataset is higher in single-frame mode. The main reason for this seems to be that since chairs are relatively small compared to tables, limited aperture doesn't pose as much of a problem in single-frame mode. On the other hand, since the chairs were relatively close together, registration errors and accumulated sensor noise in full-scene mode often lead to two chairs being recognized as one object, preventing detection.

**Table 3.2:** Run times of the three steps in our processing pipeline on the seminar room data set. The results for single point clouds were averaged over all 379 point clouds. Table reproduced from Günther et al. (2017).

| | **single point cloud** | **full registered scene** |
|---|---|---|
| **Surface Reconstruction** | 6.53 s | 117.06 s |
| **Planar Region Classification** | 1.02 s | 2.67 s |
| **Final Pose Adjustment** | 1.53 s | 26.11 s |
| **Total** | 9.09 s | 145.84 s |

### 3.3.2 Runtime of the System

The runtime performance of our system and its components are shown in Table 3.2. All experiments have been performed on a standard laptop (2.6 GHz Core i7 CPU, 8 GB RAM). The single point clouds are processed at full resolution (245,304 points on average), while the full scene was downsampled to about one-tenth of all points (9,475,220 points).

Most of the processing time is spent on the surface reconstruction step. This could in principle be replaced by faster but less accurate methods. For the RGB-D data used here, Kinect Fusion is a good candidate, although the meshes have to be post-processed to get a topologically correct mesh representation (Wiemann et al., 2013), which is required for region growing. For our experiments, we decided to use the Marching Cubes implementation from our LVR library, since it computes meshes that are accurate and topologically sound. The classification and pose adjustment steps are fast. The whole scene consisting of 379 point clouds, i.e., about 9.5 million points, was processed in about 2.5 Minutes.

## 3.4 Discussion and Future Work

We have presented a semantic mapping system that creates a triangle mesh of an office environment, detects several classes of furniture, and replaces them with their corresponding CAD models, based on Kinect point cloud data captured using a mobile robot. The system was evaluated both on single frames and fully registered scenes of two datasets containing 810 single point clouds. Our system achieved a detection rate of 46.0 % for the office dataset, containing one particularly large object and several seamlessly connected instances of shelves and 79.4 % on the seminar room with less variation of the classified furniture.

The current system creates a *hybrid* semantic map (i.e., the map data contains geometric

information as well as semantic knowledge); it does so *incrementally* by processing each RGB-D frame separately; and it also does so in *closed loop* by feeding the output of the reasoning system back into the low-level data interpretation routine. In future work, we intend to implement an active data interpretation / gathering loop that identifies and actively explores regions with potentially valuable but missing data, thereby closing the loop even down to the action level, not just the perception level.

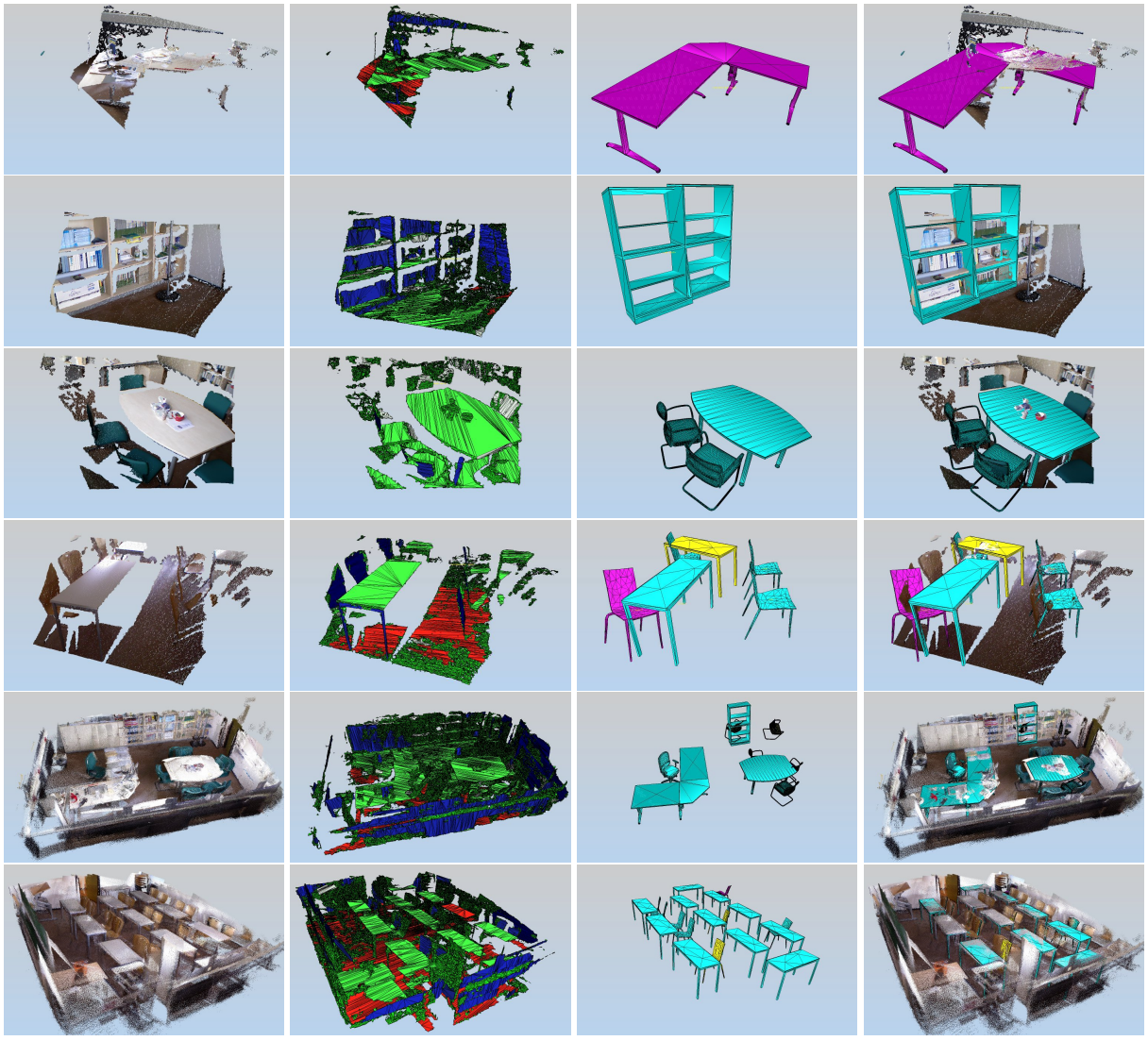


**Figure 3.3:** Results of each step in the processing pipeline. Columns, from left to right: (a) the original point cloud from a single Kinect frame; (b) the reconstructed triangle mesh; (c) sampled CAD models after ICP pose correction; (d) the point cloud overlaid with the recognized CAD models. Row 1: error in pose estimation due to only partial visibility of the desk; Row 2-3: correct placement of the recognized objects; Row 4: several good matches and one false positive. The color of the models indicates the quality of the recognition: cyan indicates a true positive, where the pose fits well with the actual pose, magenta is a true positive where the final pose is not well aligned and yellow shows a false positive. Rows 1-3 show data from single frames of the office dataset, row 4 from the seminar room dataset, while row 5 and 6 show the full scene of the office and seminar room datasets, respectively. Figure reproduced from Günther et al. (2017).

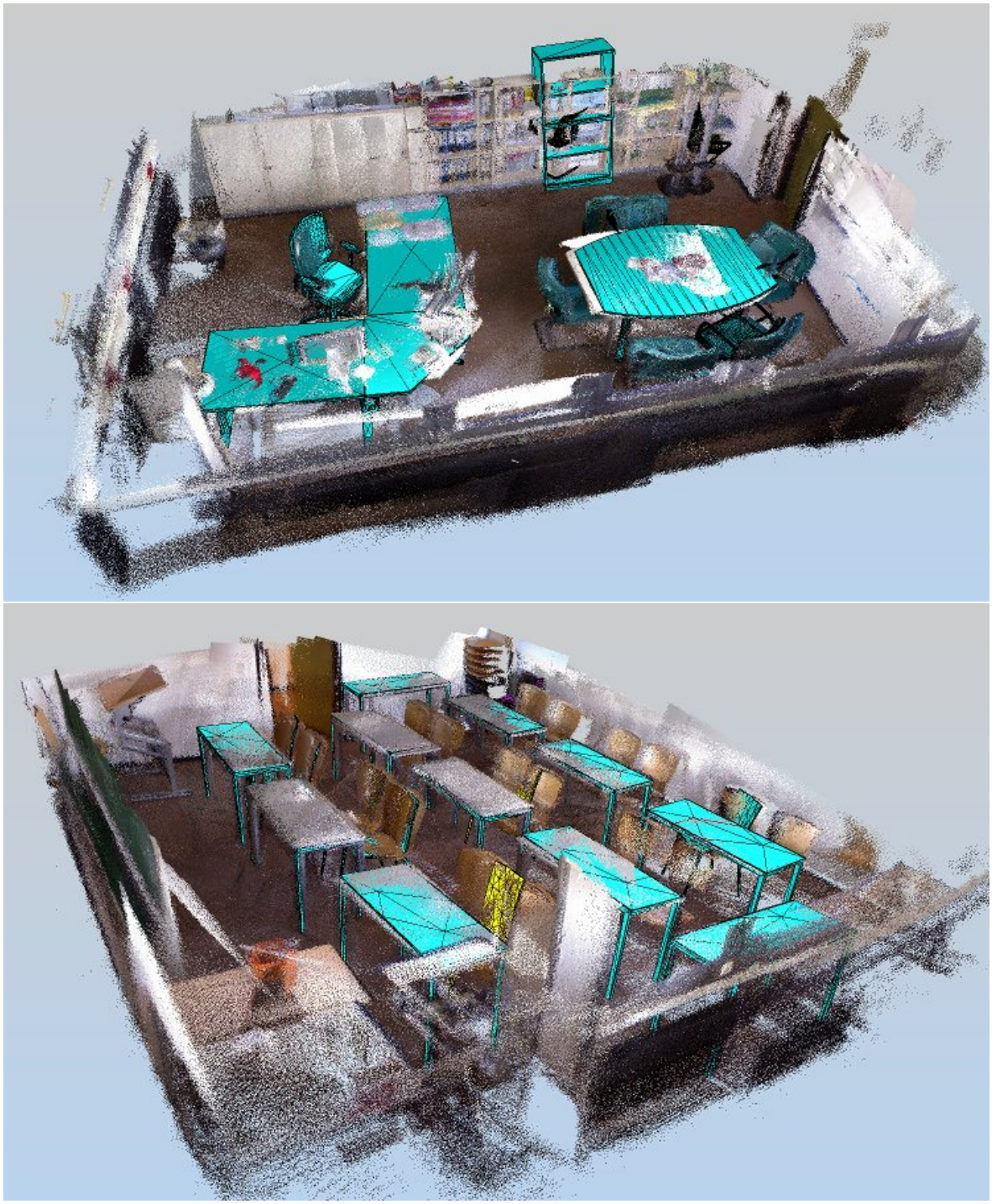


**Figure 3.4:** Final results of our system in full-scene mode (registered full-scene point clouds overlaid with the recognized furniture models). Top: Office dataset (431 frames). Bottom: seminar room dataset (379 frames). Note that point color information is displayed for reader convenience, our approach does not require it. Figure reproduced from Günther et al. (2017).

# Chapter 4

# Context-Aware Anchoring

Any system for plan-based robot control must necessarily create and maintain a correspondence between the symbols used by the planner to denote objects and the object percepts detected in the robot's sensor stream. This is known as the *anchoring problem*:

**Definition 1.** "We call *anchoring* the process of creating and maintaining the correspondence between symbols and sensor data that refer to the same physical objects. The *anchoring problem* is the problem of how to perform anchoring in an artificial system." (Coradeschi and Saffiotti, 2003, p. 86f).

This chapter presents an anchoring module that anchors the symbols used by the high-level robot control modules, such as the hierarchical hybrid planner CHIMP (Stock, 2017a,b), to the output of a local object recognition method. A central feature of our anchoring module is that it is *context-aware*: It improves the output of the local object recognition method by taking the context of an object (in the sense of geometrical relations to other objects) into account.

Figure 4.1 shows the role of the anchoring module within the RACE architecture (see Chapter 2). All knowledge about physical objects is exchanged between modules via object fluents on the blackboard. Some of these fluents are created by external knowledge sources (for example, a module that initializes the blackboard with the locations of known objects, or a module that gathers knowledge about object locations from dialogue with a human). They are read by the planner, which uses this information to create the initial state of the planning problem. They are both read, updated and created by the execution monitor (for example, when the robot

Parts of the contributions described in this chapter have first been published in the following publication: Ruiz-Sarmiento et al. (2017c). A revised version of this chapter has later been published in Günther et al. (2018).

picks up a mug from the table, the execution monitor ends the fluent that specifies that the mug is on the table and adds one that specifies that it is now being held by the robot gripper). The anchoring module's role is to anchor the object fluents on the blackboard to percepts of individual objects produced by the object recognition system and update the blackboard accordingly. This involves updating fluents (when an object has moved or a previously untracked anchor has been reacquired), ending fluents (when an object is found to no longer be in the location stored on the blackboard), and creating new fluents (when new objects not yet represented on the blackboard are detected).

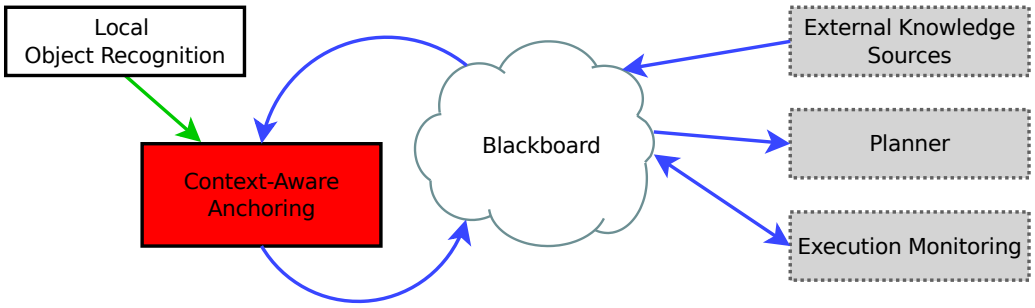



**Figure 4.1:** The role of the context-aware anchoring system within the RACE architecture. Blue arrows: fluents related to physical objects; green arrow: object recognition results.

A distinctive feature of our anchoring system is that it does not simply copy the classification results from the object recognition system, but instead uses the relations between objects (their spatial *context*) to improve those classification results. This is motivated by the fact that objects rarely occur in independent configurations at identically distributed locations. Rather, there is a coherent structure inherent to most real-world scenes. For example, a longish object in front of a monitor has a high probability of being a keyboard, whereas an object with the same local appearance next to a bread knife is more likely a cutting board.

In cases where local appearance features are not sufficient, contextual features can disambiguate object appearance in object recognition tasks (Galleguillos and Belongie, 2010). Jointly considering context-based object categorization and anchoring as parts of a *context-aware anchoring system* benefits both sub-problems: Anchoring receives better and more stable object classification results, while context-based object categorization can use contextual relations with anchored objects that extend beyond the sensor aperture.

As an example of this, consider Figure 4.2. Due to the position of the robot and the aperture of the RGB-D camera, only part of the table scene is visible in the current sensor frame, so the full context is not available. This poses a problem for most existing context-aware object recog-

nition systems, which fall roughly into one of the following two categories. The first category is single-frame recognition systems, which recognize objects relying on single observations of the scene in the form of RGB, depth or RGB-D images (Galleguillos et al., 2008; Hoiem et al., 2008; Ren et al., 2012; Shotton et al., 2009; Torralba et al., 2003; Xiang et al., 2010). Regarding the exploitation of contextual information, single-frame systems are seriously limited by the sensor aperture and occlusions, given that they can only observe a portion of the objects and relations appearing in the whole scene. The second category is offline recognition systems, which register several observations before the recognition process to obtain a wider view of the scene (Alberti et al., 2014; Anand et al., 2013; Koppula et al., 2011; Kunze et al., 2014a; Ruiz-Sarmiento et al., 2014, 2015a; Thippur et al., 2015; Valentin et al., 2013; Xiong and Huber, 2010). This solves the problems caused by sensor aperture and occlusions; however, the need to finish recording the sensor data before running the object recognition process prevents online operation, which is a requirement for most plan-based robot control systems.

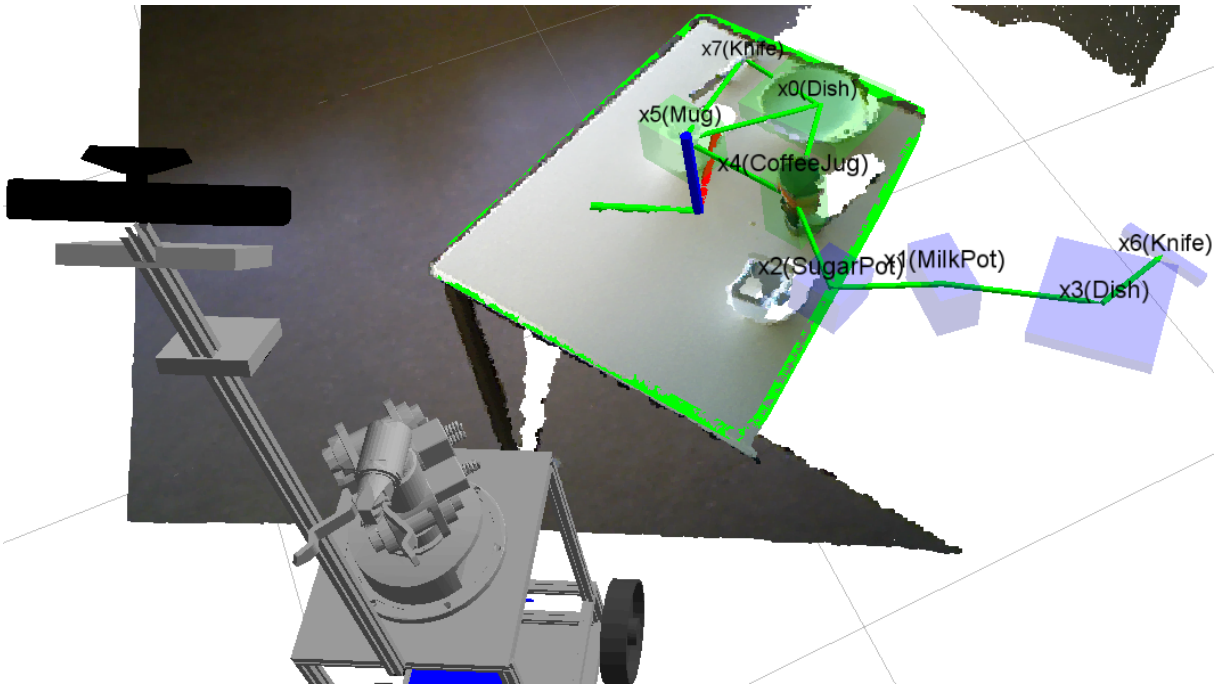



**Figure 4.2:** Robot observing a partially visible tabletop scene. Notice that the RGB-D camera only captures part of the table, but the world state model maintained by the anchoring process still retains previously observed objects. See Figure 4.7 on page 66 for a description of the visualization elements.

Our approach is to continually process single frames using a local object recognition method and integrate the object recognition results into a persistent probabilistic world model. We

then use a Conditional Random Field (CRF; Koller and Friedman, 2009) to exploit contextual relations between objects in the current scene as well as relations with previously perceived objects from the world model to improve the recognition results.

This approach has the following advantages:

- Our system can exploit contextual relations with objects that are currently out of view while still being capable of online operation (meaning that the output of the system is updated as soon as new sensor data comes in).
- The world model allows us to consistently assign the same object ID to an object without requiring that the object be constantly tracked. By anchoring fluents on the blackboard to the objects reported by the local object recognition method, the planner and plan executor can refer to an object by the same symbol even after the object has disappeared from view for a prolonged period.
- The proposed system can integrate any state-of-the-art local object recognition system that processes single frames and supplement it with additional context information, which achieves a significant boost in classification accuracy at a low computational overhead.

The next section relates our work to the state of the art. In Section 4.2, we describe the proposed context-aware anchoring system, and experimental results are reported in Section 4.3. Finally, Section 4.4 contains the discussion of the contributions in this chapter and possible future work.

## 4.1 State of the Art

The related work is grouped into four subsections: offline context-aware object recognition; single-frame context-aware object recognition; dense 3D semantic mapping; and anchoring and world modeling.

### 4.1.1 Offline, Full-Scene Context-Aware Object Recognition

As mentioned in the introduction to this chapter, the first group of related systems performs *offline* object recognition. That is, prior to object recognition, the recording of point clouds must have finished, and all point clouds must have been registered into one global point cloud. No world model is kept and updated between classifications. In contrast, the method presented

here continually fuses new information into a world model, which is required for robotic applications that require online feedback, such as plan-based task execution.

One of the most influential systems in this area is the one by Koppula et al. (2011). Starting from a registered point cloud of a room (obtained from RGBDSLAM), all planar segments in a scene are extracted. An object can consist of multiple planar segments (such as "printerFront" and "printerSide"). The system uses a probabilistic graphical model (concretely, a Markov Random Field combined with structural SVMs) to label the planar segments, exploiting local features and the geometric context between segments. An extension to the original paper shows that this model can also be used for contextual object search on a robot (Anand et al., 2013). Given a partially recognized scene, a "hallucinated" segment is inserted into several sampled locations in the scene. Based on the contextual relations to already recognized segments, the most likely location of the target segment can be inferred. The robot then moves closer to this location and repeats the one-shot recognition process.

Xiong and Huber (2010) present a similar system to semantically label a registered point cloud of the interior of a whole building that was obtained using a high-definition 3D laser scanner. Planar patches are extracted and labeled according to four object classes ("wall", "floor", "ceiling" and "clutter") using a CRF that employs both local features and context between patches. A comparison between the CRF method and a method using only local features reveals that the CRF method performs better, demonstrating that context is important in distinguishing challenging objects.

Ruiz-Sarmiento et al. (2014, 2015a) present a method that incorporates semantic knowledge from human experts into the training process of a CRF. The semantic knowledge is modeled in an OWL-DL ontology and describes the occurrence frequency, Gaussian distribution of geometric properties, and typical geometric context of each object class. This information is used to generate a synthetic dataset for training the CRF, which reduces the need for a large real data training set. The trained CRF is then used to label planar patches extracted from a full registered point cloud.

Valentin et al. (2013) first reconstruct a mesh of the whole scene from a series of depth images. Then, a CRF on the mesh is established, with each node representing a mesh face and edges being added between neighboring faces. As unary features, the CRF uses both various geometric features as well as appearance features from the corresponding RGB images. Pairwise features between two nodes represent the similarity in orientation and color between neighboring faces. In contrast to the previously presented approaches, no planar patches or individual objects are extracted; instead, each face of the mesh is assigned a semantic label.

In contrast to the proposed system, none of the systems above incorporates confidence values from a local object recognition method. Also, no system labels individual objects with arbitrary shapes; instead, they either work in the image domain (assigning a label to each pixel), the mesh domain (assigning a label to each mesh face) or on extracted planar patches. This is different in a group of papers that has emerged from the STRANDS project, where table-top objects are segmented.

In the first of these papers (Alberti et al., 2014), a Gaussian Mixture Model (GMM) is used to model the distribution of geometric features of each object category, and another set of GMMs models each object–object relation where the objects are of a different class. For each object in a scene, the objects' GMMs are used to predict the most probable class. However, context is not taken into account; instead, the object–object relations are used to compute the similarity between the current scene and trained scene types.

In the second paper (Kunze et al., 2014a), an underlying 3D local object recognition method assigns a confidence value for each object class to each object. Just as in the proposed system, the task is to improve the local recognition results using spatial context. Kunze et al. (2014a) represent context using two different types of object–object relations: Metric Spatial Relations (MSRs) and Qualitative Spatial Relations (QSRs). Based on those, two different probabilistic inference procedures are developed: a GMM-based voting scheme for the metric relations and an approach based on Bayesian inference for the qualitative relations. The two different approaches were not combined but instead evaluated separately. In the experiments, the metric relations slightly outperformed the qualitative relations. Both approaches showed better performance than the bottom-up perception alone, demonstrating that context is helpful for object recognition.

The third paper (Thippur et al., 2015) builds on the previous work by Kunze et al. (2014a) and applies the MSR- and QSR-based classifiers presented there to a new dataset consisting of snapshots of several tables that have been taken over a longer time period, i.e., with variations in the number and arrangement of objects. Again, the MSRs outperform the QSRs except in cases where not enough training samples are available. However, the authors note that QSRs are better suited for knowledge transfer, either between human and robot or from robot to robot.

The spatial relations used in this work (Section 4.2.3.2) resemble the MSRs used in Kunze et al. (2014a) and Thippur et al. (2015); however, we do not require external information about a canonical viewpoint of a scene, but only information intrinsically available to the robot.

### 4.1.2 Single-Frame Context-Aware Object Recognition

Early computer vision approaches that exploit context in object recognition are based on RGB images (see Galleguillos and Belongie (2010) for an overview). Torralba et al. (2003) first calculate a global descriptor (the "gist") of the image to classify the image as one type of scene (e.g., street or office) and use that as a prior for the objects in the scene without exploiting the relations between objects in the scene. Hoiem et al. (2008) estimate the 3D geometry of a scene from a single RGB image and use the corrected 3D position and size of objects as a prior. Galleguillos et al. (2008) use spatial context between objects and demonstrate that this improves accuracy for most object categories compared to only using co-occurrence. Shotton et al. (2009) exploit relations between objects (represented as regions in the image) in different configurations using a CRF based on inter-pixel statistics. Xiang et al. (2010) model object co-occurrence using a CRF for the task of automated image annotation (i.e., assigning multiple labels to an image without detection or segmentation of the image).

With the introduction of the Microsoft Kinect, interest in 3D computer vision has increased dramatically. Popular tasks in RGB-D scene understanding include:

1. **Image Classification:** Each input frame contains exactly one object. The task is to classify the type of object in each image, without estimating the pose of the object (for example, Eitel et al., 2015; Lai et al., 2011; Socher et al., 2012).

2. **Semantic Labeling / Semantic Segmentation:** This is currently the most popular task in RGB-D scene understanding (Song et al., 2015). The task is to label each pixel in the RGB-D image with the object class of that pixel (for example, Eigen and Fergus, 2015; Khan et al., 2014; Müller and Behnke, 2014; Ren et al., 2012; Silberman et al., 2012; Stückler et al., 2012).

3. **3D Object Detection and Pose Estimation:** In this task, not only the objects' class but also their 6D pose is retrieved (Ciocarlie et al., 2010; Fäulhammer et al., 2016; Kasaei et al., 2015; Kehl et al., 2016; Krull et al., 2015, 2017; Oliveira et al., 2014; Tremblay et al., 2018; Xiang et al., 2017), which is useful for actions like pick and place. An interesting example of its application is the Amazon Picking Challenge (Schwarz et al., 2017; Zeng et al., 2017).

By definition, the Image Classification algorithms do not take context between objects into account, since there is only one object in each image. Regarding the Semantic Labeling algorithms dealing with single frames and context, most of them work by recognizing over-segmented regions from the images and then apply a spatial consistency method such as

CRF (Khan et al., 2014; Müller and Behnke, 2014; Ren et al., 2012; Silberman et al., 2012). Most relevant to the system presented here is the third class of algorithms (3D Object Detection and Pose Estimation). However, these algorithms usually do not take context into account (except geometric consistency), but only local object features (which is why we dubbed them *local object recognition systems* here). Our system processes output from one such system and uses context between objects (even extending beyond the current sensor frame) to improve the recognition results. We have integrated two different local object recognition systems into our anchoring framework: The ROS tabletop_object_detector (Ciocarlie et al., 2010), and the spin-image-based object recognition system developed within the RACE project (Kasaei et al., 2015; Oliveira et al., 2014). In the experiments section (4.3), we only report results for the latter, since the former can only handle rotationally symmetrical objects and is based on CAD models, which makes it unsuitable for the objects present in the dataset. Our system could be combined with any of the other local object recognition systems to improve its output.

### 4.1.3 Dense 3D Semantic Mapping

The approaches discussed in the previous subsections (offline and single-frame context-aware object recognition) neither build nor maintain an internal world model. In other words, they only process the single frame (or registered set of point clouds) one at a time and do not take into account objects that are not in the current input data.

In contrast, *dense 3D semantic mapping* approaches (Hermans et al., 2014; Ma et al., 2017; McCormac et al., 2017; Stückler et al., 2015) incrementally build an internal representation by integrating each incoming RGB-D frame. Each incoming RGB-D frame is processed by a semantic image labeling method to get initial recognition guesses for each pixel/point, and this information is propagated to a point-cloud-based representation of the environment, where CRFs (or similar methods) are exploited for ensuring spatially consistent labels. The single frames are registered into the point cloud using either a SLAM method (i.e., with global loop closure) or visual odometry (without loop closure). The result is a point cloud of the full scene that is densely labeled with semantic categories.

Hermans et al. (2014) present a dense 3D semantic mapping system that is based on a random forest classifier for semantic labeling and visual odometry for point cloud registration. Stückler et al. (2015) propose a system that combines RGB-D SLAM on multi-resolution surfel maps with a random forest classifier for semantic labeling. McCormac et al. (2017) combine Convolutional Neural Networks (CNNs) for semantic labeling with a 3D-surfel-map-based

RGB-D SLAM system. The system by Ma et al. (2017) also uses CNNs together with an RGB-D SLAM method, but the system also imposes multi-view consistency during CNN training by transforming the CNN feature maps into a common reference system.

There are certain similarities between the proposed system and dense 3D semantic mapping approaches: Both incrementally fuse new data with an online model, which is a requirement for plan-based task execution, because the robot can immediately react to incoming sensor data. Furthermore, both exploit context from outside the current sensor frame to improve the classification results. However, the work presented here differs from dense 3D semantic mapping approaches in two aspects: the considered object categories, and critically the underlying world model. Since dense 3D semantic mapping strives for a dense labeling of all points in the environment, the considered object categories tend to be rather coarse and focus on large-scale structures, e.g. "wall", "table", "floor", and a general "objects" category (from the popular NYUDv2 dataset), whereas our system focuses on distinguishing objects that are relevant for robot manipulation (e.g. "plate", "spoon", "fork" etc.), so the challenges faced by both types of systems are very different. Regarding the representation, the proposed system represents individual objects and their 6DoF pose, which is a requirement for manipulation, planning and execution. Also, in a dynamic environment, an object-based representation can more easily be updated from other sources (such as another robot that is moving objects, or knowledge from human-robot interaction or other data sources).

### 4.1.4 Anchoring and World Modeling

The works presented so far have mainly focused on recognizing objects without distinguishing between different instances of an object type. The following approaches are different in the sense that they keep track of previously seen objects to support long-term robot operation and tackling the anchoring problem (i.e., resolving object identities).

The anchoring problem was first formally introduced by Coradeschi and Saffiotti (2003). Starting from a study and characterization of the problem, the authors lay out challenges arising from the anchoring problem and present several implemented systems that perform anchoring. This approach was later enhanced by Fichtner (2009, 2011), where first-order logic and the Fluent Calculus were used to formally model and represent incomplete knowledge in the anchoring problem.

Blodow et al. (2010) present an anchoring system that detects objects in the environment based on a clustering algorithm (but without doing object recognition). They use Markov

Logic Networks (a form of statistical relational learning) to assign identities to object perceptions. Like our system, it is passive, i.e., it builds a model of the objects in the robot's environment while the robot is performing its tasks. The RoboSherlock framework (Beetz et al., 2015; Blodow, 2014) builds on the idea of unstructured information management to create a framework for realizing robot perception systems. Different recognition algorithms and analysis engines can be integrated as so-called annotators, and linked to knowledge bases using first-order probabilistic reasoning.

Elfring et al. (2013) introduce probabilistic multiple hypothesis anchoring to create and update a world model. Their algorithm can track multiple models so that various kinds of prior knowledge can be accommodated. Several different experiments were performed both on 2D and 3D data, which showed that their anchoring system can successfully perform anchoring and even correct previous object–anchor associations based on new evidence.

Blumenthal et al. (2013) present a scene-graph-based world model that allows storing objects and other elements of the robot's environment while supporting uncertainty, tracking and sharing of data. Another such framework is EnViRe (Hidalgo Carrió et al., 2016), which is mainly geared towards navigation and planning systems. Both of these systems focus on the storage and exchange of data, without providing an anchoring functionality.

A basic anchoring process is also performed in the paper by Ruiz-Sarmiento et al. (2017b), where the system checks the location of two percepts, the overlapping of their bounding boxes, and their appearance to decide if they refer to the same physical element. The output of this anchoring is used to maintain a probabilistic representation of the elements in the robot workspace.

Coradeschi et al. (2013) gives a short review of further research in Symbol Grounding, including anchoring.

## 4.2 Algorithm and Implementation

The proposed context-aware anchoring system is a combination of i) a local object recognition method, ii) an anchoring process, and iii) a Conditional Random Field (see Figure 4.3). The *local object recognition method*[1] (Section 4.2.1) segments the sensor point cloud into object point clusters and yields a local recognition result for each isolated object (in the form of confidence

[1]In the following, we use the term *local* object recognition for this process to distinguish it from the *joint* object recognition performed by the CRF, which performs joint classification of all objects in the scene and takes contextual relations into account.

values for each object class). This provides the input for the *anchoring process* (Section 4.2.2), which updates its persistent world state model (a set of so-called *anchors*) with the new information. The anchoring module then creates a scene graph consisting of all the currently observed objects and all nearby objects that are currently out of view of the camera; it also adds edges between objects that are close to each other. A CRF is then built incorporating the nodes and edges of the scene graph along with their features, and the confidence values from the local object recognition (Section 4.2.3.2). Finally, a probabilistic inference process over the CRF computes the joint object recognition results, and the anchors are updated accordingly (Section 4.2.3).

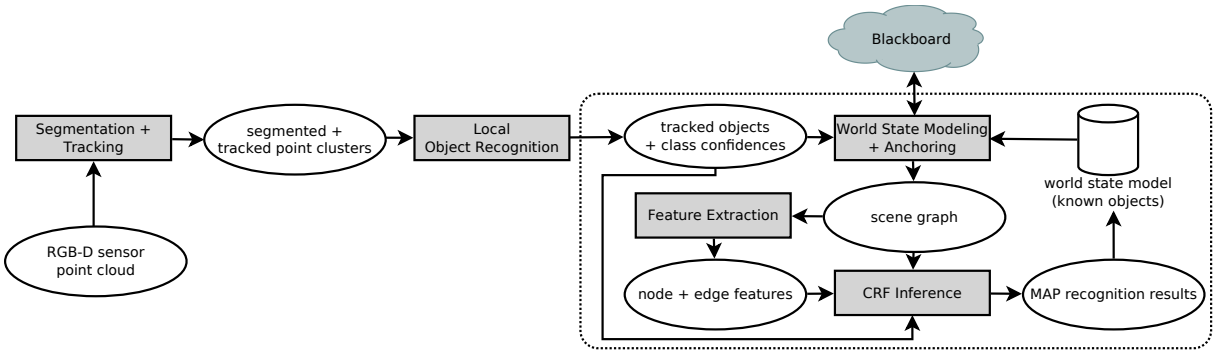



**Figure 4.3:** Overview of the proposed system. Ovals represent consumed/produced data, while boxes stand for processes. Locally tracked objects are anchored to previously observed (known) objects, and a scene graph of objects is built that includes all currently tracked objects and nearby known objects that are out of view of the camera. All objects in the scene graph are jointly classified by an inference process over a CRF, and the set of known objects is updated. Our contribution is highlighted in the dotted box.

### 4.2.1 Segmentation, Tracking and Local Object Recognition

The first steps in our processing pipeline segment the input point cloud into individual objects and perform local object recognition, which classifies each segmented object separately. The output of the local object recognition system is a set $X = \{x_1, \ldots, x_n\}$ of objects found in the input point cloud, and for each object its position, 3D bounding box and a vector $\boldsymbol{c}_{x_i} = [c_1, \ldots, c_k]$ of confidence values, representing the confidence with which object $x_i$ belongs to the respective class in the set of possible classes $L = \{l_1, \ldots, l_k\}$. The objects detected by this stage are the basic units for all later stages of the pipeline.

Here, we use the spin-image-based object recognition system developed within the RACE project (Kasaei et al., 2015; Oliveira et al., 2014). This object recognition system first performs table-top segmentation on the input point cloud: a parametric plane model is fitted to the input

point cloud using random sample consensus (RANSAC), and points that lie within a threshold of the identified plane are removed. The remaining points are clustered into objects, and tiny clusters are discarded to remove outliers. Object recognition is performed on each object $x_i$ by computing spin-image descriptors on certain keypoints of each object and comparing the result with a trained database of objects, which results in the confidence vector $\boldsymbol{c}_{x_i}$.

Apart from segmentation and local object recognition, the local object recognition system used here also provides real-time tracking of the recognized objects: a unique *track ID* is attached to all observations of the object in subsequent camera frames. Tracking is lost when the object is no longer in view of the RGB-D camera.

Note that the proposed anchoring system makes no assumptions about the local object recognition system. For instance, although the spin-based object recognition system requires that objects are separated by a certain minimum distance in order to segment them, this is not a requirement of the anchoring system. This means that the anchoring system can integrate any other 3D local object recognition method that is capable of providing a bounding box and confidence vector for each object (such as Fäulhammer et al. (2016); Kehl et al. (2016); Krull et al. (2015, 2017)). Specifically, although we make use of the tracking functionality in Section 4.2.2, our system does not strictly require it; when tracking is not provided by the local object recognition system, our system relies on its anchoring functionality alone. In fact, we have used our system to process output from the ROS tabletop_object_detector recognition system (Ciocarlie et al., 2010), which does not perform tracking, but only recognition. We did not use it in the experiments (Section 4.3), since it requires CAD models of all objects and can only handle rotationally symmetrical objects, which makes it unsuitable for the objects present in the dataset.

### 4.2.2 Anchoring

The anchoring module has two closely related functions:

1. creating and maintaining over time a persistent, probabilistic world model of the locations, identities and categories of objects perceived so far; and

2. grounding object names from a symbolic knowledge base to object identities in this world model.

As pointed out in the introduction to this chapter, even without symbolic names for objects, having a persistent world model of objects outside the robot's field of view (1.) is already beneficial to context-based joint object classification. Grounding a shared symbolic knowledge

base to this model (2.) is essential for the creation and execution of robot plans in a plan-based robot control architecture.

Our anchoring module has a persistent internal representation (the *world model*) of the objects in the environment. Following the terminology of Coradeschi and Saffiotti (2003), we call the representation of an object *anchor*. An anchor in our system is a richer representation than the purely symbolic object fluents and contains:

- the symbolic name of the object on the blackboard that corresponds to this object;
- the last known 6DoF object pose (position and orientation);
- the last observed 3D bounding box;
- the vector $\boldsymbol{c}_{x_i} = [c_1, \ldots, c_k]$ of confidence values reported last by the local object recognition for this object (see previous section);
- the MAP object type computed by the CRF (see Section 4.2.3); and
- the track ID, if the object is currently tracked by the local object recognition.

Note that an anchor contains a probabilistic representation of the possible object types in the form of the vector of confidence values $\boldsymbol{c}_{x_i}$. This vector is used by the CRF (see Section 4.2.3) when computing context relations with anchored objects outside the current camera frustum.

An anchor in our framework can be in one of two different states (Figure 4.4): *untracked* or *tracked*. If an object is currently in the robot's field of view and thus being tracked by the local object recognition, we call the corresponding anchor a *tracked anchor* and otherwise an *untracked anchor*. Figure 4.4 also shows the functionalities that create or destroy anchors and perform the transitions between states: *Find*, *Reacquire*, *Track*, *Lose*, *Discover* and *Destroy*. The first three functionalities are inspired by Coradeschi and Saffiotti (2003); the last three are additional functionalities made necessary by the bottom-up part of our framework.

*Find* creates a new anchor whenever a new object fluent is added to the blackboard by an external module. For example, the initial knowledge loader adds initially known object fluents during startup, and the plan executor adds an object fluent when an object is placed on a table by the robot. These anchors start in the *untracked* state. As soon as a matching object is detected by local object recognition, *Reacquire* binds the anchor to this object, and the anchor enters the *tracked* state. While *Find* performs top-down anchoring, *Discover* performs bottom-up anchoring: Whenever an object is reported by the object recognition module that cannot be associated with a known untracked anchor, *Discover* creates a corresponding tracked anchor.

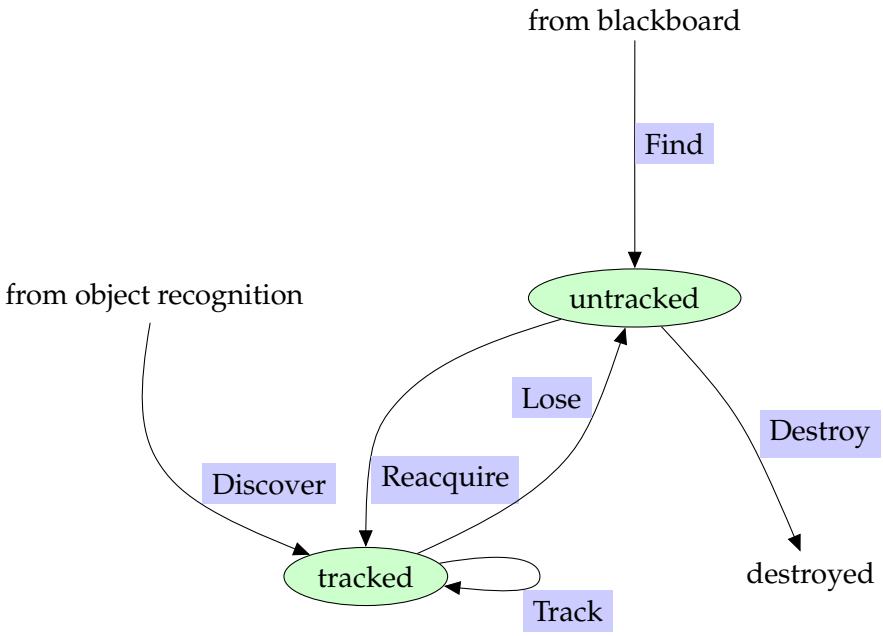



**Figure 4.4:** State diagram showing the two possible states of an anchor (green) and the functionalities that perform the transitions between them (blue).

While an object is tracked by local object recognition, *Track* updates the properties of its anchor. When it stops being tracked, *Lose* transitions the anchor to the *untracked* state. Finally, *Destroy* checks whether an object corresponding to an untracked anchor has been physically removed and deletes the corresponding anchor.

In our framework, these functionalities are implemented by executing the following steps whenever a new set of objects is reported by the local object recognition. These will be detailed in the remainder of this subsection and the following subsections.

1. **Object–Anchor Similarity Computation (Section 4.2.2.1):** Anchoring attempts to match the new objects to the nearby untracked anchors in the current world model. To do so, the *similarity* between each pair of objects is calculated.

2. **Object–Anchor Data Association (Section 4.2.2.2):** Using these similarity values, the best *assignment* of new objects to untracked anchors is calculated using the Hungarian method (Kuhn, 1955). If the cost of assigning an object does not exceed a certain threshold (i.e., the object is "similar enough" to the anchor), the object is assigned to the anchor (*Reacquire*); otherwise, a new anchor is created (*Discover*). If the object is already tracked, its track ID will be used to instantly match the new observation to its anchor and update

the anchor (*Track*). All tracked anchors that have not been assigned a tracked object are transitioned to the untracked state (*Lose*). It is also possible to integrate a local object recognition system that does not provide tracking information. In that case, all anchors are transitioned to the untracked state and must be reacquired in the next step.

3. **Removed Object Detection (Section 4.2.2.3):** Check for physically *removed objects*: All remaining nearby untracked anchors are checked whether their last known bounding box is fully within the robot's current field of view and empty; if so, the anchor is removed (*Destroy*).

4. **Joint Object Recognition (Section 4.2.3):** A scene graph for the local scene is created from the tracked anchors and nearby untracked anchors (i.e, anchors of objects that are close to currently observed objects, but occluded or outside the camera frustum). Unary features (such as size or elongation) and pairwise features (such as distance between two objects or relation between two objects with respect to the table) are extracted and added to the graph. A CRF inference procedure over the graph computes the MAP object types, which are used to update the corresponding anchors.

5. **Blackboard Synchronization (Section 4.2.4):** The changed anchors are propagated to the corresponding object fluents in the blackboard. Also, new anchors are created for object fluents that have been added to the blackboard by an external module (*Find*).

#### 4.2.2.1 Object–Anchor Similarity Computation

A prerequisite for matching newly tracked objects to existing untracked anchors is to compute the *similarity* between each tracked object and each nearby untracked anchor. Here, a higher similarity between a tracked object and an anchor means a higher likelihood that the tracked object corresponds to the physical object represented by the anchor, and thus the anchor should be assigned to that object (*reacquired*).

In our framework, there are two main features of objects that influence similarity: the difference of the two objects' classification results (specifically, the difference of their class confidence vectors $\boldsymbol{c}_{x_i}, \boldsymbol{c}_{x_i}$) and the euclidean distance between the two objects. In other words, if local object recognition reports an object with roughly the same classification in roughly the same location as one of the untracked anchors, chances are good that it is the object represented by the anchor. The importance of those two features can be explained by the assumptions we make about the information available to our algorithm and by our formulation of the problem.

We assume that the set of possible classes $L$ is already as fine-grained as possible; i.e., it is impossible to subdivide any of the classes into meaningful sub-classes based on the appearance of objects. In particular, we do not assume that any object (like "Joachim's mug") can be directly recognized based on its appearance alone. If there was such a one-to-one mapping between object appearance and object identity, the anchoring problem would be much simplified. The fact that we do not make this assumption has two consequences: first, if an object is replaced by one of similar appearance while the robot is not observing the scene, our algorithm will anchor it incorrectly; and second, if an object is moved over a large distance while the robot is not observing the scene, the anchor will be destroyed when the robot revisits the scene (see below), and the object at its new location will be assigned a new anchor when observed by the robot.

To compare different possible assignments of tracked objects to anchors (see next subsection), we need to combine the features (difference in classification results and distance) into a single similarity function. To approximate this function, a Support Vector Machine (SVM) was trained on a dataset of object/anchor pairs manually labeled as "same object"/"different object" (i.e., whether the tracked object is the same represented by the anchor).

Apart from the two features presented so far (difference of confidence vectors and distance between objects), additional SVMs were trained on different sets of features; however, the evaluation showed that the two features presented above were sufficient. All evaluated features are listed in Table 4.1. For details on the training procedure and a comparative evaluation of different sets of these features, see Section 4.3.3.

Using the trained SVM, the *similarity* between two objects $s(x_1, x_2)$ can easily be derived by computing the signed distance from the point in feature space that represents the object pair to the hyperplane that forms the SVM's decision boundary (in Hesse normal form). If this signed distance is zero, the object pair lies exactly on the decision boundary. Object pairs with a negative distance lie between the origin and the hyperplane and would be classified as "same object" by the SVM (and vice versa for positive distances). In general, the higher the signed distance, the less similar the two objects are to each other. Figure 4.10 on page 78 shows the decision boundary and the two classes (same/different) of object pairs. Note that the similarity function can also be used to compute the similarity between an object and an anchor $s(x, a)$ since all relevant properties are stored with each anchor.

**Table 4.1:** Similarity features for objects $x_1$ and $x_2$ with centroids $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and confidence vectors $\boldsymbol{c}_{x_1}$ and $\boldsymbol{c}_{x_2}$ (see Section 4.2.3.3). The functions `area_h`, `area_v`, `elong_h` and `elong_v` refer to the CRF unary features in Table 4.2 on page 69, discussed in Section 4.2.3.2.

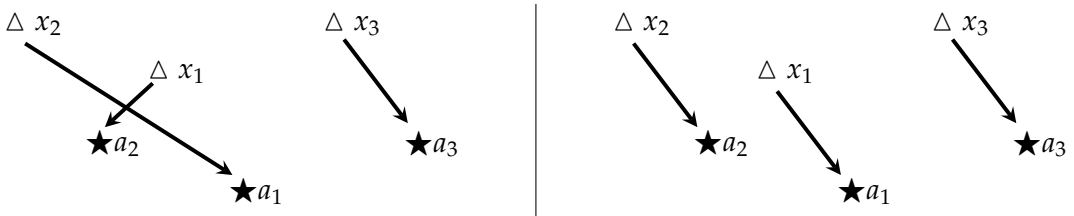
| Feature name | Definition | Description |
|---|---|---|
| `distance` | $\lvert\boldsymbol{p}_1 - \boldsymbol{p}_2\rvert$ | Distance between object centroids |
| `diff_object_types (norm)` | $\lvert\boldsymbol{c}_{x_1} - \boldsymbol{c}_{x_2}\rvert$ | Aggregated difference of the two objects' classification results (as reported by the local object recognition) |
| `diff_object_types (vector)` | $(\boldsymbol{c}_{x_1 i} - \boldsymbol{c}_{x_2 i})^2$ | Component-wise difference of the two objects' classification results (one feature for each object class $i$) |
| `diff_h_area` | $(\texttt{area\_h}(x_1) - \texttt{area\_h}(x_2))^2$ | Squared difference between horizontal bounding box areas |
| `diff_h_elong` | $(\texttt{elong\_h}(x_1) - \texttt{elong\_h}(x_2))^2$ | Squared difference between horizontal bounding box elongations |
| `diff_v_area` | $(\texttt{area\_v}(x_1) - \texttt{area\_v}(x_2))^2$ | Squared difference between vertical bounding box areas |
| `diff_v_elong` | $(\texttt{elong\_v}(x_1) - \texttt{elong\_v}(x_2))^2$ | Squared difference between vertical bounding box elongations |
| `volume_ratio` | $\texttt{volume}(x_1)/\texttt{volume}(x_2)$ | Ratio of bounding box volumes |

#### 4.2.2.2 Object–Anchor Data Association

Data association is "the process of determining the origin of sensor measurements" (Leonard and Rikoski, 2001), i.e., the process of associating uncertain sensor measurements of objects (or environment features) to existing object tracks (or features in a map). In the context of our work, the uncertain measurements are the object detections reported by the local object recognition method which we want to associate with the existing anchors.

There are numerous approaches to the data association problem. The simplest algorithm is Nearest Neighbor (NN): each detected object is greedily associated to the nearest anchor (or rather, the anchor with the highest similarity $s$, as defined above). However, NN often results in a non-optimal assignment, as illustrated in Figure 4.5. This problem is solved in the Global Nearest Neighbor (GNN) algorithm, which computes a jointly optimal assignment. A variant of GNN was used in our system.

Many more elaborate alternatives to the GNN algorithm can be found in the multi-target tracking and data association literature (Bar-Shalom et al., 2011). The most prominent represen-




**(a)** Nearest Neighbor (NN)

**(b)** Global Nearest Neighbor (GNN)

**Figure 4.5:** A comparison between the assignments produced by the NN and GNN algorithms. For NN, the order of assignments (here: $x_1, x_2, x_3$) is significant. For simplicity's sake, this figure only shows the distance between the objects, whereas the actual assignment costs are computed from the similarity function $s(x_i, a_j)$ between object and anchor, which is based on distance and appearance. (★: anchors, △: detected objects)

tatives are the Joint Probabilistic Data Association Filter (JPDAF; Blom and Bloem, 2002) and approximations of Multiple Hypothesis Tracking (MHT; Reid, 1979). These algorithms delay the decision of which object to assign to which anchor and use information from future frames to help disambiguate the assignment. This improves the assignment accuracy especially in situations with a lot of clutter, where GNN suffers from its eager assignment strategy and can produce many false positives. However, this improved ability to deal with clutter comes at the cost of a hugely increased computation time and a slight loss of real-time capability, since there is a fixed delay until final assignments are made. Since the local object detection method used here already performs some short-term tracking and clutter removal, GNN was deemed sufficient, which is confirmed by our experiments (Section 4.3.3).

Before the OBJECTANCHORASSOCIATION procedure is run, the set of objects reported by local object recognition is preprocessed. If an object is already tracked and associated with an anchor, its track ID will be used to instantly match the new observation to its anchor and update the anchor. Conversely, all anchors that are tracking an object which is no longer reported by the local object recognition method are transitioned to the "untracked" state. The remaining set of newly detected objects $X$ is passed on to the OBJECTANCHORASSOCIATION procedure shown in Algorithm 4.1, along with a set of nearby untracked anchors $A$, i.e., anchors that are in some specified gating region around the objects in $X$ and that are not associated to a currently tracked object. The output of the algorithm is a set $M$ of assignments that assigns each object $x_i \in X$ to an anchor and an updated set of anchors $A' \supseteq A$ that includes any newly created anchors in case new objects have been discovered. The first step is to build a cost matrix $\boldsymbol{C}$, where the $i$-th row and $j$-th column represents the cost of assigning $x_i$ to $a_j = s(x_i, a_j)$.

Using these similarity values, the optimal assignment of new objects to untracked anchors is calculated using the Hungarian method (Kuhn, 1955). Also known as the Kuhn–Munkres algorithm or Munkres assignment algorithm, the Hungarian method solves the assignment problem in $O(n^4)$. A later modification by Edmonds and Karp reduced this runtime to $O(n^3)$. The implementation used here (Pedregosa et al., 2011) includes this optimization; it also does not require $\boldsymbol{C}$ to be a square matrix, in contrast to the original algorithm. The output of the Hungarian algorithm is a set $I$ of index pairs of $\boldsymbol{C}$ representing the optimal assignment.

**Algorithm 4.1** ObjectAnchorAssociation

**Input:** a set of newly detected objects $X = \{x_1, \ldots, x_n\}$; a set of nearby untracked anchors $A = \{a_1, \ldots, a_m\}$; a similarity function $s(x, a)$

**Output:** a set of object–anchor assignments $M = \{\langle x_1, a' \rangle, \ldots, \langle x_n, a'' \rangle\}$; an updated set of anchors $A' = \{a_1, \ldots, a_k\}$, where $k \geq m$

1: **function** OBJECTANCHORASSOCIATION($X, A$)
2: $M \leftarrow \varnothing$
3: $A' \leftarrow A$
4: $\boldsymbol{C} \leftarrow \begin{pmatrix} s(x_1, a_1) & s(x_1, a_2) & \ldots & s(x_1, a_m) \\ s(x_2, a_1) & s(x_2, a_2) & \ldots & s(x_2, a_m) \\ \vdots & \vdots & \ddots & \vdots \\ s(x_n, a_1) & s(x_n, a_2) & \ldots & s(x_n, a_m) \end{pmatrix}$
5: $I \leftarrow$ HUNGARIANALGORITHM($\boldsymbol{C}$)
6: **for each** $\langle i, j \rangle \in I$ **do**
7: **if** $s(x_i, a_j) < 0$ **then**
8: $M \leftarrow M \cup \{\langle x_i, a_j \rangle\}$ ▷ Reacquired anchor
9: **else**
10: $a^* \leftarrow$ CREATEANCHOR($x_i$) ▷ Discovered new object
11: $A' \leftarrow A' \cup \{a^*\}$
12: $M \leftarrow M \cup \{\langle x_i, a^* \rangle\}$
13: **end if**
14: **end for**
15: **return** $M, A'$
16: **end function**

If the cost of assigning $x_i$ to $a_j$ (where $\langle i, j \rangle \in I$) does not exceed a certain threshold (i.e., the object is "similar enough" to the anchor), the object is assigned to the anchor. The natural choice for this threshold is 0 due to the definition of $s(x_i, a_j)$, where a negative value means that the SVM would classify this pair as "same object", a positive value would mean "different object", and 0 is the decision boundary.

If the cost exceeds this threshold 0, there is no anchor corresponding to this object (i.e., a

new object has been discovered). In this case, a new anchor is created and the object is assigned to that anchor.

#### 4.2.2.3 Removed Object Detection

Using the algorithm in the previous subsection, our system can discover new objects and reacquire and track existing ones. However, when the robot revisits a scene, it also has to check whether anchored objects have been physically removed in the meantime. Since the local object recognition method only yields positive information (a set of objects that have been detected), the algorithm described here has to process the raw input point cloud to perform this task.

The general idea behind the REMOVEDOBJECTDETECTION algorithm is the following. For each nearby *untracked* anchor (i.e., an anchor that could not be assigned to an existing object in the scene by the Object–Anchor Data Association), the following steps are performed:

1. check whether the bounding box of an anchor lies within the view frustum of the sensor at the current camera pose (i.e., the object is expected to be visible);
2. if yes, check whether there are any points visible inside the anchor's bounding box, or between the bounding box and the camera (that could occlude the object);
3. if the number of such points is below a certain threshold, the object is neither present nor occluded, and the anchor is destroyed.

Algorithm 4.2 on page 63 shows how these steps can be performed efficiently, given two requirements: The input point cloud $C$ must be *organized* and *projectable*. An organized point cloud is a set of points that is indexed in two dimensions, similar to a matrix or organized image. A point cloud is projectable if it is organized and the 3D coordinates of all its points can be computed from the $(u, v)$ index of a point according to a pinhole camera model. Organized, projectable point clouds are commonly produced by projective devices such as stereo cameras, Time-of-Flight cameras or structured light cameras such as the ASUS Xtion used in the experiments.

From the anchor's bounding box dimensions $\boldsymbol{d}$, the set $B$ of the eight bounding box corner points is computed and represented in homogeneous coordinates (as a four-dimensional vector, where the last component is 1). These corner points are first transformed into the camera frame by multiplying each point with the $4 \times 4$ transformation matrix $\boldsymbol{R}$ (resulting in $B'$) and then projected into the 2D image plane (resulting in $B''$ by multiplying with the $3 \times 4$ camera projection matrix $\boldsymbol{P}$.

The matrix $\boldsymbol{P}$ has the form

$$\boldsymbol{P} = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{4.1}$$

where $f$ is the camera's focal length and $c_x, c_y$ are the coordinates of the camera's principal point. The resulting elements of $B''$ have the form $\boldsymbol{b''} = (u \;\; v \;\; w)^\mathsf{T}$, where $w$ is an arbitrary scale factor, so the true pixel coordinates in the image plane are $u/w$ and $v/w$. Using $B'$ and $B''$, the algorithm checks whether all eight bounding box corners are inside the camera frustum. This is the case when each corner's $z$ coordinate lies within the minimum and maximum camera measurement depth $\hat{z}, \check{z}$, and the pixel coordinates $u/w, v/w$ are between 0 and the image width $\check{u}$ and height $\check{v}$ respectively. If at least part of the bounding box lies outside the camera frustum, the algorithm cannot confirm that the object was removed and thus returns false.

Switching back to 3D coordinates, the next step is to compute the convex hull $H$ of the bounding box corners in $B'$ and the camera origin; since the elements of $B'$ are in the camera frame, the camera origin is always at $(0 \;\; 0 \;\; 0 \;\; 1)^\mathsf{T}$. The function CONVEXHULL can be implemented efficiently using Delaunay triangulation; an example result is shown in Figure 4.6. Delaunay triangulation splits each rectangular face of the bounding box into two triangles. To increase efficiency when checking the inclusion of a point in the convex hull, such coplanar faces are removed in a post-processing step.

Any point inside this convex hull is either inside the object's bounding box (and therefore considered a potential point on the object), or between the camera and the bounding box (and therefore considered an occluding point). The final step of the algorithm is to count the number of points inside the convex hull.

This process can be sped up by making use of the fact that the input point cloud is organized. First, we compute the image-axis-aligned 2D bounding box $(u_{min}, u_{max}, v_{min}, v_{max})$ from the minimum (resp. maximum) of the $(u, v)$ coordinates of the eight bounding box corners in $B''$. Any point outside this 2D bounding box cannot be inside the 3D convex hull. Therefore, it suffices to only check the points inside the 2D bounding box for inclusion in the convex hull, greatly speeding up computation.

The actual ISINSIDE check (whether a point is inside the convex hull or not) is both straightforward and efficient. By representing the faces of the convex hull as planes in Hesse normal form, it only takes one vector multiplication for each of the 9 faces to check if the point lies on

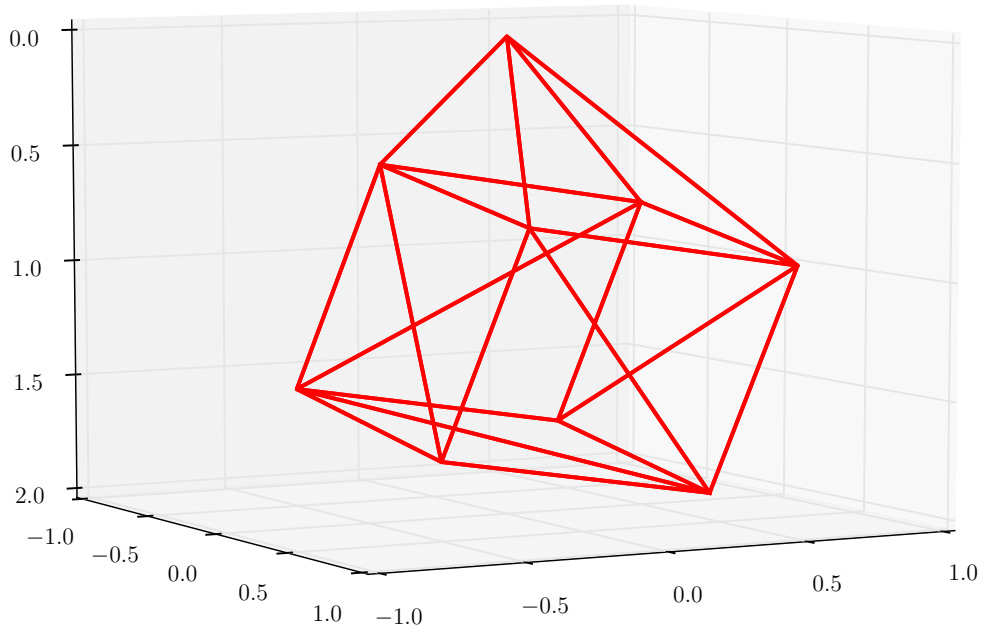



**Figure 4.6:** The convex hull of the eight corners of an object's bounding box and the camera origin (top center, at $(0\ 0\ 0\ 1)^T$). The convex hull was calculated using Delaunay triangulation, which split the rectangular faces of the bounding box into two triangles each. Such coplanar faces are removed in a postprocessing step.

the "inside" side of each plane.

If the number of points inside the hull exceeds a threshold, the algorithm returns "false" (i.e., the object is either present but could not be detected by the local object recognition, or it is occluded). Otherwise, it returns "true" (the object was removed). As was discussed in Section 4.2.2.1, an object that has been removed while the robot was not observing the scene cannot be anchored again. For this reason, when REMOVEDOBJECTDETECTION returns "true" for an anchor, the anchor is destroyed.

### 4.2.3 Joint Object Recognition

As part of our context-aware anchoring system, we need to solve the *joint object recognition* problem, where the aim is to assign to each of the objects in a scene their respective class (e.g., mug, dish, spoon), taking into account both the local features of each object and their contextual relations. To this end, we build a scene graph from the tracked anchors and all nearby untracked anchors and use a Conditional Random Field (CRF; Koller and Friedman, 2009) to compute the most probable classes of all objects, considering their context.

**Algorithm 4.2** RemovedObjectDetection

**Input:** an untracked anchor's bounding box dimensions $\boldsymbol{d} = \begin{pmatrix} d_x & d_y & d_z \end{pmatrix}^\mathsf{T}$ and pose, given as a rigid transformation $\boldsymbol{R}$ from the bounding box frame to the camera frame; an organized, projectable point cloud $C = \{\boldsymbol{p}_{11}, \boldsymbol{p}_{12}, \ldots, \boldsymbol{p}_{\check{u}\check{v}}\}$, where $\check{u}$ is the width and $\check{v}$ is the height of the image; a camera projection matrix $\boldsymbol{P}$; minimum and maximum camera measurement depth $\hat{z}, \check{z}$; a threshold on the number of points $n$ below which the object is considered removed

**Output:** true if the object was removed, false if occluded or outside camera field of view

**function** REMOVEDOBJECTDETECTION($\boldsymbol{d}, \boldsymbol{R}, C, \check{u}, \check{v}, \boldsymbol{P}, \hat{z}, \check{z}, n$)

$$B \leftarrow \left\{ \begin{pmatrix} -d_x/2 \\ -d_y/2 \\ -d_z/2 \\ 1 \end{pmatrix}, \begin{pmatrix} -d_x/2 \\ -d_y/2 \\ d_z/2 \\ 1 \end{pmatrix}, \ldots, \begin{pmatrix} d_x/2 \\ d_y/2 \\ d_z/2 \\ 1 \end{pmatrix} \right\}$$

$B' \leftarrow \{\boldsymbol{R}\boldsymbol{b} \mid \boldsymbol{b} \in B\}$

$B'' \leftarrow \{\boldsymbol{P}\boldsymbol{b'} \mid \boldsymbol{b'} \in B'\}$

**if** $\exists \boldsymbol{b'} = \begin{pmatrix} x & y & z & 1 \end{pmatrix}^\mathsf{T} \in B'$: $z \notin [\hat{z}, \check{z}]$ **then**

  **return** false

**else if** $\exists \boldsymbol{b''} = \begin{pmatrix} u & v & w \end{pmatrix}^\mathsf{T} \in B''$: $({}^{u}/_{w} \notin [0, \check{u}]) \vee ({}^{v}/_{w} \notin [0, \check{v}])$ **then**

  **return** false

**else**

  $H \leftarrow \text{CONVEXHULL}\left(B' \cup \left\{\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^\mathsf{T}\right\}\right)$

  $u_{min}, u_{max}, v_{min}, v_{max} \leftarrow \min_{u/w}\left\{\begin{pmatrix} u & v & w \end{pmatrix}^\mathsf{T} \in B''\right\}, \ldots$

  $C' \leftarrow \{\boldsymbol{p}_{uv} \in C \mid (u_{min} \leq u \leq u_{max}) \wedge (v_{min} \leq v \leq v_{max}) \wedge \text{ISINSIDE}(\boldsymbol{p}_{uv}, H)\}$

  **return** $|C'| \geq n$

**end if**

**end function**

#### 4.2.3.1 CRF Formulation for Joint Object Recognition

We model the joint object recognition problem as a CRF as follows. Let $X = \{x_1, \ldots, x_n\}$ be a set representing $n$ observed objects within a given scene, where each object $x_i$ is characterized through a vector of $m$ features $\boldsymbol{f}_{x_i u} = \begin{pmatrix} f_{x_i u_1} & \ldots & f_{x_i u_m} \end{pmatrix}^\mathsf{T}$, e.g., size, height or elongation, $L = \{l_1, \ldots, l_k\}$ the set of the $k$ possible object classes, and $Y = \{y_i, \ldots, y_n\}$ a set of discrete random variables over $L$, that assign to each object in $X$ a class from $L$. Thus, the joint object recognition problem, modeled by a CRF, is the problem of maximizing the probability distribution $P(Y|X)$, i.e., to find the most probable classes assignation to $Y$ given the characterized objects' observations in $X$.

A CRF is represented through a graph structure $H = (V, E)$, where $V$ is a set of nodes

representing random variables, and $E$ a set of edges linking related nodes. Concretely, in the joint object recognition problem, each variable in $Y$ introduces a node in $V$, and two contextually related variables[2] set an edge in $E$ between their respective nodes. Then, according to the Hammersley–Clifford theorem (Koller and Friedman, 2009), a number of functions called factors are defined over parts of $H$, encoding each one a piece of $P(Y|X)$. In this work, we rely on two factor types: *unary*, related to nodes, and *pairwise*, associated with edges. The insight behind this is that unary factors encode the likelihood of a variable $y_i$ to be assigned to a certain class $l_i$ given the characterized object $x_i$, while pairwise factors express the compatibility of two related variables belonging to a certain pair of classes.

Concretely, unary factors $U(\cdot)$ and pairwise factors $I(\cdot)$ are defined by a linear classification model as follows:

$$U(y_i, x_i, \boldsymbol{\omega}) = \sum_{l \in L} \delta(y_i = l) \boldsymbol{\omega}_l f(x_i) \tag{4.2}$$

$$I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) = \sum_{l_1 \in L} \sum_{l_2 \in L} \delta(y_i = l_1) \delta(y_j = l_2) \boldsymbol{\theta}_{l_1, l_2} g(x_i, x_j) \tag{4.3}$$

where $f(x_i)$ is the function that computes the vector of unary features $\boldsymbol{f}_{x_i u}$ of the object $x_i$, $g(x_i, x_j)$ provides the pairwise features $\boldsymbol{f}_{x_i x_j p} = (f_{x_i x_j p_1} \ \dots \ f_{x_i x_j p_q})^\mathsf{T}$ for objects $x_i$ and $x_j$, $\boldsymbol{\omega}_l = (\omega_{1,l} \dots \omega_{m,l})$ and $\boldsymbol{\theta}_{l_1,l_2} = (\theta_{1,l_1,l_2} \dots \theta_{q,l_1,l_2})$ are vectors of weights associated to the class $l$ and the combination of classes $l_1$ and $l_2$ respectively, both learned during the CRF training, and $\delta(y_i = l)$ is the Kronecker delta function.

Once these factors have been defined, the computation of $P(Y|X)$ can be expressed through log-linear models as:

$$P(Y|X, \boldsymbol{\omega}, \boldsymbol{\theta}) = \frac{1}{Z(X, \boldsymbol{\omega}, \boldsymbol{\theta})} e^{-\epsilon(Y, X, \boldsymbol{\omega}, \boldsymbol{\theta})} \tag{4.4}$$

where $Z(\cdot)$ is the partition function, which plays a normalizing role so that $\sum_{\xi(Y)} P(Y|X, \boldsymbol{\omega}, \boldsymbol{\theta}) = 1$, with $\xi(Y)$ being a possible assignation to the variables in $Y$, and $\epsilon(\cdot)$ is the so-called energy function defined as the sum of all the factors defined over the graph:

[2] Two variables $y_i$ and $y_j$ are related if their associated objects $x_i$ and $x_j$ are located close to each other in the scene.

$$\epsilon(Y, X, \boldsymbol{\omega}, \boldsymbol{\theta}) = \sum_{i \in V} U(y_i, x_i, \boldsymbol{\omega}) + \sum_{(i,j) \in E} I(y_i, y_j, x_i, x_j, \boldsymbol{\theta}) \tag{4.5}$$

The CRF training yields the weights $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ that maximize the likelihood function:

$$\max_{\boldsymbol{\omega}, \boldsymbol{\theta}} L_P(\boldsymbol{\omega}, \boldsymbol{\theta} : D) = \max_{\boldsymbol{\omega}, \boldsymbol{\theta}} \prod_{d \in D} P\left(y_d | x_d\right) \tag{4.6}$$

where $D$ is the set of all the scenes used for training, $x_d$ is a set containing the characterized objects in the scene $d \in D$, and $y_d$ are their corresponding ground truth classes.

Since solving Equation 4.6 requires the computation of the partition function, which is intractable in practice, usually an approximated function replaces it. In this work, we employ the pseudo-likelihood function, which has shown that, given a large number of training samples, the optimization results converge to the ones yielded by the likelihood function (Koller and Friedman, 2009).

Despite this simplification, the learning process remains complex if the number of considered object classes $|L|$ and features used to describe them $\left|\boldsymbol{f}_{x_i u}\right|$ and their relations $\left|\boldsymbol{f}_{x_i x_j p}\right|$ is high, which results in a large number of weights as well. On the other hand, it is desirable to account for a comprehensive variety of features to properly characterize both objects and relations. In Section 4.2.3.3 we describe our approach to tackle this issue.

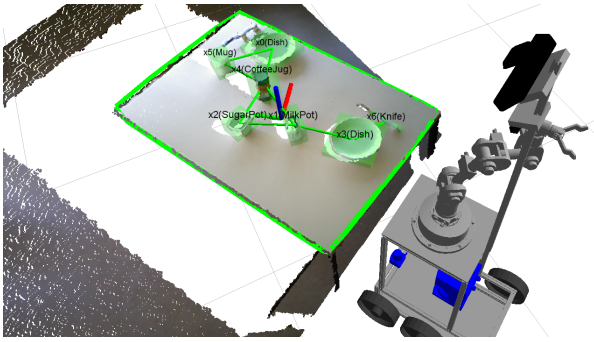
#### 4.2.3.2 Scene Graph Construction and Feature Extraction

A scene graph for the local scene is created by first adding one node for each tracked anchor, then recursively adding nodes for untracked anchors that are closer than a certain context range to a node already in the graph (which can include known objects that are close to currently observed objects, but outside the camera frustum). An edge is added to the graph between each pair of nodes that are within context range of each other (see Figure 4.7).

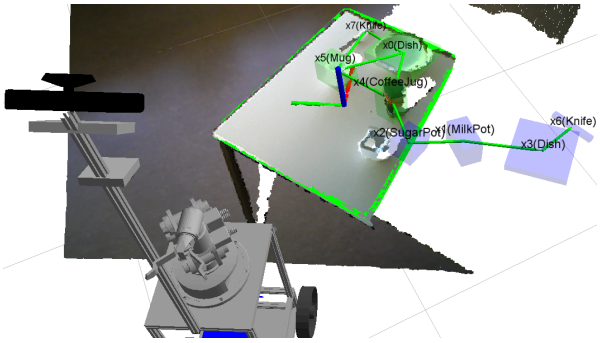
The scene graph corresponds to a CRF graph structure $H = (V, E)$, where $V$ is the set of nodes, each one associated with a random variable from $Y$ representing the class of an object $x_i$, and $E$ is the set of edges/relations $E$ between those objects.

These objects and relations are subsequently characterized through the vectors of features $\boldsymbol{f}_{x_i u}$ and $\boldsymbol{f}_{x_i x_j p}$ respectively, which are integrated into the CRF model as introduced in Equation 4.2 and Equation 4.3 on the facing page.

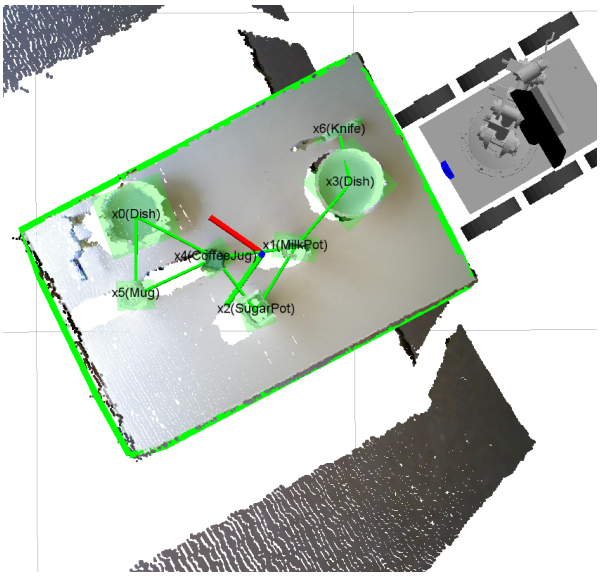
The features are based on the objects' bounding boxes and the table boundary and center.



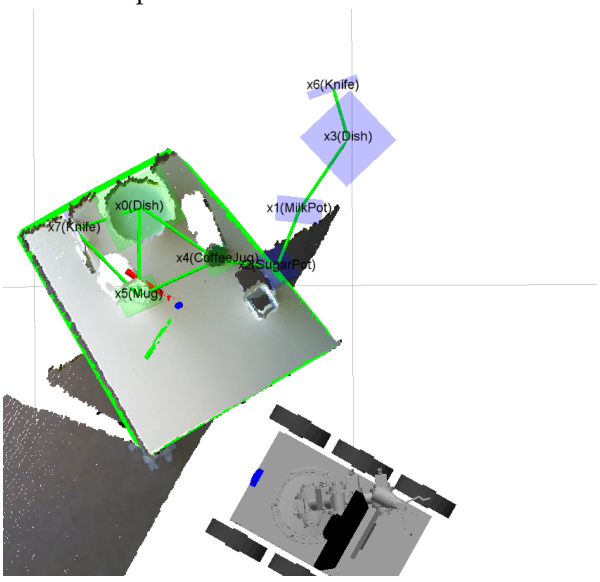

**(a)** robot observing a tabletop scene




**(b)** a later observation of the same scene from a different viewpoint




**(c)** top-down view of (a)




**(d)** top-down view of (b)

**Figure 4.7:** Two subsequent robot observations of a tabletop scene. Transparent boxes: bounding boxes of anchored objects with their CRF classification result (CRF nodes; green: currently tracked, blue: not currently tracked); lines: context edges between nearby objects (CRF edges). Also shown is the bounding polygon of the currently observed table surface and its center.

The bounding boxes are computed so that their $z$-axis aligns with the map frame's $z$-axis, which is pointing upwards. The table boundary is computed from the sensor data and represented as a (not necessarily rectangular or convex) polygon. Figure 4.8 illustrates the table boundary and some of the features that relate to it.

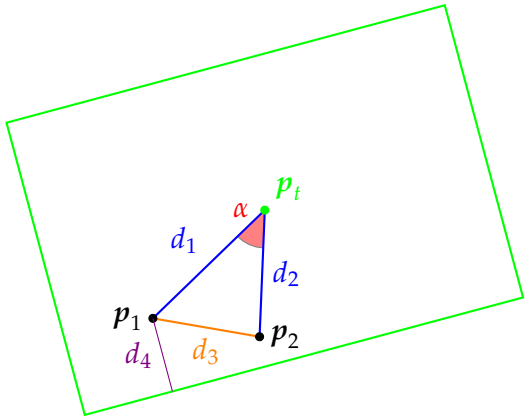



**Figure 4.8:** Illustration of some of the CRF unary and pairwise features (Tables 4.2 and 4.3).
$p_1$, $p_2$: object centroids
$p_t$: table center
$d_1$, $d_2$: distance from $p_1$ (resp. $p_2$) to $p_t$ (`table_center_distance`$(x_1)$, `table_center_distance`$(x_2)$)
$d_3$: horizontal distance between $p_1$ and $p_2$ (`hdist_centroids`$(x_1, x_2)$)
$d_4$: distance from $p_1$ to table border (`table_border_distance`$(x_1)$)
$\alpha$: angle between $p_1$, $p_2$ and $p_t$ (`table_angle_diff`$(x_1, x_2)$).

The unary features used here are listed in Table 4.2. They describe the shape, size and position on the table of an object independent of other objects. This allows the CRF to learn both the typical shape and size of different object classes as well as their typical position on the table: some objects might occur more often near the center of a table, while others are usually near the table border.

The pairwise features are listed in Table 4.3 and describe the relative size and position on the table of two objects. These features allow the CRF to exploit context between the objects; for example, some objects usually occur close to each other or at the same distance from the table border.

We chose the features to fulfill two criteria. The first criterion is that the features should be continuous/quantitative and not discrete/qualitative. The reason for preferring quantitative over qualitative features is that it avoids hard discrete jumps in the value of the feature near thresholds, which would negatively impact the classification performance and would introduce another set of parameters (namely, the thresholds).

The second criterion is that the features should not rely on any implicit reference poses that arise from peculiarities of how the robot has recorded its data. For example, Anand et al. (2013) use the node feature "Distance from the scene boundary". In their system, the scene boundary is implicitly given by the fringe of the area mapped by the robot. Since in their experiments, the robot has always mapped exactly one room, this feature gives the distance to the room's walls; however, this feature is not applicable when only part of a room or multiple rooms are mapped. Another example from Anand et al. (2013) is the edge feature "Relative position from camera (in-front-of /behind)". Since their system runs offline on a pre-registered dataset from many camera poses, there is no unique reference camera pose. The paper does not include details, but presumably the "camera pose" here refers to the origin of the registered map, which is usually arbitrarily set to the pose of the first camera frame. This again introduces a dependency on a particularity of the used dataset. Likewise, the features used by Kunze et al. (2014a) use a reference system that is based on the front-left table corner. This is only meaningful because all of their datasets have been recorded from the "front" of the office desks, i.e., with the monitor in the back and the keyboard closer to the robot.

The only assumption made in the features used in this work is that the sensor data is represented in (or can be transformed into) a coordinate frame where the $z$-axis points up, which is the case on all ground-based robots. Some of the features are expressed in relation to a table, but that table center and boundary are extracted from the sensor data by our system and are independent of any reference poses. Without a reference pose, it is not directly possible to capture "in front of / behind / left of / right of" relations; however, the `table_center_distance` and `table_border_distance` features capture some of the "in front of / behind" information, and the signed `table_angle_diff` feature approximates the "left of / right of" relation.

#### 4.2.3.3 Integrating Local Object Recognition Results

Our CRF model can be enriched with the results from any local object recognition method (or a combination of them) able to provide a confidence vector of its results, which permits the recognition system to take advantage of methods exploiting specialized feature descriptors. For that, we introduce the confidence vector $\boldsymbol{c}_{x_i}$ of an object $x_i$ into the usual CRF unary factors formulation (see Equation 4.2) in the following way:

$$U(y_i, x_i, \boldsymbol{\omega}) = \sum_{l \in L} \delta(y_i = l) \boldsymbol{\omega}_l f(x_i) \boldsymbol{c}_{x_i}(l) \tag{4.7}$$

**Table 4.2:** CRF unary features. The variables $d_x$, $d_y$, $d_z$ used in the first five features are the dimensions of the object's bounding box; the $z$-axis is oriented upwards. The two features `table_border_distance` and `table_center_distance` are illustrated in Figure 4.8.

| Feature name | Definition | Description |
|---|---|---|
| `volume` | $d_x d_y d_z$ | Volume of the bounding box |
| `area_h` | $d_x d_y$ | Horizontal area of the bounding box |
| `area_v` | $d_z \sqrt{(d_x)^2 + (d_y)^2}$ | Vertical area of the bounding box |
| `elong_h` | $\max(d_x, d_y) / \min(d_x, d_y)$ | Horizontal elongation of the bounding box |
| `elong_v` | $d_z / \sqrt{(d_x)^2 + (d_y)^2}$ | Vertical elongation of the bounding box |
| `table_border_distance` | — (see Figure 4.8) | Distance to the table border polygon |
| `table_center_distance` | — (see Figure 4.8) | Distance to the table center |

**Table 4.3:** CRF pairwise features for an edge between objects $x_1$ and $x_2$ with centroids $\boldsymbol{p}_1 = \begin{pmatrix} p_{x1} & p_{y1} & p_{z1} \end{pmatrix}^\mathsf{T}$ and $\boldsymbol{p}_2 = \begin{pmatrix} p_{x2} & p_{y2} & p_{z2} \end{pmatrix}^\mathsf{T}$. The functions refer to the unary features in Table 4.2. The three table-related features are illustrated in Figure 4.8.

| Feature name | Definition | Description |
|---|---|---|
| `bias` | 1 | Bias term (models co-occurrence probability between object classes) |
| `volume_ratio` | $\texttt{volume}(x_1) / \texttt{volume}(x_2)$ | Ratio of bounding box volumes |
| `hdist_centroids` | $\sqrt{(p_{x1} - p_{x2})^2 + (p_{y1} - p_{y2})^2}$ | Horizontal distance between centroids |
| `table_border_diff` | $\texttt{table\_border\_distance}(x_1)$ $- \texttt{table\_border\_distance}(x_2)$ | Difference of distances to the table border |
| `table_center_diff` | $\texttt{table\_center\_distance}(x_1)$ $- \texttt{table\_center\_distance}(x_2)$ | Difference of distances to the table center |
| `table_angle_diff` | — (see Figure 4.8) | Signed angle between $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and table center |

Thus, the components of the confidence vector play the role of an additional feature, not related to any weight, that have a different value for each object class. This integration also leads to a reduction in the number of weights present during the CRF learning, alleviating the training complexity. In this way, the CRF focuses on exploiting high-level features of objects and relations and releases the work with specialized feature descriptors to the local method, which modeling into the CRF typically involves a large number of weights. For example, in our experiments, we have considered 9 different object classes, 7 unary and 6 pairwise features (see Section 4.2.3.2). The number of weights associated with features of objects is $\left|\boldsymbol{f}_{x_i u}\right| \cdot |L| = 7 \cdot 9 = 63$, while the number of those related to contextual features is $\left|\boldsymbol{f}_{x_i, x_j p}\right| \cdot |L|^2 = 6 \cdot 81 = 486$, giving a total of 549 weights. The local object recognition system represents each object by a set of 10 shape features, each one with a descriptor vector of 45 components, resulting in a total descriptor length of 450. Thus, their modeling into a CRF would suppose the addition of $450 \cdot |L| = 4050$ weights, which clearly would increase the training complexity, even dropping the recognition performance.

#### 4.2.3.4 Probabilistic Inference

Once the CRF has been modeled including: a graph representation $H = (V, E)$, the extracted features of the graph components, and the confidence values from the local object recognition method, a probabilistic inference process over such a CRF performs joint object classification. Concretely, it finds an assignment of classes $\hat{Y}$ to the objects $X$ that maximizes the probability $P(Y|X)$, that is:

$$\hat{Y} = \arg\max_{Y} P(Y|X, \boldsymbol{\omega}, \boldsymbol{\theta}) \tag{4.8}$$

This calculus is commonly referred to as the Maximum a Posteriori problem (MAP), which in this work is carried out utilizing the Iterated Conditional Modes method (Besag, 1986), an approximated solution that mitigates the heavy computational burden required by exact approaches. We have made our implementation of the aforementioned training and inference processes available as open source (Ruiz-Sarmiento et al., 2015b).

### 4.2.4 Blackboard Synchronization

The context-aware anchoring system is embedded in the rest of the plan-based robot control architecture via the blackboard (see Figure 4.1 on page 42). The anchoring system however

maintains its own internal world model (see Section 4.2.2) instead of directly storing all anchors on the blackboard for the following reasons: (1) the anchors in the internal world model are a richer representation of the objects (including a probabilistic representation of the possible object types) than the purely symbolic fluents on the blackboard, and (2) the internal anchors have to be updated at a high frequency, while the blackboard is designed to store knowledge over longer time horizons with slower update rates.

In order to make the poses and identities of objects available to other modules (e.g., the planner and plan executor), the internal world model of the anchoring system is continually synchronized with the blackboard. This synchronization happens in two directions, as explained next: (1) reading fluents from the blackboard and (2) writing fluents to the blackboard.

**Reading fluents from the blackboard** When starting up, all already existing object-related fluents are retrieved from the blackboard using the following SPARQL query:

```
SELECT DISTINCT ?instance WHERE
{
  ?instance upper:hasBoundingBox ?any .
  ?instance a ?subclass .
  {
    {?subclass rdfs:subClassOf+ race:Kitchenware}
    UNION {?subclass rdfs:subClassOf+ race:PersonalItem}
  }
}
```

This query is also registered with the blackboard, so whenever a new fluent is added to the blackboard that matches this query, the anchoring system is immediately notified.

A sample set of object fluents retrieved by this query is shown in Listing 4.1 on page 73. Each object is represented by a set of fluents. The main object fluent (here: `mug1`) states the object's class and a list of poses and bounding boxes. Each bounding box fluent specifies the dimensions of the bounding box and its centroid pose. Note that this results in each object having two poses. The reason is that when a CAD-model-based local object recognition is used, the object's pose relates to the CAD model's reference frame, and the pose of the bounding box is always the bounding box's centroid (see Figure 4.9 for an illustration). For local object recognition methods that are not based on CAD models, the object's pose and the bounding box's pose coincide.

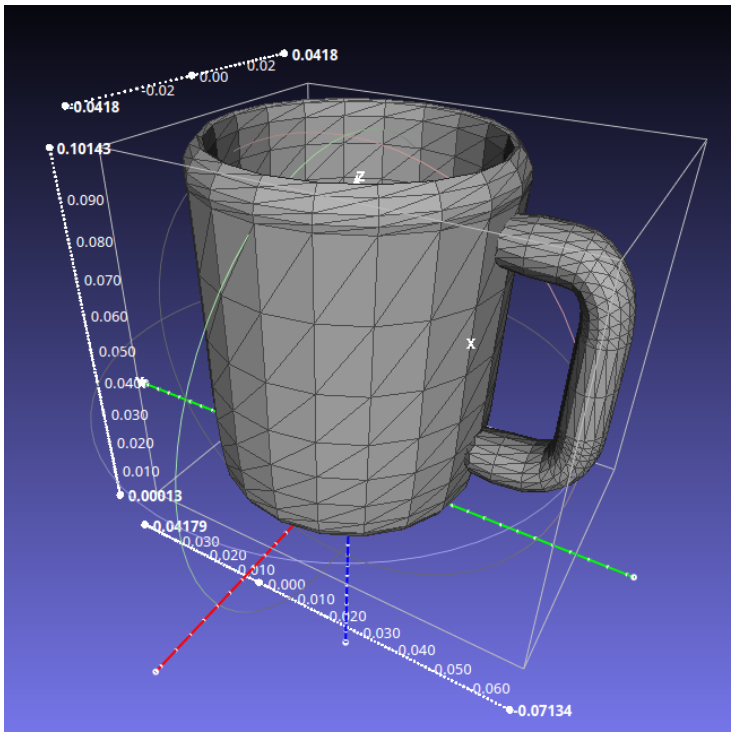



**Figure 4.9:** CAD model of the object "mug". The colored axes show the object's reference frame, and the white box is the object's bounding box.

**Writing fluents to the blackboard** When a new object is *discovered*, a corresponding fluent is immediately created on the blackboard. Conversely, when an anchor is *destroyed*, the corresponding fluent is ended immediately.

When an existing fluent is updated (via *Reacquire* or *Track*), the object's pose is updated as shown in Listing 4.2 on page 74:

- The existing pose, bounding box and bounding box pose fluents are ended (i.e., the `FinishTime` is set to the current time).
- New pose, bounding box and bounding box pose fluents are created which start at the current time and are open (i.e., the FinishTime is unknown, signified by the special value 0).
- The new fluents are added to the main object fluent (here: `mug1`).

By storing each object pose in a new fluent (instead of simply overwriting the old fluents), the blackboard enables reconstruction of an object's full trajectory over time. This was one

**Listing 4.1** Initial fluents about the object `mug1` from an external knowledge source. Properties not explicitly specified (like `hasRoll`, `hasPitch`) have the default value 0.

```
!Fluent
Class_Instance: [Coffee, mug1]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseMug1_0]
   - [hasBoundingBox, BBox, bBoxMug1_0]
---
!Fluent
Class_Instance: [Pose, poseMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.30]
   - [hasZ, xsd:float, 0.71]
   - [hasYaw, xsd:float, -1.40]
```

```
!Fluent
Class_Instance: [BBox, bBoxMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseBBMug1_0]
   - [hasXSize, xsd:float, 0.070]
   - [hasYSize, xsd:float, 0.094]
   - [hasZSize, xsd:float, 0.084]
---
!Fluent
Class_Instance: [Pose, poseBBMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.29]
   - [hasZ, xsd:float, 0.75]
   - [hasYaw, xsd:float, 0.0]
```

of the blackboard's design goals since this functionality is essential for other tasks within the RACE project (such as learning from experiences).

To avoid overloading the blackboard with minute pose changes at a high frequency, the object poses are only updated if the pose difference is above a certain threshold or after a timeout.

## 4.3 Results

To evaluate our system, we collected a dataset of 15 different scenes. After a description of the dataset and cross-validation scheme, we first present a separate evaluation of the object association part of our system (Section 4.3.3). This is followed by an evaluation of the complete system (Section 4.3.4).

### 4.3.1 Data Recording and Preparation

In recent years, many RGB-D datasets have become available; a recent survey (Firman, 2016) lists 102 different 3D datasets, with the majority having been recorded with an RGB-D camera, mostly the Microsoft Kinect 1. Some of the best-known datasets are:

**Listing 4.2** Updated fluents about the object `mug1` after one observation

```
!Fluent
Class_Instance: [Coffee, mug1]
StartTime: [0.0, 0.0]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseMug1_0]
   - [hasBoundingBox, BBox, bBoxMug1_0]
   - [hasPose, Pose, poseMug1_1]
   - [hasBoundingBox, BBox, bBoxMug1_1]
---
!Fluent
Class_Instance: [Pose, poseMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [1423225357.28, 1423225357.28]
Properties:    # unchanged
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.30]
   - [hasZ, xsd:float, 0.71]
   - [hasYaw, xsd:float, -1.40]
---
!Fluent
Class_Instance: [BBox, bBoxMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [1423225357.28, 1423225357.28]
Properties:    # unchanged
   - [hasPose, Pose, poseBBMug1_0]
   - [hasXSize, xsd:float, 0.070]
   - [hasYSize, xsd:float, 0.094]
   - [hasZSize, xsd:float, 0.084]
---
!Fluent
Class_Instance: [Pose, poseBBMug1_0]
StartTime: [0.0, 0.0]
FinishTime: [1423225357.28, 1423225357.28]
Properties:    # unchanged
   - [hasX, xsd:float, 5.59]
   - [hasY, xsd:float, 10.29]
   - [hasZ, xsd:float, 0.75]
   - [hasYaw, xsd:float, 0.0]
!Fluent
Class_Instance: [Pose, poseMug1_1]
StartTime: [1423225357.28, 1423225357.28]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.63]
   - [hasY, xsd:float, 10.24]
   - [hasZ, xsd:float, 0.79]
   - [hasRoll, xsd:float, 0.015]
   - [hasPitch, xsd:float, 0.001]
   - [hasYaw, xsd:float, 2.186]
---
!Fluent
Class_Instance: [BBox, bBoxMug1_1]
StartTime: [1423225357.28, 1423225357.28]
FinishTime: [0.0, 0.0]
Properties:
   - [hasPose, Pose, poseBBMug1_1]
   - [hasXSize, xsd:float, 0.101]
   - [hasYSize, xsd:float, 0.107]
   - [hasZSize, xsd:float, 0.076]
---
!Fluent
Class_Instance: [Pose, poseBBMug1_1]
StartTime: [1423225357.28, 1423225357.28]
FinishTime: [0.0, 0.0]
Properties:
   - [hasX, xsd:float, 5.63]
   - [hasY, xsd:float, 10.24]
   - [hasZ, xsd:float, 0.79]
   - [hasRoll, xsd:float, 0.015]
   - [hasPitch, xsd:float, 0.001]
   - [hasYaw, xsd:float, 2.186]
```

1. NYU Depth v2 (Silberman et al., 2012), the successor of the popular NYU Depth v1 dataset (Silberman and Fergus, 2011)

2. Berkeley B3DO (Janoch et al., 2013)

3. SUN3D (Xiao et al., 2013)

4. SUN RGB-D (Song et al., 2015)

The first three datasets are geared solely towards the evaluation of semantic segmentation and labeling algorithms, i.e., they contain per-pixel class labels for each frame, but no bounding box or object pose. The SUN RGB-D dataset addresses this shortcoming by incorporating selected images from the NYU Depth v2, Berkeley B3DO and SUN3D datasets and own images, and adding additional ground truth information, like 3D bounding boxes and object orientation. Also, the SUN RGB-D dataset provides a transformation for each frame that aligns the frame to the gravity direction. Since this information was missing from the original datasets, the annotation was done semi-automatically using the distribution of normals as a heuristic and correcting errors manually.

What is missing from all standard benchmark datasets however is a transformation of each RGB-D frame to a global coordinate frame, not only gravity direction. This is due to the fact that the datasets were recorded using a handheld camera. In robotics, it is commonplace to have access to this information from laser scanner-based localization, so our algorithm uses it to get an initial guess for the position of previously encountered objects. Also, the two local object recognition methods (Ciocarlie et al., 2010; Oliveira et al., 2014) that were integrated with our system only deal with tabletop scenarios, whereas in the standard benchmarking datasets many large-scale structures are labeled (e.g., wall, ceiling, book cabinet). This is also the case in the recent Robot@Home dataset (Ruiz-Sarmiento et al., 2017a), that although includes a global coordinate frame, also contains those structures. For this reason, we decided to record our own dataset using a mobile robot.

This dataset consists of 15 scenes inspected by a robot equipped with an RGB-D camera driving around a table and turning towards it from different locations. To demonstrate that the approach is independent of the used robot, 5 of the scenes were recorded using the PR2 robot and 10 were recorded using the Calvin robot (see Section 2.6.2). The table contained several objects in varying table settings. In total, the dataset contains 1387 seconds of observation and 144 unique objects from 9 categories: sugar pot, milk pot, coffee jug, mug, dish, fork, knife,

spoon, and table sign.[3]

Segmentation, tracking and local object recognition were run on the recorded sensor data, and its output (tracked objects and local recognition results) was added to the dataset. Since the objects were observed from multiple perspectives and tracking was lost while the robot was moving from one observation pose to another, the dataset contains more than one track ID for most objects (one for each subsequent observation of the object). Each track ID was manually labeled with the ground truth class of the object it represented. Additionally, all track IDs belonging to the same object were manually grouped to allow evaluation of the anchoring process. Track IDs that did not correspond to any object on the table (but instead to objects on different tables, pieces of the table itself or other artifacts) were manually removed. In total, out of 432 track IDs, 410 (94.9 %) were associated with true objects, while 22 (5.1 %) were removed as artifacts.

### 4.3.2 Cross-Validation

For evaluation, $k$-fold cross validation was performed. When choosing the splitting scheme of the data into folds, it is important to account for the fact that the samples in the dataset are not independently and identically distributed. Instead, there is a high statistical dependency between samples from each of the 15 scenes, since the configuration of objects within a scene did not change (only the perspective of the robot changed). A random split of the samples into the $k$ folds would result in overfitting and an inflated validation score since the system would be tested on samples that are artificially close to the training samples. For this reason, a splitting scheme was chosen where iteratively all samples from one scene were left out of the dataset during training, and those samples were then used for testing. This cross-validation scheme is sometimes called leave-one-label-out[4] (LOLO; scikit-learn documentation, 2021). In the neurosciences, it is called leave-one-subject-out (LOSO; Esterman et al., 2010).

### 4.3.3 Object Association

This section evaluates the capabilities of the anchoring process carried out before contextual relations are exploited. This step first computes the similarities among the objects being observed and the set of anchors already present in the system (Section 4.3.3.1) and then associates these

[3]The dataset is available at the following URL: `https://doi.org/10.5281/zenodo.1257047`

[4]The term "label" in this context denotes the scene to which a sample belongs, not the more common usage of the target class of a sample.

observations to anchors (Section 4.3.3.2). As illustrated in Figure 4.3 on page 51, the outcome of this anchoring is further updated after contextual object recognition is carried out.

#### 4.3.3.1 Object–Anchor Similarity Computation

We used the dataset to train and evaluate the SVM that computes the similarity between two object observations to decide if they are observations of the same object or not (see Section 4.2.2.1). To transform the dataset into training and test data for the SVM, we ran our system while simulating "perfect" object association: instead of using the combination of the SVM and the Hungarian Algorithm for object–anchor data association, we used ground truth to associate new track IDs to the correct anchors. For each combination of new track IDs and untracked anchor, the similarity features were computed, and a sample was added to the transformed dataset. The sample was labeled as "same object" if the association was correct according to ground truth and "different object" otherwise.

Figure 4.10 shows a scatter plot of the transformed data set. For visualization purposes, only two similarity features are used in the figure (*distance* and *diff_object_types (norm)*). As expected, samples labeled "same object" are grouped in the lower right corner: Both distance and object type difference are low. Somewhat surprisingly, the distance between object observations ranges up to 0.3 m; since the objects were not moved during the recording of the dataset, this observed distance is fully caused by localization errors. The figure also shows that the two classes are not linearly separable and that both the distance between observed positions over time (feature "distance") and similar appearance (feature "diff_object_types") are useful features to correctly associate an object to previous observations.

To evaluate which features yield the best classification results, the SVM was trained on four different sets of features:

**(a) distance + diff_object_types (norm):** Distance between the centroids of the object observations + norm of the difference vector between the object types. This feature set corresponds to Figure 4.10.

**(b) distance + diff_object_types (vector):** Same as (a), but instead of the norm of the object type difference vector, each object type difference is used as a separate feature.

**(c) distance + diff_object_types (norm) + additional features:** Same as (a), but with additional features (*diff_v_elong*, *diff_h_elong*, *diff_v_area*, *diff_h_area*, *volume_ratio*).

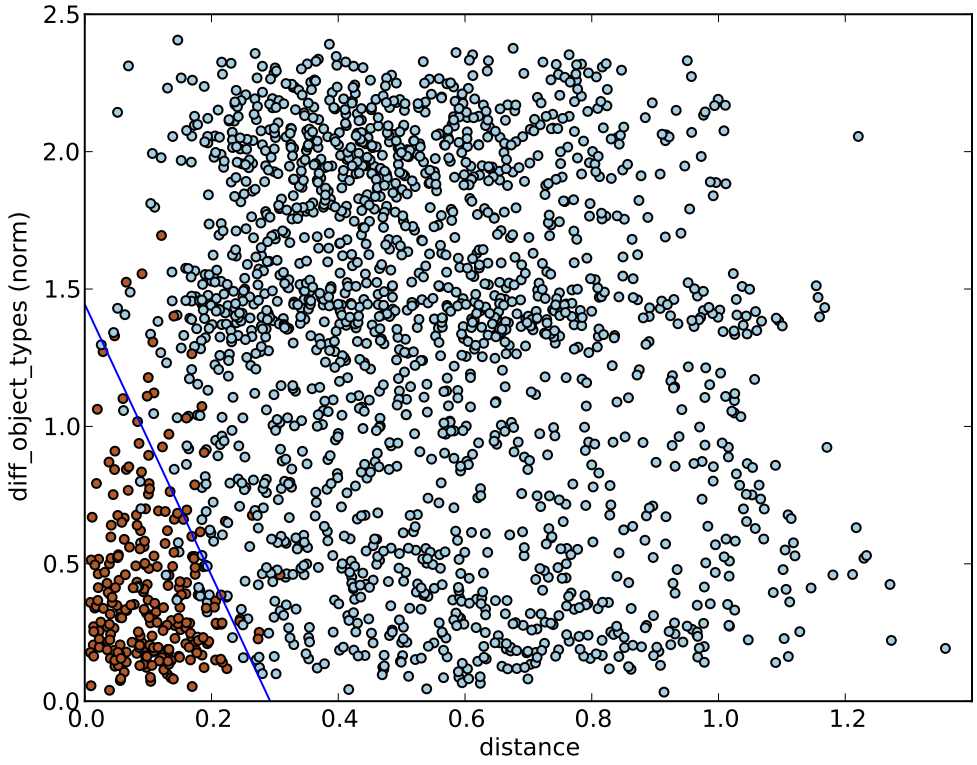



**Figure 4.10:** Distribution of samples in the transformed dataset. Red dots: samples labeled "same object", blue dots: samples labeled "different object". The only two similarity features used here are the euclidean distance between object centroids ($x$-axis) and the norm of the difference vector between the object types ($y$-axis). Blue line: decision boundary learned by the SVM.

**(d) distance + diff_object_types (vector) + additional features:** Same as (b), but with the additional features from (c).

The results are shown in Table 4.4. Even the smallest set of features (a) already yields good classification results ($F_1$ score: 0.97), which shows that the distance between objects and the aggregated difference of object types (a) is sufficient information for the classifier to decide whether two objects are the same. Providing additional information (b–d) does not further increase the classification performance significantly. Since simpler models with fewer parameters are more robust and less prone to overfitting, the simplest model (a) was chosen for the rest of this evaluation. To investigate whether a combination of features that is not included in a–d would yield a better result, Recursive Feature Elimination (RFE) was applied to the full feature set (c). During RFE, the features are removed one-by-one by eliminating the least informative feature. On some classification problems, this increases accuracy by reducing overfitting. RFE

**Table 4.4:** Classification reports of the SVM trained on different sets of features

| | precision | recall | $F_1$ score | support |
|---|---|---|---|---|
| *(a) distance + diff_object_types (norm)* | | | | |
| different object | 0.99 | 0.98 | 0.98 | 1816 |
| same object | 0.88 | 0.90 | 0.89 | 249 |
| avg / total | 0.97 | 0.97 | 0.97 | 2065 |
| *(b) distance + diff_object_types (vector)* | | | | |
| different object | 0.99 | 0.98 | 0.99 | 1816 |
| same object | 0.89 | 0.92 | 0.90 | 249 |
| avg / total | 0.98 | 0.98 | 0.98 | 2065 |
| *(c) distance + diff_object_types (norm) + additional features* | | | | |
| different object | 0.99 | 0.98 | 0.98 | 1816 |
| same object | 0.86 | 0.91 | 0.89 | 249 |
| avg / total | 0.97 | 0.97 | 0.97 | 2065 |
| *(d) distance + diff_object_types (vector) + additional features* | | | | |
| different object | 0.99 | 0.98 | 0.99 | 1816 |
| same object | 0.88 | 0.92 | 0.90 | 249 |
| avg / total | 0.98 | 0.98 | 0.98 | 2065 |

results in the following ranking of features by importance:

1. *distance*
2. *diff_object_types (norm)*
3. *diff_v_elong*
4. *diff_v_area*
5. *diff_h_area*
6. *volume_ratio*
7. *diff_h_elong*

None of the iteratively generated feature sets outperformed (a) significantly. This indeed confirms that the feature set consisting of *distance* and *diff_object_types (norm)* is the best choice.

**Table 4.5:** Decisions made by the data association process when using the SVM from Table 4.4(a). Row labels: Whether in the best assignment found by the Hungarian algorithm, the track ID and the assigned untracked anchor corresponded to *different objects* or the *same object* according to ground truth. Column labels: Whether the algorithm decided to create a new anchor for the track ID or whether to associate it to the assigned anchor, based on the similarity rating from the SVM. Correct decisions are highlighted in **bold**. Total accuracy is 94,12 %.

| | created new anchor | associated to existing anchor | total |
|---|---|---|---|
| different object | **28 (84,8 %)** | 5 (15,2 %) | 33 (100 %) |
| same object | 11 (4,6 %) | **228 (95,4 %)** | 239 (100 %) |

**Table 4.6:** Classification report of the data association process when using the SVM from Table 4.4(a). The data in this table corresponds to Table 4.5.

| | precision | recall | $F_1$ score | support |
|---|---|---|---|---|
| different object | 0.72 | 0.85 | 0.78 | 33 |
| same object | 0.98 | 0.95 | 0.97 | 239 |
| avg / total | 0.95 | 0.94 | 0.94 | 272 |

#### 4.3.3.2 Object–Anchor Data Association

To assess the performance of the complete object–anchor data association algorithm that combines the SVM trained in the previous subsection and the Hungarian algorithm, another experiment was performed. We ran the object–anchor data association algorithm (see Section 4.2.2.2) in its normal mode of operation on the untransformed dataset. For each frame, the Hungarian algorithm first found the globally best assignment between new track IDs and untracked anchors (as determined by the similarity value from the SVM). Next, for each assignment, a threshold on the similarity value was used to determine whether to create a new anchor for an object or whether to associate it with the assigned anchor. The results are shown in Table 4.5. In 256 out of 272 cases (94,12 %), the algorithm made the correct decision. These numbers do not include the trivial cases where no untracked anchors are present (such as at the beginning of each run of the algorithm), because in this case, the Hungarian algorithm is not needed; instead, a new anchor is created directly for each track ID. If these cases were included, the reported accuracy would be even higher.

Since the decisions reported in Table 4.5 ultimately depend on the classification made by the SVM, we also show the classification report of the data association process in Table 4.6, for easy comparison with the SVM results on the transformed dataset (Table 4.4). However, one should

note that the two tables are not directly comparable: Table 4.4 contains multiple entries for each new track ID (one for each possible assignment of the track ID to an untracked anchor), whereas Table 4.6 only has one entry for the best assignment. This results in two opposing effects on the classification accuracy. On the one hand, the Hungarian algorithm ensures that in those cases where the SVM would accept two or more possible matches, only the best match is counted, thereby increasing accuracy on the hard-to-decide cases near the class boundary and improving the accuracy of the category "same object". On the other hand, this step also eliminates all the easy-to-decide cases of different objects far away from the class boundary (the ones in the top right corner of Figure 4.10 on page 78), which results in lower reported accuracy for the class "different object". This is also evident in the fact that in Table 4.6, the support of class "different object" is much lower than in Table 4.4. Also, the transformed dataset used "perfect object association" (from ground truth), whereas, in the actual data association, errors can propagate: When a track ID is assigned to the wrong untracked anchor, that anchor is not available for the correct assignment later on. However, even given those interactions between the Hungarian algorithm and the SVM, the object–anchor data association still reaches an $F_1$ score of 0.94 (compared to 0.97 for the SVM alone on the transformed dataset), which demonstrates that our data association step is able to correctly assign almost all track IDs to the correct anchor.

### 4.3.4 Complete System

The collected dataset of table-top settings has two characteristics that are especially challenging for object recognition methods that use local appearance and the current sensor data alone, which further motivates the exploitation of contextual features and relations with objects outside the current sensor view. On the one hand, the small objects (fork, knife and spoon) only produce a small set of points in the captured point clouds (or pixels in the RGB-D images); they are almost the same size and contain reflective parts, so only the handle is actually visible. This combination makes them hard to be distinguished based only on local features. On the other hand, the robot normally sees only a part of the scene, which means that the full object context information is not available from the current sensor data (see Figure 4.7 on page 66).

Since an object is assigned a new track id when it leaves and then re-enters the robot's field of view, the number of tracks (387) is higher than the number of unique objects (144). The predicted category was determined by the class that was reported most often by the CRF for one particular track ID.

The results are summarized in Table 4.7. As expected, recognizing small objects yields poor

accuracy results when using only the local object recognition method. However, our combined system exploits context to achieve a significant improvement (29.22 % increase in accuracy). For the remaining six objects, the local object recognition method already delivers a high accuracy (91.15 %), so the improvement from exploiting context is not as large (2.0 % increase in accuracy). This shows that exploiting context is most useful in those cases where objects are ambiguous and hard to distinguish based on local appearance alone, but also improves accuracy in the other cases. The results also demonstrate the performance boost by using context with objects outside the current field of view (CRF+A, Section 4.2.2) and integrating local object recognition results into the CRF (CRF+Loc, Section 4.2.3). In total, the complete combined system (CRF+A+Loc) achieved an increase of 10.18 % in accuracy over the employed local object recognition method (Oliveira et al., 2014). Figure 4.11 shows the aggregated confusion matrix yielded by the proposed system on the dataset.

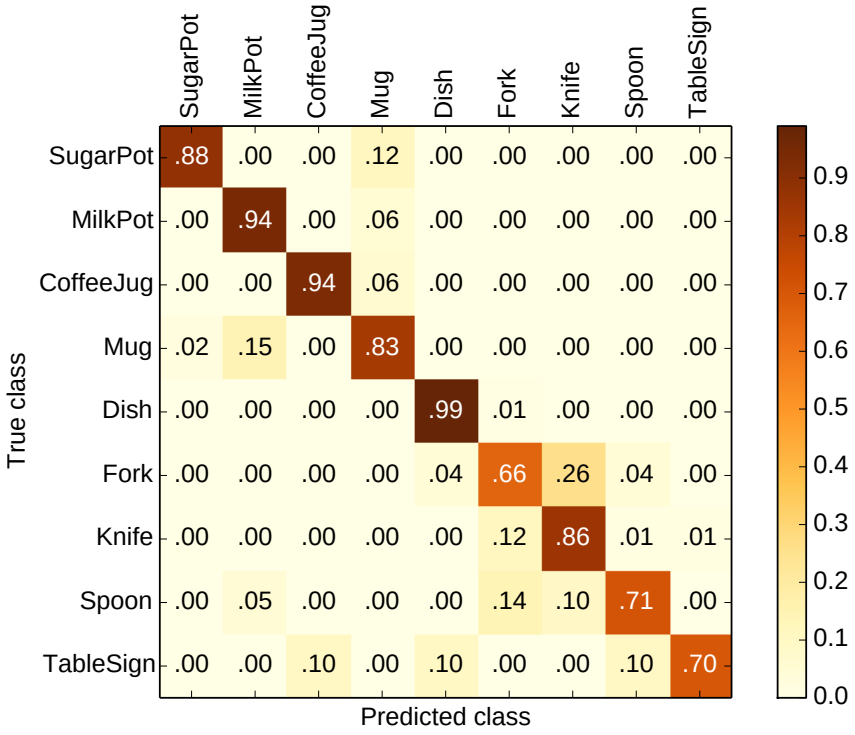



**Figure 4.11:** Confusion matrix of the combined method (CRF+A+Loc in Table 4.7) – classification accuracy: 86.82 %.

Notice that the achieved accuracy partially depends on the success of the local object recognition method, as mentioned in Section 4.2.1. Since this method could be replaced by any other

**Table 4.7:** Classification accuracy on sub-datasets. *Loc*: local object recognition only (base line; Oliveira et al., 2014); *CRF+A*: CRF only, but with objects outside the field of view supplied by Anchoring; *CRF+Loc*: CRF integrated with local object recognition; *CRF+A+Loc*: complete, proposed system (CRF, anchoring, local object recognition)

| | Loc | CRF+A | CRF+Loc | CRF+A+Loc |
|---|---|---|---|---|
| small objects (fork, knife, spoon) | 46.32 % | 75.00 % | 72.58 % | 75.54 % |
| other objects | 91.15 % | 90.73 % | 94.38 % | 93.15 % |
| total | 76.64 % | 85.05 % | 85.43 % | 86.82 % |

method providing the same output, the important thing here is not the absolute reported accuracy, but the fact that the exploitation of contextual relations largely improves its performance.

Regarding computational costs, training, which has to be completed only once, took on average 1421.5 s per fold. The runtime of feature extraction was 0.064 s and for MAP inference 0.005 s per scene graph (excluding time for the local object recognition). All experiments have been performed on a standard laptop (2.6 GHz Core i7 CPU, 8 GB RAM).

## 4.4 Discussion and Future Work

This chapter has presented an anchoring system that creates and maintains the correspondence between object symbols used by the higher-level modules in the RACE architecture, such as the planner, and the sensor data concerning these objects in the form of object recognition results from an arbitrary local object recognition method. Moreover, the anchoring system is able to exploit the contextual relations of the objects in the scene to improve the local object recognition results. To achieve this, we employ a probabilistic, context-aware representation of the environment based on Conditional Random Fields (CRF). These Probabilistic Graphical Models are typically used to model and exploit the relations in a given scene, but by using the world model created through anchoring, our system also considers nearby objects beyond the field of view of the sensor, which also improves its performance. With this combination of techniques, our system is capable of online operation, continually processing and integrating single frames, which is a requirement for online plan-based robot control.

The usefulness of the system has been demonstrated in the RACE project, where it was employed for anchoring object symbols used by the planner, execution monitor and human-robot interaction to object perceptions from 3D sensor data produced by an object recognition system.

To evaluate the performance of the proposed system, we collected a dataset of 15 scenes with table-top settings. This dataset is especially challenging for both local and contextual object recognition methods, so it results in a good testbed for the system. The conducted experiments assessed both the object association and object recognition capabilities of the system. For object association, the system achieved a precision of 0.95, a recall of 0.94, and an F1-score of 0.94. The full context-aware anchoring system achieved 86.82 % accuracy in object classification (compared to 76.64 % accuracy when using the local object recognition alone), which demonstrates that our system successfully exploits contextual relations between objects. The computational costs of the method were shown to be low (0.064 s for feature extraction and 0.005 s for CRF inference per scene graph).

In the future, we plan to employ our system to predict the most promising locations of non-perceived objects of a target class to support active object search, which will be covered in the next chapter.

# Chapter 5

# Forward-Looking Active Perception

Many robot tasks involve objects in the robot's environment. For example, in RACE's restaurant domain, the robot needs to bring objects such as a pepper mill or a cup to the guest's table. In a home service robot scenario, the robot might be instructed to bring a TV remote or other object to the owner. However, in a dynamic environment such as a restaurant or home, the robot does not always know about the exact location of the target object, which requires the robot to search for it.

This chapter presents the active perception system FLAP4CAOS (Forward-Looking Active Perception for Context-Aware Object Search). It controls the robot, based on sensor feedback, in order to autonomously explore an environment and actively search for target objects. In doing so, it exploits probabilistic information about the likely whereabouts of the target object and plans several steps ahead to minimize the expected time to find the object.

The anchoring system from Chapter 4 and the active perception system presented in this chapter are complementary. The anchoring system passively processes the sensor data generated by the robot, while the active perception system brings the robot's sensors into configurations that are most likely to contain the target object (see Figure 5.1). The two systems are largely independent: The anchoring system does not care whether the robot movements are generated by teleoperation, pre-planned sequences of actions or fully autonomously through the active perception system, and the active perception system does not care directly what happens with the sensor data that is being produced during its operation. This independence between the two systems allows to exchange each system independently of the other, depending

The contributions described in this chapter have first been published in the following publication: Gedicke et al. (2016)

on the task - for example, the active perception system could just as well be used to explore an environment in combination with the semantic mapping system from Chapter 3. Most of the high-level autonomous robot behavior emerges from the interplay between the two systems purely indirectly, via the sensor data. However, there are two more direct ways in which the two systems can interact: First, the anchoring system signals to the active perception system when the target object has been found, so the object search can be aborted. Second, the active perception system processes probabilistic knowledge about probable locations of the target object. This information can be supplied by a number of sources, for example, user input or background knowledge that is stored in a semantic map. Conceivably, this information could also come from the context probabilities computed by the anchoring system, thereby closing the loop; however, this is still future work.

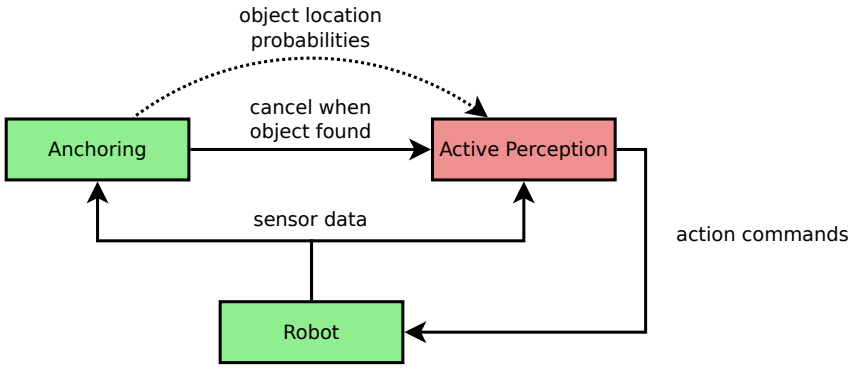



**Figure 5.1:** Simplified data flow between the robot, the anchoring system (Chapter 4) and the active perception system (this chapter). In the actual implementation, the anchoring and active perception systems do not interact directly, but via the executive and blackboard; this is omitted here for clarity.

The main contribution of this chapter is an object search and exploration algorithm that has the following distinctive features:

- it can exploit prior knowledge about probable object locations,
- it includes an efficient anytime 3D view sampling technique for calculating the next best views (Section 5.2.1), and
- it plans multiple steps ahead in order to minimize the expected time to find the target object (Section 5.2.2).

The system has been demonstrated on several different robots and has been thoroughly evaluated qualitatively and quantitatively (Section 5.3).

## 5.1 State of the Art

This section begins with reviewing the state of the art related to the main features of our algorithm: prior knowledge in object search (Section 5.1.1), 3D view sampling (Section 5.1.2) and planning ahead (Section 5.1.3). Afterward, the state of the art in fields that are adjacent to object search is reviewed: exploration and coverage path planning (Section 5.1.4) and continuous exploration trajectories (Section 5.1.5).

### 5.1.1 Object Search Exploiting Prior Knowledge

An important aspect of object search is the creation of a prior hypothesis regarding probable locations of the target object so that the search process can be directed towards promising locations. Ye and Tsotsos (1999) model this using a probability distribution function for a single target object that assigns to each cell in a discretized workspace the probability of the target being located there. Shubina and Tsotsos (2010) revisit and build upon this. Several approaches have been examined that allow robots to obtain and access the required kind of world knowledge. These include probabilistic environment models (Kunze et al., 2012), indirect search using spatial relations to intermediate targets (Anand et al., 2013; Kunze et al., 2014b), relational affordances (Moldovan and De Raedt, 2014), visual saliency (Rasouli and Tsotsos, 2014), semantic maps based on probabilistic programming languages (Veiga et al., 2016), probabilistic conceptual maps (Hanheide et al., 2011) and general semantic knowledge of indoor spaces (Aydemir et al., 2011, 2013).

### 5.1.2 Next Best Views

A typical trait of collision-free active perception systems is a phase during program execution that determines the *next best view*, i.e., the next sensor configuration that the robot should assume to optimize some utility function. This function incorporates a measure of the expected information gain and often also the predicted action cost. Ye and Tsotsos (1999) and Shubina and Tsotsos (2010) use a concept called sensed sphere to identify the visible area for each candidate of a discrete set of camera poses and calculate the expected probability of detecting the target at that pose. A cost function then combines this probability with the expected execution time to generate a final score. Sensing actions may also be guided by targeting *frontiers*, i.e., the boundary between free and unknown space. This concept was proposed by Yamauchi (1997) for 2D and later extended by Surmann et al. (2003) to 3D. Blodow et al. (2011) combine the size

of visible frontier regions with other factors in a cost function to evaluate a discretized pose space to find the next best view. As part of a multi-level process, they cast rays towards frontiers to precisely evaluate promising candidates in three dimensions, as Surmann et al. (2003) have done in a 2D projection of 3D space. A similar approach is pursued by Dornhege and Kleiner (2013), where rays are cast from unknown regions towards their associated frontiers as a method to explore enclosed areas with poor visibility. The original deterministic form of this approach is modified in later research by Dornhege et al. (2016) to employ sampling. Atanasov et al. (2014) present a system for classification and pose estimation of an object of interest in a cluttered scene. The system employs a detailed probabilistic model of the quality of a given view that takes degrees of occlusion into account and plans a sequence of views using a POMDP approach. The result outperforms popular greedy viewpoint selection approaches. McGreavy et al. (2016) developed a Next Best View planning approach for object recognition that into account under which angles an object is best recognized. Possible views are evaluated based on the proportion of the camera's field of view that is filled by the object, taking occlusions and the camera pose relative to the object into account. However, the system requires a good initial 6DoF pose estimation of the candidate object (or object hypotheses, in case of uncertainty about the object class).

### 5.1.3 Planning Ahead

Most of the above-cited next best view selecting systems follow a greedy approach in the way that only a single sensing action is planned ahead of time. The objective in the works by Ye and Tsotsos (1999) and Shubina and Tsotsos (2010) is to maximize the probability of finding the object within a given time constraint, and the greedy approach is shown to deliver good results. The workspace exploration system described by Renton et al. (1999) recursively pushes future view poses on a stack to ensure that a region is explored before the sensor is moved inside. Sarmiento et al. (2003) examine a planning strategy with the goal to minimize the expected time until the target is found. They prove the planning problem to be NP-hard but develop a fast heuristic method that yields a good approximate solution. Dornhege et al. (2016) use an approximation of a set cover/TSP decomposition of the problem to speed up the planning process. Rasouli and Tsotsos (2016) experimentally compare three different search strategies: GSC (a variant of a greedy strategy), EGS and DLAS (two planning strategies). In their experiments, the greedy strategy outperformed the other two; however, the initial plans were calculated without any knowledge of the environment, and in contrast to our method, replanning was

not done continually, but only when an action could not be performed (for example, due to obstacles), probably due to the high cost of replanning in their system. On the other hand, for the greedy variant, only the next action was chosen at each time, so it could incorporate knowledge gathered during the search in its action selection. This might have biased the results in favor of the greedy variant.

### 5.1.4 3D Exploration and Coverage Path Planning

Closely related to the task of object search is the task of exploration, in particular 3D exploration (Dang et al., 2018; Dornhege et al., 2016; Renton et al., 1999; Shade and Newman, 2011; Surmann et al., 2003). While the objectives of these two tasks differ, it can be argued that exploration is a special case of object search: The only differences are that first, exploration is not guided by prior probabilities of a target object's location, but will treat the whole target region equally; and second, exploration only terminates after the search space is exhausted. For this reason, the system presented here can also be used for the exploration of a 3D volume by providing it with a uniform probability distribution for the (non-existent) target object, and never canceling the search before it has exhausted the search space. The 3D map of the environment is not known to our algorithm in advance, since we assume the presence of dynamic obstacles that can cause occlusions, and so our algorithm always performs a limited form of exploration during object search. However, the 2D map is assumed to remain static and used by our algorithm for path planning. For our algorithm to perform a full exploration of an unknown environment, it would have to be extended to perform frontier-based exploration of the 2D map as well.

Another field related to Object Search is that of Coverage Path Planning (CPP; Galceran and Carreras, 2013). In CPP, the task is to find a motion plan that covers a given area or volume with a tool (e.g., in cleaning robots) or with a sensor's field of view (e.g., in inspection robots). The difference between Object Search and CPP is that Object Search minimizes the time to find a target object, whereas CPP minimizes the time to full coverage of an environment.

While the majority of CPP algorithms only address planar 2D environments, the approach by Englot and Hover (2012) – which was later extended by Papadopoulos et al. (2013) – performs a full 3D inspection of a complex 3D structure. The target application is autonomous ship hull inspection, where the whole surface of the hull (including the complex parts of the running gear) has to be covered by a sonar mounted to a 6DoF UAV. Given an a-priori 3D model of the hull, they employ random sampling-based path planning to construct an off-line plan which solves the coverage path planning problem.

Exploration Coverage Path Planning combines exploration and path planning; in other words, no a-priori map of the environment is required. Jia et al. (2016) present a CPP approach that performs exploration in a 2D environment with a simplified sensor model. Similar to FLAP4CAOS, the Receding Horizon Next-Best-View (RH-NBV) planner (Bircher et al., 2016, 2018) computes a random tree of possible view poses that is limited by a receding horizon and executed in a continuous replanning loop. By switching out the objective function for evaluating the quality of the nodes in the random tree, RH-NBV can be used either for the exploration of an unknown volume or coverage planning of a given surface manifold.

### 5.1.5 Continuous Exploration Trajectories

A drawback of the methods presented above (and of FLAP4CAOS as well) is that they employ random sampling to limit the complexity of the planning problem. This means that the sensor orientation is only evaluated at discrete view poses, and sensor readings that occur on the path between view poses are not considered in the planning process. FLAP4CAOS attempts to still take advantage of those sensor readings by keeping the camera turned towards the center of the target volume while moving, and continuously updating the 3D map with all received sensor data (even if it was not planned for). However, this is only a heuristic; optimal behavior might include turning the camera away from the center of the target volume while moving.

An exploration system that does not share the trait of explicit evaluation and selection of a next best view pose is presented by Shade and Newman (2011), where a potential field method inspired by fluid dynamics is used to generate smooth exploration trajectories directed towards frontiers. Charrow et al. (2015) use a two-stage approach, where they first use low-frequency global planning combined with local motion primitives to generate an exploration trajectory. In a second stage, they run trajectory optimization at a higher frequency to maximize an information-theoretic objective function, leading to exploration trajectories that point the robot's sensor towards unexplored space.

## 5.2 Algorithm and Implementation

Our system searches for an object of a target class (e.g., a mug) within a given target region. It is designed as a sub-component of a cognitive architecture that is capable of recognizing individual objects when they are detected by the sensor and maintains a belief state regarding probable object locations. When the object search component is triggered by the system, it is


**Figure 5.2:** PR2 during a search process. The target volume on top of the table has several occlusions and a cavity. Figure reproduced from Gedicke et al. (2016).

given a set of bounding volumes (e.g., a box that encompasses the top of the table shown in Figure 5.2) and the degree of belief for each of these volumes to contain an object of the target class. The actions available to the robot are navigating to a new position (using a path planner), pointing the head, and raising or lowering the telescoping torso, and any combination of these may be used to transition between view poses. The search process is terminated when the target is detected within the sensor's field of view.

Figure 5.3 shows an overview of the data flow between the involved components and highlights the active perception components that make up the FLAP4CAOS system in a darker shade. A 3D map is continually built from sensor data and is used by a view sampling module, which computes a set of possible view poses along with the expected information gain for each pose. These samples are per request fed to an object search planning and execution module, which aims to minimize the expected time until the target is found and iterates between requesting view samples, planning, moving the robot, and sensing.

Figure 5.4 shows a flowchart of the algorithm. At the beginning of each iteration, a set of

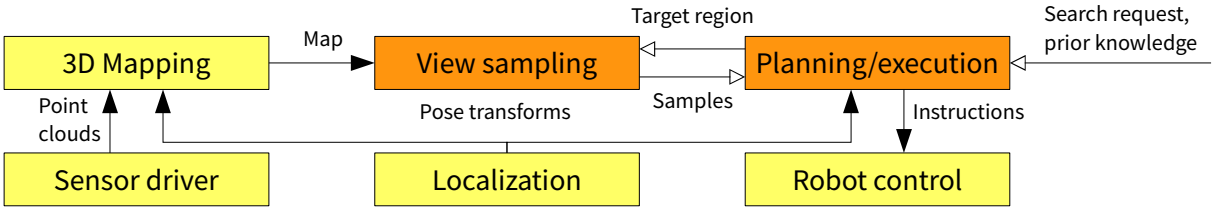



**Figure 5.3:** Structural overview of inter-component data flow. Figure reproduced from Gedicke et al. (2016).

view poses is sampled (Section 5.2.1). If none of the sampled views observes an unexplored part of the target region of interest, the algorithm terminates. Otherwise, three steps are performed (Section 5.2.2): First, pairwise transition times between views are calculated as a prerequisite for view planning. Next, a sequence of views is planned that minimizes the expected time to find the object. Finally, the camera is moved to the first view pose in that sequence by sending commands to the robot navigation and commanding other joints on the robot such as a telescoping spine or pan-tilt head. While moving, all incoming sensor information is integrated into a 3D occupancy map. If the object was found or the probability of finding the object in the remaining unobserved part of the region of interest falls below a certain threshold, the algorithm terminates; otherwise, all steps are repeated in the next iteration, incorporating the updated information in the 3D map.

### 5.2.1 Mapping and View Sampling

The system maintains an octree representation of the robot's workspace with cells that take values of either free, occupied, or unknown. The OctoMap library (Hornung et al., 2013) provides a mapping framework that allows to efficiently query the map and to integrate point clouds using a probabilistic sensor model. We maintain a set of frontier cells that consists of all unknown cells with at least one known free neighbor cell, akin to Blodow and colleagues' (2011) "fringe voxels". The map is built incrementally by continual integration of preprocessed sensor data.

Arbitrary regions of the map can be reset to unknown to express that the region is assumed to have changed and should be observed anew. This can be triggered manually or by external modules that use semantic knowledge to deduce that the information regarding a certain region is out of date based on the observation of some situation or activity. Future work may implement a version of fading knowledge that gradually decreases confidence in individual cells when they have not been observed for a certain amount of time.

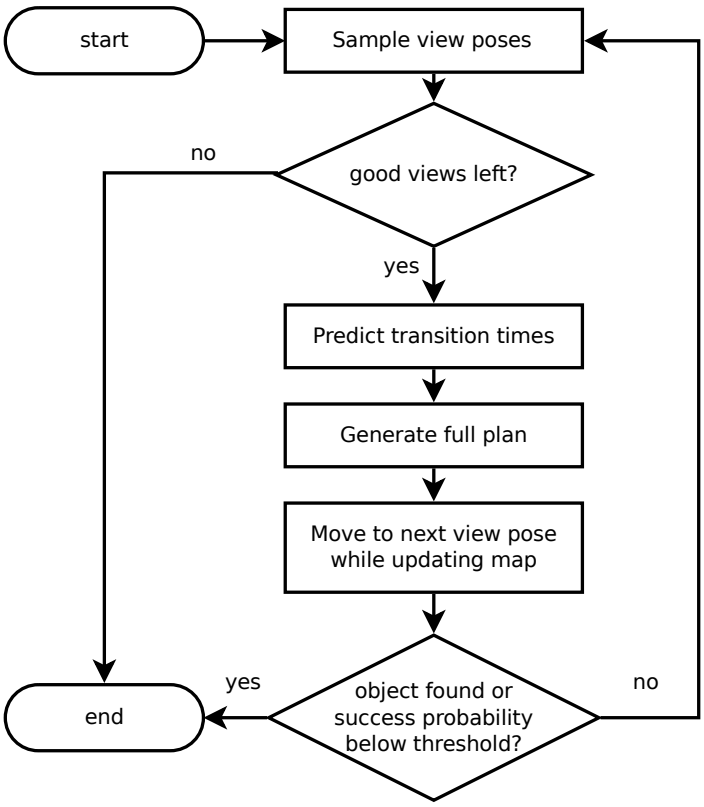



**Figure 5.4:** Flowchart of the main phases of the algorithm

The map constitutes the basis of the view sampling process that determines and evaluates possible sensor poses for the observation of unknown cells within some region of interest. View sampling is an anytime operation that allows to adjust the amount of computational effort invested on the fly and, given enough time, will find all feasible view poses. The sampling system's output consists of 2-tuples, wherein the first element is a sensor pose and the second element is a set of unknown cells that are expected to be visible from this pose. In the following, such pairs shall be called *views*.

A query to the sampling system consists of a region of interest (ROI), defined as a set of bounding volumes $\mathcal{V}$. For each volume $V_i \in \mathcal{V}$, the set of all frontier cells inside $V_i$ is gathered. Unknown cells at the boundary of $V_i$ that are not marked as frontiers are additionally included as frontiers for this volume only to enable the system to find views for regions even if they are enclosed in unknown space. The union of all collected frontier cells within the ROI is in the following denoted as $\mathcal{F}$.

The position of each view sample is picked from a constrained space around a randomly

chosen target frontier cell $f \in \mathcal{F}$. All views are oriented to point directly towards their respective target cell. View poses are randomly chosen such that they obey the following three general constraints. The first constraint simply demands the position to be within the sensor's range limits relative to the target cell. Secondly, the sensor pose must be attainable according to a configurable model of the robot's kinematics; for example, the sensor's height is limited to certain bounds for robots with a telescoping spine, and also the pitch angle is variable within limits for robots with a pan-tilt head. Finally, all potential camera poses should be located at the known-space side of the frontier, looking towards the ROI. This constraint is enforced by estimating the frontier plane's normal vector at $f$ and restricting the angle between the line of sight and the frontier normal to be less than 90°. The frontier normal of a target cell is calculated by connecting the mean of the center points of all unknown neighbors with the mean of the center points of all free neighbors. While a straightforward implementation could simply pick random coordinates and reject those samples which lie outside the constrained region, we reduce the number of rejects significantly using a local consistency approach in the above-outlined constraint network.

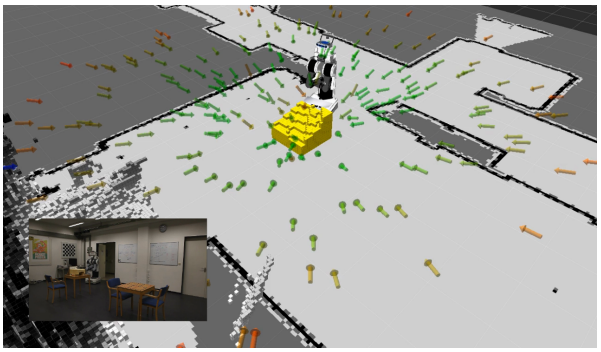
To determine the set of cells that are expected to be revealed by the sensor at a sample pose, we scan the field of view of a virtual camera and compile the set of all unknown cells within the ROI that are traversed by rays. When a ray hits an unknown cell, it is allowed to pass through; hence, view sampling assumes unknown space to be free and is, in this sense, optimistic.

The sampling procedure loops until either the required number of views is generated or a timeout is reached. See Figure 5.5 and Figure 5.6 for visualizations of the generated view samples.

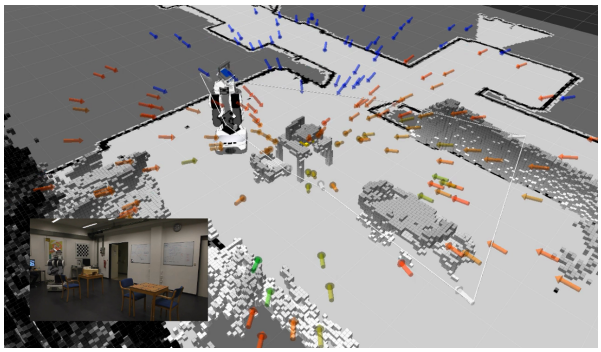
### 5.2.2 Planning and Execution

The object search component acts on search tasks that define a target object and a search region and runs until either the target is located or the search region is exhausted. A planner constructs sequences of views with the aim to minimize the expected duration of a search process that successively visits all views in the sequence.

The search region defined in each task is represented as a set of bounding volumes $\mathcal{V}$. Each volume $V_i \in \mathcal{V}$ is annotated with a probability $P_i$ of the volume containing *at least one* object of the target type according to the robot's current state of belief. Since all $P_i$ are independent of each other, it is not necessary that $\sum P_i = 1$. Each cell $c_k \in V_i$ is assumed to equally and

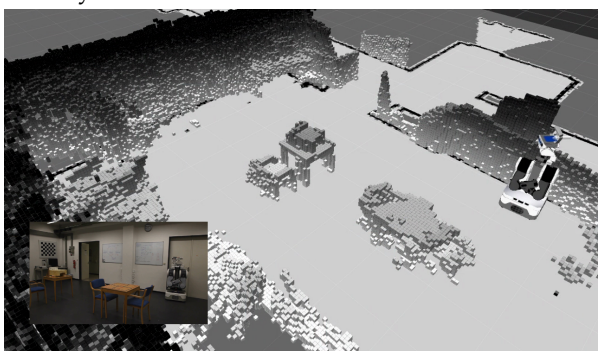


**(a)** Initial set of views. Part of the ROI was already observed by the robot in its initial position.



**(b)** After the first few actions, most of the ROI is already observed.

**(c)** When the robot reaches the other side of the table, only a few unobserved voxels inside the cavity remain unobserved.



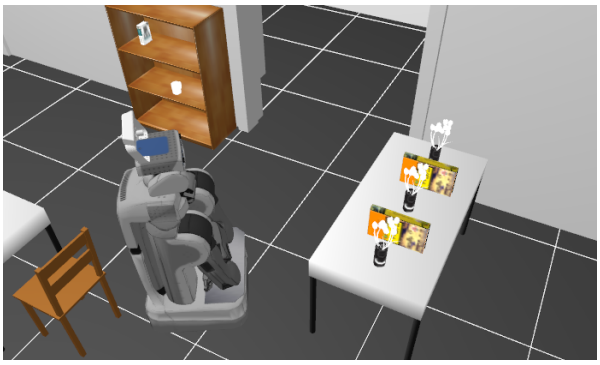
**(d)** Finally, the robot drives as far as possible away from the table while remaining within the sensor's maximum range and lowers its torso completely to look inside the cavity.

**Figure 5.5:** Object search execution on a real PR2 robot. Views are visualized as arrows with color-coded normalized target space visibility (blue for zero visibility). Unknown cells within the ROI on the table are marked yellow; occupied cells in the environment are marked in gray. The planned path is shown in white.

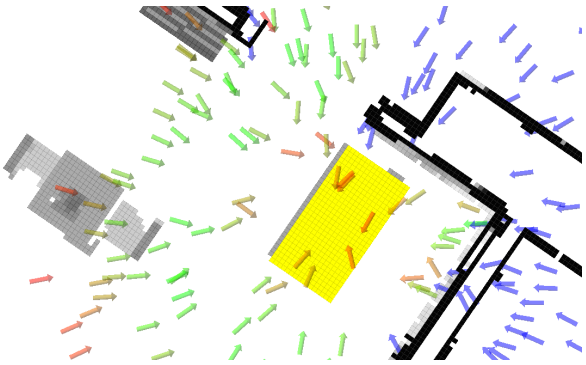
independently contribute to $P_i$, resulting in a probability value for each cell of

$$\forall c_k \in V_i : p(c_k) = \begin{cases} 1 - \sqrt[N_i]{1 - P_i} & \text{if } \mathit{unknown}(c_k) \\ 0 & \text{else}, \end{cases} \tag{5.1}$$

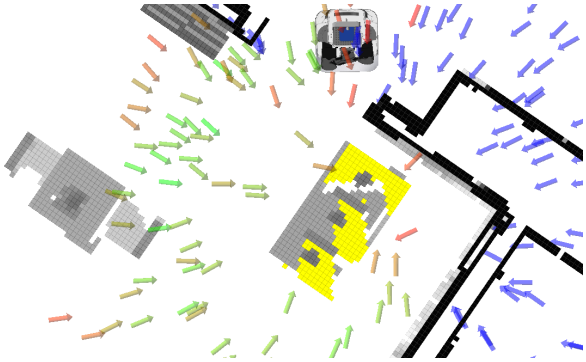
where $N_i$ is the number of cells in $V_i$. Comparable object search systems, such as those of Ye and Tsotsos (1999) and Shubina and Tsotsos (2010), model the probability distribution of a single target object's location. In such a model, the sum of all location probabilities necessarily equals 1, and all probabilities must be updated accordingly when a set of cells is seen without locating the target. Because our approach makes no assumption about the number of target



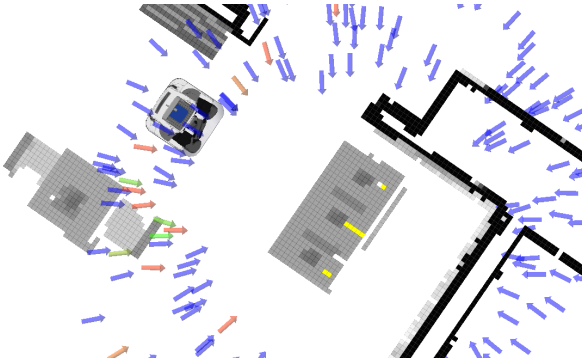
**(a)** Overview of the setting.



**(b)** Initial samples.



**(c)** Unknown cells after partially observing the ROI. The unknown cells closest to the robot are below the sensor's field of vision.



**(d)** After observation from a second angle, few occluded cells remain.

**Figure 5.6:** View sampling during different stages of object search in a Gazebo simulation. Views are visualized as arrows with color-coded normalized target space visibility (blue for zero visibility). Unknown cells within the ROI on the table are marked yellow. Note: We show an orthographic projection; all parts of the algorithm work in 3D. Figure reproduced from Gedicke et al. (2016).

objects, all $P_i$ and all $p(c_k)$ can be considered independent and no update step is necessary.

View sampling is called for the region of interest to generate a set of $n$ views $\mathcal{S}$. The returned views include only unknown cells so that the total detectable space according to the view set is

$$\hat{V}(\mathcal{S}) = \bigcup_{s_i \in \mathcal{S}} \mathit{cells}(s_i) \subseteq \mathcal{V}\,, \tag{5.2}$$

where $\mathit{cells}(s_i)$ yields the set of all cells expected to be seen from the view pose $s_i$. We disregard the reliability of detection algorithms and assume that if a target is in view of the sensor, it will be detected every time. The probability of detecting at least one instance of the target type by moving the sensor to a view pose $s_i$ can be defined in terms of the set of cells seen $C = \mathit{cells}(s_i)$

as

$$p(C) = 1 - \prod_{c_i \in C} (1 - p(c_i)) \,. \tag{5.3}$$

The total expected probability to see a target using all views is simply $p(\hat{V}(\mathcal{S}))$. The object search task contains a termination criterion in the form of a threshold probability $\omega$. The search process terminates when $p(\hat{V}(\mathcal{S})) < \omega$.

An external path planner is used to generate paths from the current robot location to all view locations. All unreachable views are filtered out. Subsequently, predicted transition times $t(s_i, s_j)$ between the remaining views are calculated. Since these times are needed in every step of the planning process, a lot of time is saved by pre-computing them as a lookup table. Further time is saved by clustering close-by view poses together to reduce the quadratic number of path planning requests necessary. Transition times comprise the driving time (predicted using a linear regression over path costs returned by the planner), the time needed to lift the robot's torso to achieve different sensor heights, the time needed to move the pan/tilt head, and a fixed amount of sensing time at the target pose.

Let a sequence of views be denoted as $L = \langle L_0, \ldots, L_n \rangle$, where $L_0$ is the current state and each $L_k$ has a corresponding view $s_{L,k} \in \mathcal{S}$. The goal is to find a sequence $\hat{L}$ that minimizes the expected time until the target is found which is given by

$$E[T|L] = \sum_{i=0}^{n-1} \prod_{k=0}^{i} \big(1 - p(s_{L,k} | \langle L_0, \ldots, L_{k-1} \rangle)\big) t(L_i, L_{i+1}) \,, \tag{5.4}$$

where $p(s_i|L)$ is the probability to find a target with observation $s_i$ after having already seen the sequence $L$. Accounting for all cells already seen by $L$, this is calculated as

$$p(s_i|L) = p\left( cells(s_i) \setminus \bigcup_{k=0}^{n} cells(L_k) \right) \,. \tag{5.5}$$

Sarmiento et al. (2003) examine this optimization problem in a similar context and employ a heuristic algorithm called *utility greedy* to quickly compute approximate solutions. Our planner uses the key heuristics of utility greedy in a depth-first branch and bound approach to construct plans without consuming too much memory. Since the objective function is the plan's expected runtime, the obvious greedy heuristic to determine the order in which new views $s_i \in \mathcal{S}$ are appended to partial plans is to minimize the local increase in expected runtime. However, this performs poorly, confirming the observation made by Sarmiento et al. who propose to improve

efficiency using a different utility function given by

$$util(s_i|L) = \frac{p(s_i|L)}{t(L_n, s_i)}\,. \tag{5.6}$$

Since this function proves to be effective, we adopt it for our planner[1]. To further accelerate planning, Sarmiento et al. sacrifice optimality and introduce a pruning heuristic that discards *strictly dominated* views $s_k$ given a partial plan $L$ according to

$$dominated(s_k|L) :\Leftrightarrow \exists s_i \in \mathcal{S} : p(s_i|L) > p(s_k|L) \wedge t(L_n, s_i) < t(L_n, s_k)\,. \tag{5.7}$$

In other words, a view is discarded if another view exists that is both faster to reach and more likely to reveal the target. During the discussion of the experimental results (Section 5.3), we will use the flag $\xi \in \{0, 1\}$ to denote whether this pruning heuristic was enabled.

We introduce two additional mechanisms to reduce the amount of branching in the planner. First, branching is only allowed up to a *depth limit* $\psi$. After the depth limit is reached, the plan is completed by greedily picking remaining views with the highest utility value (5.6) until the termination criterion (5.9) is reached. Second, only expansions are allowed for which the utility value is not smaller than the best utility value divided by a *branch limit* factor $\phi$. As the value of $\phi$ increases, more branching takes place, since fewer views are discarded. These parameters allow balancing the optimality of the planner versus speed. At one extreme, for $\psi = \infty$, $\phi = \infty$ and $\xi = 0$, the full search space will be exhausted, guaranteeing optimality. At the other extreme, for $\psi = 0$ or $\phi = 1$, the planner constructs a single greedy sequence without branching.

As evidenced by (5.4) to (5.7), the planner needs to perform set operations to track a plan's quality and to evaluate heuristic functions. It should be noted that, although (5.5) is formulated in terms of set union, the planner is implemented using only set difference, resulting in faster set operations with increasing planning depth. Since the number of search nodes increases exponentially with depth, this reformulation results in a huge total speedup.

The total probability of success for a plan can be calculated analogously to (5.3) as

$$p(L) = 1 - \prod_{L_i \in L} \left(1 - p(L_i|\langle L_0, \ldots, L_{i-1}\rangle)\right). \tag{5.8}$$

[1]See Sarmiento et al. (2003) for an analysis of the relation between the utility function and the objective function and why the naive utility function performs poorly.

A plan is complete when the remaining probability to encounter a target by adding more view poses falls below the termination threshold $\omega$, as expressed by

$$complete(L|\mathcal{S}) :\Leftrightarrow p(L) > 1 - \frac{1 - p(\hat{V}(\mathcal{S}))}{1 - \omega}\,. \tag{5.9}$$

After planning finishes, the robot navigates to move the sensor into the first planned position, senses and waits for sensor data integration. If the search process is not externally interrupted, that means that no target object has yet been found, and the procedure starts another cycle. Each cycle retains the previously planned views, which are reevaluated and augmented with additional views using the view sampling system. A new plan is constructed continually in each cycle until $p(\hat{V}(\mathcal{S})) < \omega$.

## 5.3 Results

The FLAP4CAOS system has been tested in reality on a variety of robots, including the Willow Garage PR2 (Figure 5.2 on page 91), Osnabrück University's Calvin robot (Figure 2.6 on page 24) and a team of three Turtlebots (Figure 5.7). This demonstrates that FLAP4CAOS can easily be deployed on a range of different robot kinematics and also already has limited support for multi-robot collaboration; for a more detailed discussion, see Section 5.4.

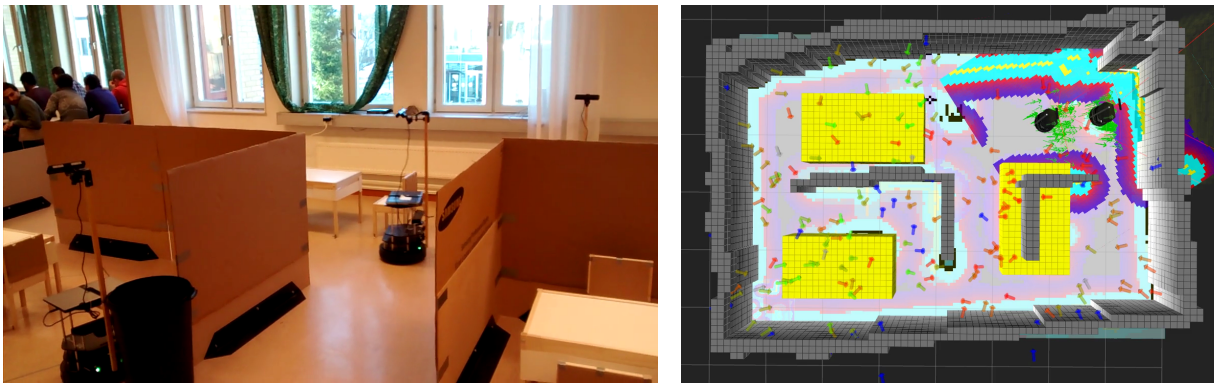


**Figure 5.7:** A team of Turtlebots using FLAP4CAOS to collaboratively explore an environment during the Lucia Winter School on Artificial Intelligence and Robotics in Örebro, Sweden (December 11-16, 2016).

The remainder of this section presents the results of a series of quantitative experiments in simulation. The robot used in these experiments is a Willow Garage PR2 with a head-mounted RGB-D camera which can be panned 350° and tilted 115°. The head is mounted on a telescoping

torso with a lift range of 31 cm. The real-world functionality of the active perception system has been shown as part of the final demonstration of the RACE project[2]. We evaluated the system in detail using the Gazebo simulator with a simplified robot model that retains the physical limitations of the real version. Robot motion is replaced by updating poses directly in the simulation with simulated transition times that are obtained from the same model that is used by the planner to predict execution times. Robot localization is obtained directly from the simulated ground truth. This intentionally eliminates any influence that imperfect robot localization and navigation, as well as an inaccurate execution time model, may have on object search performance. Figure 5.8 shows an overview of the simulated environment used in all experiments. Run times of all algorithms were measured on an Intel Core i7 870 CPU.

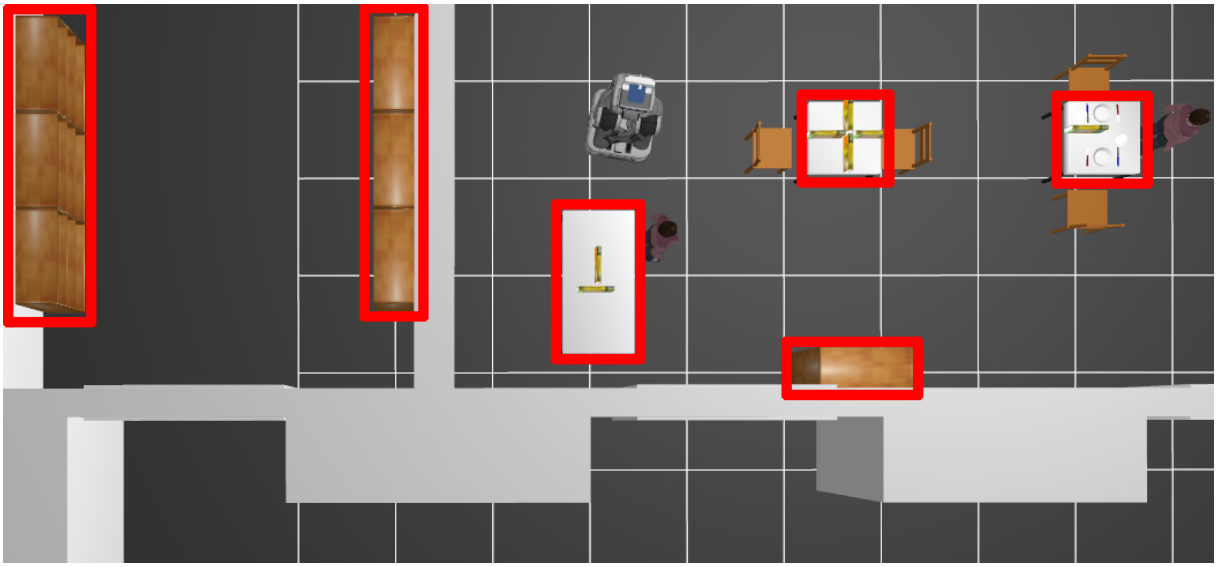


**Figure 5.8:** Top-down view of the simulated environment used in the experiments, consisting of two connected rooms with various target volumes (red boxes: two shelf arrangements, one single shelf, three cluttered tables). Figure reproduced from Gedicke et al. (2016).

We ran three series of experiments evaluating several planner configurations, and, while computing the optimal solution was not feasible, compared the performance of the planning algorithm with a greedy approach as a baseline. It should be noted that instead of implementing a naïve greedy approach, where the view with the highest probability of observing the target would be picked, our baseline greedy approach instead uses the same utility function as the planner (5.6), which also takes the time to reach the view pose into account. The only

[2]A video of the system running on the PR2 can be found at `http://kos.informatik.uos.de/flap4caos/`

**Table 5.1:** Average algorithm run times. Table reproduced from Gedicke et al. (2016).

| | |
|---|---|
| View sampling (200 views) | $2.4 \pm 0.1$ s |
| Transition time prediction (50 clusters) | $6.4 \pm 0.6$ s |

difference between the baseline greedy approach and the planner is that the greedy approach does not perform tree search. The task in all experiments is to locate a target within the volumes shown in Figure 5.8 with a probability of $P_i = 0.2$ for each volume to contain at least one instance of the target. Each experiment comprises 20 trials starting with a random initial robot pose and ending when the remaining probability of detecting the target falls below $\omega = 0.05$. To measure object search efficiency, we determine the mean expected value of the time needed to complete search tasks as the sum of an execution time component (motion and sensing), and a planning time component[3].

We generate 200 view samples for each trial, sufficient for almost complete coverage of our test environment, and share them between all planner configurations to enable a direct comparison. Accordingly, a lookup table for predicted transition times between views is generated once per trial and shared between all planner configurations. Table 5.1 shows the average computation times for these operations over all experiments. The total probability to find a target is determined by the structure of the environment, the set of available view poses, the choice of target volumes, and their target encounter probabilities, and is hence equal for all planner configurations within each trial. In the first two experiments, we test offline and continual planning given a full 3D map of the environment, thereby enabling our planner to fully optimize the search process from the start. In the third experiment, we test continual planning without an initial map (only walls are known in advance), so that the system is forced to adapt the search strategy on the go.

In total, 112 different planner configurations (i.e., combinations of $\psi$, $\phi$ and $\xi$) were tested on each of the 20 trials (37 configurations for the first experiment, 48 for the second and 27 for the third). Figure 5.9 shows the results for the best-performing configuration in each experiment; for the full results, see Gedicke (2015). As shown in Figure 5.9, the object search planner outperforms the greedy strategy in all experiments with regard to the expected execution time of the resulting plan as well as the combined expected run time of the search process (i.e., planning time plus execution time). In case a map is available beforehand, continual planning achieves

[3]See (5.4) for the calculation of expected values

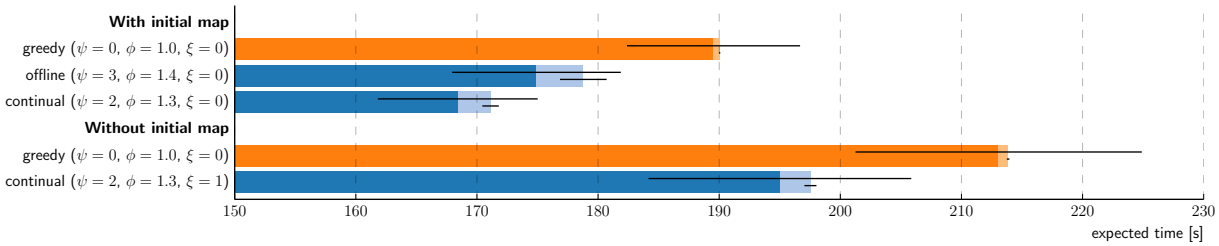



**Figure 5.9:** Mean expected execution times (dark) and mean expected total planning times (bright) for the greedy strategy vs. the best performing planner configuration in each experiment. $\psi$: depth limit; $\phi$: branch limit factor; $\xi$: use pruning heuristic (5.7). Note: The x-axis is truncated and starts at 150 s. Figure reproduced from Gedicke et al. (2016).

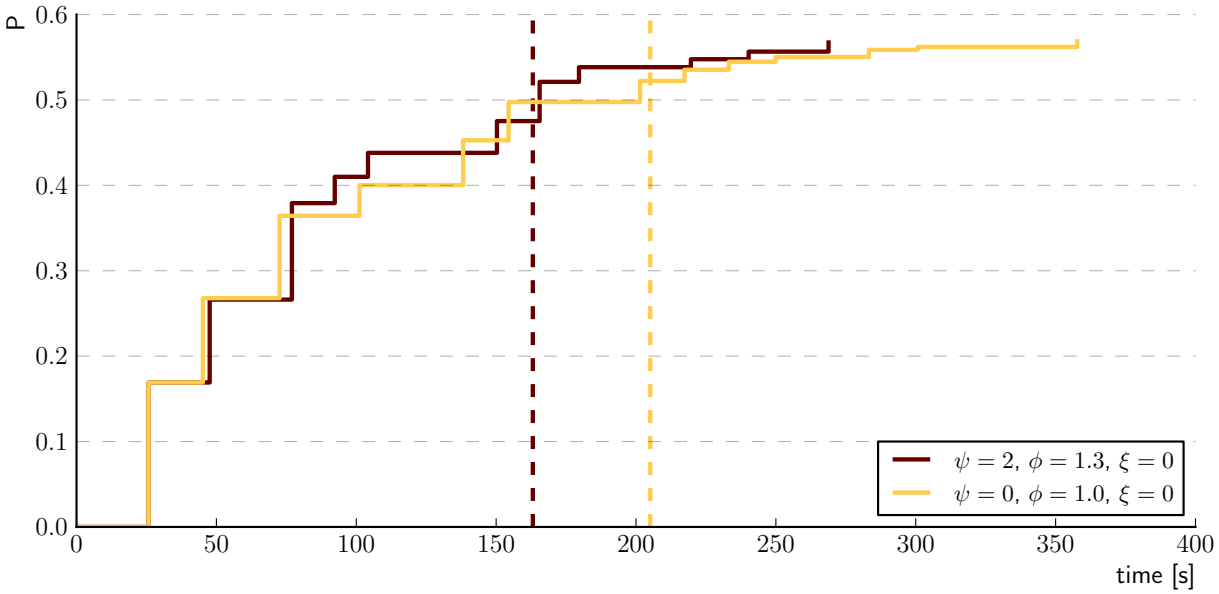



**Figure 5.10:** Cumulative probability of success over time for one run of greedy search (yellow) vs. continual planning (brown) with initial map. Vertical steps: sensing actions; horizontal steps: pose transitions; dashed lines: expected execution times. $\psi$: depth limit; $\phi$: branch limit factor; $\xi$: use pruning heuristic (5.7). Figure reproduced from Gedicke et al. (2016).

run times that are on average 10 % shorter compared to greedy, and it compares favorably to offline planning, which is remarkable considering the computational cost of repeated planning. This is due to the fact that sensor data is continuously integrated into the map while the robot is moving between planned view poses, as discussed in Sections 5.1.5 and 5.2.1. Since the continual planner can react to these additional "free" observations, it generally has to move to fewer view poses than the offline planner. The reactivity also means that the optimal parameters for continual planning turned out to be stricter depth limit $\psi$ and branch limit factor $\phi$ than the optimal parameters for offline planning, which reduces the total planning time despite the additional overhead of repeated planning.

While the advantage of planning decreases when no initial map is available, we still achieve a 7 % reduction of average search time using knowledge acquired during the ongoing search process itself. A visualization of search progress during a single task is shown in Figure 5.10. The planning algorithm achieves coverage of the search region using fewer sensing steps (11 vs. 13) and also tends to reduce average transition times between sensing poses. Although the planning algorithm yields a shorter total search time, the simple greedy strategy is still a viable alternative with the advantages of being parameter-free and computationally cheap. The relative competitiveness of greedy strategies is not uncommon in related work as reviewed in Section 5.1.

## 5.4 Discussion and Future Work

This chapter has presented a plan-based active perception system for object search. The results demonstrate that although a greedy baseline approach already performs very well in this domain, planning several steps results in a faster total time for completing the object search task, even taking the additional time required for planning into account.

It is part of future work to examine how appropriate planner parameterizations change with different search tasks and working environments. The speed of plan generation may be less important in applications with different cost functions, e.g., energy consumption or risk of failure, which our planner can be adapted for by choosing appropriate parameters with enlarged planning scope.

As stated in Section 5.2.2, the current approach disregards issues of object recognition. Future developments will have to address the fact that objects in real applications cannot be recognized with full reliability and may be only partially visible due to occlusion or field of view limits. Additionally, some unexplored volumes could be excluded from the search simply be-

cause they are too small to contain the target object. It is also worthwhile to explore biasing the view sampling towards candidates that are more likely to yield reliable detections similar to Atanasov et al. (2014) or McGreavy et al. (2016).

Our experiments used a simple model of target location probabilities and manually selected target volumes. In principle, both can come from arbitrary sources (see Section 5.1) and can be updated during the search process. A source for target object location probabilities is available with the probabilistic anchoring module (Chapter 4), based on a probabilistic model of spatial relations. Future work will investigate the integration of both systems.

Another possible direction of future research is multi-robot collaboration. While initial experiments were performed with a team of three Turtlebots (Figure 5.7), there was very limited collaboration. Each robot was running its own instance of FLAP4CAOS while sharing the generated 3D map between them. This means that while the observations of the other robots were taken into account as they occurred, they were not considered by the planner beforehand, leaving room for future improvements.

Lastly, the view sampling module exploits the assumption that the pose of the robot's base is fully determined by the camera pose. This means that the kinematic model currently supports robots in which the camera is rigidly linked to the robot's base (such as the Turtlebots or the Calvin robot) or which have a telescoping spine or pan-tilt head (such as the PR2 or Tiago robots). Most UAVs can also already be used with FLAP4CAOS if a suitable 3D navigation system is provided. However, for eye-in-hand configurations, where the camera is fixed to a robot arm's end-effector (such as DFKI's Mobipick robot), the kinematic model becomes much more involved. Obviously, the robot's movements can be restricted to conform to the existing kinematic model, but to fully exploit the robot's range of movements, a motion planner needs to be used, which will drastically increase the computation time. Finding a solution to this problem is a topic of ongoing work.

# Chapter 6

# Conclusions

One of the challenges of plan-based robot control is that it needs a world model of the task-relevant aspects of the robot's environment. In unstructured, every-day environments, such a world model has to be actively created and continually updated based on noisy sensor data. The symbols used in high-level planning and reasoning have to be anchored to objects and structures detected in the environment. To do this effectively, robot perception cannot be treated as a passive bottom-up process alone. Exploiting top-down knowledge and context can be useful to complete the robot's interpretation of sensor data and guide further exploration.

In this dissertation, we have presented three contributions that tackle the challenges of building a world model:

First, a *model-based semantic mapping* system was presented that recognizes larger-scale structures like furniture based on semantic descriptions in an ontology. Here, ontological knowledge about the composition of compound objects was be used to recognize larger-scale structures from their parts.

Second, a *context-aware anchoring* process was presented that creates and maintains the links between object symbols and the sensor data corresponding to those objects while exploiting the geometric context of objects. Here, knowledge about the spatial context of an object and common relations to other objects was exploited to improve the quality of the world model.

Third, an *active perception* system was presented that plans several steps ahead to actively search for a required object that is missing from the world model. Here, knowledge about the probable locations of the object from other parts of the system can be used.

Possible avenues for future research in each separate system have already been discussed at the end of each chapter. In the following, we will therefore focus on the ongoing work and the

bigger picture. The work that was started in this dissertation is currently being continued in the ongoing project "CoPDA" (Comprehensive Perception and Dynamic Anchoring; Günther et al., 2020). In this project, several enhancements of the anchoring and active perception systems presented here are planned:

**Multi-Hypothesis Tracking (MHT)** The probabilistic anchoring module will track multiple hypotheses that are continually updated with new observations and model predictions, which avoids committing too early to the most likely hypothesis when there is insufficient evidence for either hypothesis.

**ASP-based Anchoring** The probabilistic anchoring module will work in tandem with a logic anchoring module based on Answer Set Programming (ASP). The logic anchoring module exploits knowledge about the evolution of objects and their dynamic attributes for anchoring.

**Knowledge-based Prediction** A physics-based model for prediction will be complemented by a prediction module based on common sense knowledge and domain knowledge to allow for more precise anchoring.

**Active Perception for Object Recognition** The active perception system will be enhanced to bias view selection towards views that are more likely to produce good object recognition and pose estimation results.

**Closer integration of anchoring and active perception** The anchoring system will trigger the active perception system for lost or missing objects of interest and provide a probability distribution of likely object locations, based on its current world model and domain knowledge.

Looking at the bigger picture, we hope that this dissertation can be a stepping stone towards comprehensive scene understanding and long-term autonomy.

# Bibliography

Marina Alberti, John Folkesson, and Patric Jensfelt. Relational approaches for joint object classification and scene similarity measurement in indoor environments. In *AAAI Spring Symposium, Qualitative Representations for Robots*, Palo Alto, California, March 2014. (Cited on pages 43 and 46.)

Sven Albrecht, Thomas Wiemann, Martin Günther, and Joachim Hertzberg. Matching CAD object models in semantic mapping. In *Proc. ICRA 2011 workshop: Semantic Perception, Mapping and Exploration, SPME '11*, Shanghai, China, 2011. (Cited on page 25.)

Oscar Alonso-Ramirez, Antonio Marin-Hernandez, Homero V. Rios-Figueroa, Michel Devy, Saul E. Pomares-Hernandez, and Ericka J. Rechy-Ramirez. A graph representation composed of geometrical components for household furniture detection by autonomous mobile robots. *Applied Sciences*, 8(11), 2018. doi: 10.3390/app8112234. (Cited on page 28.)

Rares Ambrus, Nils Bore, John Folkesson, and Patric Jensfelt. Meta-rooms: Building and maintaining long term spatial models in a dynamic world. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 1854–1861. IEEE, 2014. doi: 10.1109/IROS.2014.6942806. (Cited on page 28.)

Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *Int. J. Rob. Res.*, 32(1):19–34, 2013. doi: 10.1177/0278364912461538. (Cited on pages 43, 45, 68, and 87.)

Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, George J. Pappas, and Kostas Daniilidis. Nonmyopic view planning for active object classification and pose estimation. *IEEE Transactions On Robotics*, 30(5):1078–1090, 2014. doi: 10.1109/TRO.2014.2320795. (Cited on pages 88 and 104.)

Alper Aydemir, Kristoffer Sjöö, John Folkesson, Andrzej Pronobis, and Patric Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China, May 2011. (Cited on page 87.)

Alper Aydemir, Andrzej Pronobis, Moritz Göbelbecker, and Patric Jensfelt. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, 2013. doi: 10.1109/TRO.2013.2256686. (Cited on page 87.)

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. ISBN 0-521-78176-0. (Cited on page 31.)

Yaakov Bar-Shalom, Peter K. Willett, and Xin Tian. *Tracking and Data Fusion – A Handbook of Algorithms*. YBS Publishing, 2011. ISBN 9780964831278. (Cited on page 57.)

Michael Beetz, Ferenc Balint-Benczedi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer, and Zoltan-Csaba Marton. RoboSherlock: Unstructured information processing for robot perception. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 1549–1556. IEEE, 2015. doi: 10.1109/ICRA.2015.7139395. (Cited on page 50.)

J. Besag. On the statistical analysis of dirty pictures. *Royal Statistical Society*, Series B(2):259–302, 1986. (Cited on page 70.)

P. Besl and N. McKay. A method for registration of 3–D shapes. *IEEE T. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992. (Cited on page 33.)

Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon "next-best-view" planner for 3D exploration. In Danica Kragic, Antonio Bicchi, and Alessandro De Luca, editors, *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pages 1462–1468. IEEE, 2016. doi: 10.1109/ICRA.2016.7487281. (Cited on page 90.)

Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon path planning for 3D exploration and surface inspection. *Auton. Robots*, 42(2):291–306, 2018. doi: 10.1007/s10514-016-9610-0. URL `https://doi.org/10.1007/s10514-016-9610-0`. (Cited on page 90.)

Nico Blodow. *Managing Belief States for Service Robots*. Dissertation, Technische Universität München, München, 2014. (Cited on page 50.)

Nico Blodow, Dominik Jain, Zoltan-Csaba Marton, and Michael Beetz. Perception and probabilistic anchoring for dynamic world state logging. In *Humanoids*, pages 160–166. IEEE, 2010. ISBN 978-1-4244-8688-5. doi: 10.1109/ICHR.2010.5686341. (Cited on page 49.)

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 *IROS*, pages 4263–4270. IEEE, 2011. ISBN 978-1-61284-454-1. doi: 10.1109/IROS.2011.6094665. (Cited on pages 87 and 92.)

H. A. P. Blom and E. A. Bloem. Interacting multiple model joint probabilistic data association, avoiding track coalescence. In *Proc. IEEE Conf. Decision Control*, volume 3, pages 3408–3415, December 2002. (Cited on page 58.)

Sebastian Blumenthal, Herman Bruyninckx, Walter Nowak, and Erwin Prassler. A scene graph based shared 3D world model for robotic applications. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 453–460. IEEE, 2013. doi: 10.1109/ICRA.2013.6630614. (Cited on page 50.)

Jonathan Bohren, Radu Bogdan Rusu, E. Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the PR2. In *ICRA*, pages 5568–5575. IEEE, 2011. doi: 10.1109/ICRA.2011.5980058. (Cited on page 13.)

Claus Brenner, Jan Böhm, and Jens Gühring. CAD-based object recognition for a sensor/actor measurement robot. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives)*, volume XXXII-5, pages 209–216, Hakodate, Japan, 1998. ISPRS. (Cited on page 28.)

Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3D mapping. In Lydia E. Kavraki, David Hsu, and Jonas Buchli, editors, *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015. doi: 10.15607/RSS.2015.XI.003. URL `http://www.roboticsproceedings.org/rss11/p03.html`. (Cited on page 90.)

Matei T. Ciocarlie, Kaijen Hsiao, Edward Gil Jones, Sachin Chitta, Radu Bogdan Rusu, and Ioan Alexandru Sucan. Towards reliable grasping and manipulation in household environments. In Oussama Khatib, Vijay Kumar, and Gaurav S. Sukhatme, editors, *Experimental Robotics - The 12$^{th}$ International Symposium on Experimental Robotics, ISER 2010, December 18-21, 2010, New Delhi and Agra, India*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 241–252. Springer, 2010. doi: 10.1007/978-3-642-28572-1_17. (Cited on pages 47, 48, 52, and 75.)

Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robot. Auton. Syst.*, 43(2-3):85–96, 2003. (Cited on pages 41, 49, and 53.)

Silvia Coradeschi, Amy Loutfi, and Britta Wrede. A short review of symbol grounding in robotic and intelligent systems. *KI*, 27(2):129–136, 2013. doi: 10.1007/s13218-013-0247-2. (Cited on page 50.)

Tung Dang, Christos Papachristos, and Kostas Alexis. Autonomous exploration and simultaneous object search using aerial robots. In *2018 IEEE Aerospace Conference*, Big Sky, MT, USA, March 2018. doi: 10.1109/AERO.2018.8396632. (Cited on page 89.)

José de Gea Fernández, Dennis Mronga, Martin Günther, Tobias Knobloch, Malte Wirkus, Martin Schröer, Mathias Trampler, Stefan Stiene, Elsa Kirchner, Vinzenz Bargsten, Timo Bänziger, Johannes Teiwes, Thomas Krüger, and Frank Kirchner. Multimodal sensor-based whole-body control for human-robot collaboration in industrial settings. *Robot. Auton. Syst.*, 94: 102–119, August 2017a. doi: 10.1016/j.robot.2017.04.007. 2017. (Cited on page 1.)

José de Gea Fernández, Dennis Mronga, Martin Günther, Malte Wirkus, Martin Schröer, Stefan Stiene, Elsa Kirchner, Vinzenz Bargsten, Timo Bänziger, Johannes Teiwes, Thomas Krüger, and Frank Kirchner. iMRK: Demonstrator for intelligent and intuitive human–robot collaboration in industrial manufacturing. *KI - Künstliche Intelligenz*, 31(2):203–207, May 2017b. doi: 10.1007/s13218-016-0481-5. URL `http://rdcu.be/ogar`. (Cited on page 1.)

Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. W3C recommendation, W3C, February 2004. URL `http://www.w3.org/TR/owl-ref/`. (Cited on page 31.)

Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous

exploration in 3D. *Adv. Robotics*, 27(6):459–468, 2013. ISSN 0169-1864. doi: 10.1080/01691864.2013.763720. (Cited on page 88.)

Christian Dornhege, Alexander Kleiner, Andreas Hertle, and Andreas Kolling. Multirobot coverage search in three dimensions. *J. Field Robot.*, 33(4):537–558, 2016. ISSN 1556-4967. doi: 10.1002/rob.21573. (Cited on pages 88 and 89.)

Ivan Dryanovski, Roberto G. Valenti, and Jizhong Xiao. Fast visual odometry and mapping from RGB-D data. In *Proc. ICRA-2013*, Karlsruhe, Germany, 2013. (Cited on page 25.)

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 2650–2658, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.304. (Cited on page 47.)

Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin A. Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust RGB-D object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 681–687. IEEE, 2015. doi: 10.1109/IROS.2015.7353446. (Cited on page 47.)

Jos Elfring, S. van den Dries, M. J. G. van de Molengraft, and Maarten Steinbuch. Semantic world modeling using probabilistic multiple hypothesis anchoring. *Robot. Auton. Syst.*, 61 (2):95–105, 2013. doi: 10.1016/j.robot.2012.11.005. (Cited on page 50.)

F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, MA, USA, May 2012. (Cited on page 25.)

Brendan Englot and Franz S. Hover. Sampling-based coverage path planning for inspection of complex structures. In *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'12, pages 29–37. AAAI Press, 2012. (Cited on page 89.)

Michael Esterman, Benjamin J. Tamber-Rosenau, Yu-Chin Chiu, and Steven Yantis. Avoiding non-independence in fMRI data analysis: Leave one subject out. *NeuroImage*, 50(2):572–576, 2010. doi: 10.1016/j.neuroimage.2009.10.092. (Cited on page 76.)

Thomas Fäulhammer, Michael Zillich, Johann Prankl, and Markus Vincze. A multi-modal RGB-D object recognizer. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 733–738. IEEE, 2016. doi: 10.1109/ICPR.2016.7899722. (Cited on pages 47 and 52.)

Matthias Fichtner. *Anchoring symbols to percepts in the fluent calculus*. PhD thesis, Dresden University of Technology, 2009. (Cited on page 49.)

Matthias Fichtner. Anchoring symbols to percepts in the fluent calculus - A general approach to the symbol anchoring problem of cognitive robots. *KI*, 25(1):77–80, 2011. doi: 10.1007/s13218-010-0051-1. (Cited on page 49.)

Michael Firman. RGBD datasets: Past, present and future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*, pages 661–673. IEEE Computer Society, 2016. doi: 10.1109/CVPRW.2016.88. (Cited on page 73.)

Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013. doi: 10.1016/j.robot.2013.09.004. URL `https://doi.org/10.1016/j.robot.2013.09.004`. (Cited on page 89.)

Cipriano Galindo, Alessandro Saffiotti, Silvia Coradeschi, P. Buschka, J. A. Fernández-Madrigal, and J. González. Multi-hierarchical semantic maps for mobile robotics. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3492–3497, Edmonton, CA, 2005. (Cited on page 27.)

Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Comput. Vis. Image Underst.*, 114(6):712–722, June 2010. ISSN 1077-3142. doi: 10.1016/j.cviu.2010.02.004. (Cited on pages 42 and 47.)

Carolina Galleguillos, Andrew Rabinovich, and Serge J. Belongie. Object categorization using co-occurrence, location and appearance. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008. doi: 10.1109/CVPR.2008.4587799. (Cited on pages 43 and 47.)

Thorsten Gedicke. FLAP for CAOS: Forward-looking active perception for clutter-aware object search. Master's thesis, Osnabrück University, May 2015. (Cited on pages 8 and 101.)

Thorsten Gedicke, Martin Günther, and Joachim Hertzberg. FLAP for CAOS: Forward-looking active perception for clutter-aware object search. In *Proc. 9$^{th}$ IFAC Symposium on Intelligent*

*Autonomous Vehicles (IAV)*, volume 49 (15) of *IFAC-PapersOnLine*, pages 114–119, Leipzig, Germany, June 2016. IFAC. doi: 10.1016/j.ifacol.2016.07.718. (Cited on pages 85, 91, 92, 96, 100, 101, and 102.)

Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Model-based object recognition from 3D laser data. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings*, volume 7006 of *Lecture Notes in Computer Science*, pages 99–110. Springer, 2011. doi: 10.1007/978-3-642-24455-1_9. (Cited on pages 25 and 30.)

Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Building semantic object maps from sparse and noisy 3D data. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 2228–2233. IEEE, 2013. doi: 10.1109/IROS.2013.6696668. (Cited on pages 24, 25, and 32.)

Martin Günther, Thomas Wiemann, Sven Albrecht, and Joachim Hertzberg. Model-based furniture recognition for building semantic object maps. *Artif. Intell.*, 247:336–351, June 2017. doi: 10.1016/j.artint.2014.12.007. Available online: Jan 23, 2014. (Cited on pages 25, 30, 33, 35, 37, 39, and 40.)

Martin Günther, José Raúl Ruiz-Sarmiento, Cipriano Galindo, Javier González-Jiménez, and Joachim Hertzberg. Context-aware 3D object anchoring for mobile robots. *Robot. Auton. Syst.*, 110:12–32, December 2018. doi: 10.1016/j.robot.2018.08.016. (Cited on pages 8 and 41.)

Martin Günther, Friedemann Kammler, Oliver Ferdinand, Joachim Hertzberg, Oliver Thomas, and Oliver Zielinski. Automatische Wiedererkennung von individuellen Objekten mit dem Dynamic Anchoring Agent. *HMD Prax. Wirtsch.*, 57:1173–1186, October 2020. doi: 10.1365/s40702-020-00675-y. (Cited on page 106.)

Marc Hanheide, Charles Gretton, Richard Dearden, Nick Hawes, Jeremy Wyatt, Andrzej Pronobis, Alper Aydemir, Moritz Göbelbecker, and Hendrik Zender. Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. International Joint Conference on Artificial Intelligence (IJCAI-11), July 16-22, Barcelona, Spain*. IJCAI, July 2011. (Cited on page 87.)

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth

cameras for dense 3D modeling of indoor environments. *Int. J. Rob. Res.*, 31(5):647–663, April 2012. (Cited on page 25.)

A. Hermans, G. Floros, and B. Leibe. Dense 3D semantic mapping of indoor scenes from RGB-D images. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2631–2638, May 2014. doi: 10.1109/ICRA.2014.6907236. (Cited on pages 28 and 48.)

Joachim Hertzberg, Jianwei Zhang, Liwei Zhang, Sebastian Rockel, Bernd Neumann, Jos Lehmann, Krishna S. R. Dubba, Anthony G. Cohn, Alessandro Saffiotti, Federico Pecora, Masoumeh Mansouri, Štefan Konečný, Martin Günther, Sebastian Stock, Luís Seabra Lopes, Miguel Oliveira, Gi Hyun Lim, Hamidreza Kasaei, Vahid Mokhtari, Lothar Hotz, and Wilfried Bohlken. The RACE project. *KI - Künstliche Intelligenz*, 28(4):297–304, 2014. ISSN 0933-1875. doi: 10.1007/s13218-014-0327-y. (Cited on pages 12 and 20.)

Javier Hidalgo Carrió, Sascha Arnold, Arne Böckmann, Anna Born, Raúl Domínguez, Daniel Hennes, Christoph Hertzberg, Janosch Machowinski, Jakob Schwendner, Yong-Ho Yoo, and Frank Kirchner. EnviRe - environment representation for long-term autonomy. In *ICRA Workshop on AI for Long-term Autonomy*, Stockholm, May 2016. (Cited on page 50.)

Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. *Int. J. Comput. Vision*, 80(1):3–15, 2008. doi: 10.1007/s11263-008-0137-5. (Cited on pages 43 and 47.)

Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.*, 34(3):189–206, 2013. doi: 10.1007/s10514-012-9321-0. (Cited on page 92.)

Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C member submission, World Wide Web Consortium, 2004. URL `http://www.w3.org/Submission/SWRL`. (Cited on page 31.)

International Federation of Robotics. Executive summary world robotics service robots 2020. 2020. URL `https://ifr.org/free-downloads/`. (Cited on page 1.)

Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3D object dataset: Putting the Kinect to work. In Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige, editors, *Consumer Depth Cameras for Computer Vision, Research Topics and Applications*, Ad-

vances in Computer Vision and Pattern Recognition, pages 141–165. Springer, 2013. doi: 10.1007/978-1-4471-4640-7_8. (Cited on page 75.)

Dan Jia, Martin Wermelinger, Remo Diethelm, Philipp Krüsi, and Marco Hutter. Coverage path planning for legged robots in unknown environments. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2016, Lausanne, Switzerland, October 23-27, 2016*, pages 68–73. IEEE, 2016. doi: 10.1109/SSRR.2016.7784279. URL `https://doi.org/10.1109/SSRR.2016.7784279`. (Cited on page 90.)

S. Hamidreza Kasaei, Miguel Oliveira, Gi Hyun Lim, Luís Seabra Lopes, and Ana Maria Tomé. Interactive open-ended learning for 3D object recognition: An approach and experiments. *J. Intell. Robot. Syst.*, 2015. doi: 10.1007/s10846-015-0189-z. (Cited on pages 14, 47, 48, and 51.)

Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. *CoRR*, abs/1607.06038, 2016. (Cited on pages 47 and 52.)

Christian Kerl, Jörg Stückler, and Daniel Cremers. Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2264–2272. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.261. (Cited on page 25.)

Salman Hameed Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Geometry driven semantic labeling of indoor scenes. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 679–694, Zurich, Switzerland, September 2014. Springer International Publishing. ISBN 978-3-319-10590-1. (Cited on pages 47 and 48.)

Ulrich Klank, Dejan Pangercic, Radu Bogdan Rusu, and Michael Beetz. Real-time CAD model matching for mobile manipulation and grasping. In *9th IEEE-RAS Intl. Conf. on Humanoid Robots*, Paris, France, December 7-10 2009. (Cited on page 28.)

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. (Cited on pages 44, 62, 64, and 65.)

Štefan Konečný, Sebastian Stock, Federico Pecora, and Alessandro Saffiotti. Planning domain + execution semantics: A way towards robust execution? In *Qualitative Representations for Robots, AAAI Spring Symposium*, Stanford, USA, March 2014. (Cited on page 13.)

Hema Swetha Koppula, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3D point clouds for indoor scenes. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *25th Annual Conference on Neural Information Processing Systems (NIPS-2011)*, pages 244–252, Granada, Spain, December 2011. (Cited on pages 43 and 45.)

Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robot. Auton. Syst.*, 66:86–103, 2015. doi: 10.1016/j.robot.2014.12.006. (Cited on page 27.)

Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6D pose estimation in RGB-D images. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 954–962, 2015. (Cited on pages 47 and 52.)

Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. PoseAgent: Budget-constrained 6D object pose estimation via reinforcement learning. In *2017 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, 2017. (Cited on pages 47 and 52.)

H. W. Kuhn. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.*, 2(1-2): 83–97, 1955. ISSN 1931-9193. doi: 10.1002/nav.3800020109. (Cited on pages 54 and 59.)

Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robot. Auton. Syst.*, 8(1-2):47–63, 1991. doi: 10.1016/0921-8890(91)90014-C. (Cited on page 26.)

L. Kunze, M. Beetz, M. Saito, H. Azuma, K. Okada, and M. Inaba. Searching objects in large-scale indoor environments: A decision-theoretic approach. In *ICRA*, pages 4385–4390, May 2012. doi: 10.1109/ICRA.2012.6224965. (Cited on page 87.)

Lars Kunze, Chris Burbridge, Marina Alberti, Akshaya Tippur, John Folkesson, Patric Jensfelt, and Nick Hawes. Combining top-down spatial reasoning and bottom-up object class recognition for scene understanding. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pages 2910–2915. IEEE, 2014a. doi: 10.1109/IROS.2014.6942963. (Cited on pages 43, 46, and 68.)

Lars Kunze, Keerthi Kumar Doreswamy, and Nick Hawes. Using qualitative spatial relations for indirect object search. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 163–168. IEEE, 2014b. doi: 10.1109/ICRA.2014.6906604. (Cited on page 87.)

Mathieu Labbé and François Michaud. Long-term online multi-session graph-based SPLAM with memory management. *Autonomous Robots*, 42(6):1133–1150, 2018. doi: 10.1007/s10514-017-9682-5. (Cited on page 25.)

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view RGB-D object dataset. In *IEEE International Conference on Robotics and Automation, ICRA2011, Shanghai, China, 9-13 May 2011*, pages 1817–1824. IEEE, 2011. doi: 10.1109/ICRA.2011.5980382. (Cited on page 47.)

John J. Leonard and Richard J. Rikoski. *Experimental Robotics VII*, chapter Incorporation of Delayed Decision Making into Stochastic Mapping, pages 533–542. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-45118-1. doi: 10.1007/3-540-45118-8_53. (Cited on page 57.)

Lingni Ma, Jörg Stückler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with RGB-D cameras. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 598–605. IEEE, 2017. doi: 10.1109/IROS.2017.8202213. (Cited on pages 28, 48, and 49.)

Jharna Majumdar and A. G. Seethalakshmy. A CAD model based system for object recognition. *J. Intell. Robot. Syst.*, 18(4):351–365, April 1997. ISSN 0921-0296. doi: 10.1023/A:1007902728509. (Cited on page 28.)

Julian Mason and Bhaskara Marthi. An object-based semantic world model for long-term change detection and semantic querying. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 3851–3858, 2012. doi: 10.1109/IROS.2012.6385729. (Cited on page 28.)

John McCormac, Ankur Handa, Andrew J. Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, Singapore, May 2017. IEEE. doi: 10.1109/ICRA.2017.7989538. (Cited on pages 28 and 48.)

John McCormac, Ronald Clark, Michael Bloesch, Andrew J. Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level SLAM. In *2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018*, pages 32–41. IEEE Computer Society, 2018. doi: 10.1109/3DV.2018.00015. (Cited on page 28.)

Christopher McGreavy, Lars Kunze, and Nick Hawes. Next best view planning for object recognition in mobile robotics. In *34th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*, Huddersfield, United Kingdom, December 2016. (Cited on pages 88 and 104.)

Vahid Mokhtari, Luís Seabra Lopes, and Armando J. Pinho. Experience-based planning domains: an integrated learning and deliberation approach for intelligent robots - robot task learning from human instructions. *Journal of Intelligent and Robotic Systems*, 83(3-4):463–483, 2016. doi: 10.1007/s10846-016-0371-y. (Cited on page 14.)

Bogdan Moldovan and Luc De Raedt. Occluded object search by relational affordances. In *ICRA (IEEE International Conference on Robotics and Automation ), Hong Kong, May 31 - June 7, 2014*, June 2014. (Cited on page 87.)

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. *Robot. Autom. Mag.*, 18(2):22–32, June 2011. (Cited on page 28.)

Andreas C. Müller and Sven Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6237. IEEE, May 2014. (Cited on pages 47 and 48.)

Dana S. Nau, Héctor Muñoz-Avila, Yue Cao, Amnon Lotem, and Steven Mitchell. Total-order planning with partially ordered subtasks. In Bernhard Nebel, editor, *IJCAI*, pages 425–430. Morgan Kaufmann, 2001. ISBN 1-55860-777-3. (Cited on page 13.)

Bernd Neumann, Lothar Hotz, Pascal Rost, and Jos Lehmann. A robot waiter learning from experiences. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, pages 285–299, Cham, 2014. Springer International Publishing. (Cited on page 14.)

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W. Fitzgibbon.

KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136. IEEE, 2011. ISBN 978-1-4577-2183-0. doi: 10.1109/ISMAR.2011.6092378. (Cited on page 25.)

Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robot. Auton. Syst.*, 56(11):915–926, 2008. doi: 10.1016/j.robot.2008.08.001. (Cited on pages 26 and 27.)

Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. 6D SLAM – 3D mapping outdoor environments. *J. Field Robot.*, 24(8-9):699–722, 2007. doi: 10.1002/rob.20209. (Cited on page 34.)

Miguel Oliveira, Gi Hyun Lim, Luís Seabra Lopes, S. Hamidreza Kasaei, Ana Tomé, and Aneesh Chauhan. A perceptual memory system for grounding semantic representations in intelligent service robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, Illinois, 2014. IEEE. (Cited on pages 14, 47, 48, 51, 75, 82, and 83.)

Dejan Pangercic, Benjamin Pitzer, Moritz Tenorth, and Michael Beetz. Semantic object maps for robotic housework - representation, acquisition and use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 4644–4651, 2012. doi: 10.1109/IROS.2012.6385603. (Cited on page 27.)

Georgios Papadopoulos, Hanna Kurniawati, and Nicholas M. Patrikalakis. Asymptotically optimal inspection planning using systems with differential constraints. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pages 4126–4133. IEEE, 2013. doi: 10.1109/ICRA.2013.6631159. URL `https://doi.org/10.1109/ICRA.2013.6631159`. (Cited on page 89.)

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12: 2825–2830, 2011. (Cited on page 59.)

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009. (Cited on page 20.)

A. Rasouli and J.K. Tsotsos. Visual saliency improves autonomous visual search. In *Computer and Robot Vision (CRV)*, pages 111–118, May 2014. doi: 10.1109/CRV.2014.23. (Cited on page 87.)

Amir Rasouli and John K Tsotsos. Sensor planning for 3D visual search with task constraints. In *Computer and Robot Vision (CRV), 13th Conference on*, pages 37–44. IEEE, 2016. (Cited on page 88.)

D. Reid. An algorithm for tracking multiple targets. *IEEE Trans. Automat. Contr.*, 24:843–854, December 1979. (Cited on page 58.)

Xiaofeng Ren, Liefeng Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *CVPR*, pages 2759–2766, June 2012. doi: 10.1109/CVPR.2012.6247999. (Cited on pages 43, 47, and 48.)

Pamela Renton, Michael A. Greenspan, Hoda A. ElMaraghy, and Hassen Zghal. Plan-N-Scan: A robotic system for collision-free autonomous exploration and workspace mapping. *J. Intell. Robot. Syst.*, 24(3):207–234, 1999. ISSN 0921-0296. doi: 10.1023/A:1008090503603. (Cited on pages 88 and 89.)

Sebastian Rockel, Bernd Neumann, Jianwei Zhang, Krishna S. R Dubba, Anthony G. Cohn, Štefan Konečný, Masoumeh Mansouri, Federico Pecora, Alessandro Saffiotti, Martin Günther, Sebastian Stock, Joachim Hertzberg, Ana Maria Tomé, Armando J. Pinho, Luís Seabra Lopes, Stephanie von Riegen, and Lothar Hotz. An ontology-based multi-level robot architecture for learning from experiences. In *Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, Stanford, USA, March 2013. (Cited on pages 12 and 13.)

José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Mobile robot object recognition through the synergy of probabilistic graphical models and semantic knowledge. In *ECAI Workshop on Cognitive Robotics (CogRob)*, 2014. (Cited on pages 43 and 45.)

José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Exploiting semantic knowledge for robot object recognition. *Knowl.-Based Syst.*, 86:131–142, 2015a. doi: 10.1016/j.knosys.2015.05.032. (Cited on pages 43 and 45.)

José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. UPGMpp: a software library for contextual object recognition. In *3rd Workshop on Recognition and Action for Scene Understanding (REACTS)*, 2015b. (Cited on page 70.)

José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Robot@Home, a robotic dataset for semantic mapping of home environments. *I. J. Robotics Res.*, 36(2):131–141, 2017a. doi: 10.1177/0278364917695640. (Cited on page 75.)

José Raúl Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. Building multiversal semantic maps for mobile robot operation. *Knowl.-Based Syst.*, 119(Supplement C): 257–272, 2017b. ISSN 0950-7051. doi: 10.1016/j.knosys.2016.12.016. (Cited on page 50.)

José Raúl Ruiz-Sarmiento, Martin Günther, Cipriano Galindo, Javier González-Jiménez, and Joachim Hertzberg. Online context-based object recognition for mobile robots. In *17th International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, April 2017c. (Cited on page 41.)

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Emanuel Dolha, and Michael Beetz. Towards 3D point cloud based object maps for household environments. *Robot. Auton. Syst.*, 56(11):927–941, 2008. doi: 10.1016/j.robot.2008.08.005. (Cited on page 27.)

Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1352–1359. IEEE, 2013. (Cited on page 28.)

A. Sarmiento, R. Murrieta, and S.A. Hutchinson. An efficient strategy for rapidly finding an object in a polygonal world. In *IROS*, volume 2, pages 1153–1158, October 2003. doi: 10.1109/IROS.2003.1248801. (Cited on pages 88, 97, and 98.)

Max Schwarz, Anton Milan, Arul Selvam Periyasamy, and Sven Behnke. RGB-D object detection and semantic segmentation for autonomous manipulation in clutter. *I. J. Robotics Res.*, 0 (0):3–15, 2017. doi: 10.1177/0278364917713117. (Cited on page 47.)

scikit-learn documentation. Cross-validation: evaluating estimator performance, 2021. URL `http://scikit-learn.org/stable/modules/cross_validation.html`. (Cited on page 76.)

R. Shade and P. Newman. Choosing where to go: Complete 3D exploration with stereo. In *ICRA*, pages 2806–2811, May 2011. doi: 10.1109/ICRA.2011.5980121. (Cited on pages 89 and 90.)

Jamie Shotton, John M. Winn, Carsten Rother, and Antonio Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput. Vision*, 81(1):2–23, 2009. doi: 10.1007/s11263-007-0109-1. (Cited on pages 43 and 47.)

Ksenia Shubina and John K. Tsotsos. Visual search for an object in a 3D environment using a mobile robot. *Comput. Vis. Image Und.*, 114(5):535–547, May 2010. ISSN 1077-3142. doi: 10.1016/j.cviu.2009.06.010. (Cited on pages 87, 88, and 95.)

Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pages 601–608. IEEE, 2011. doi: 10.1109/ICCVW.2011.6130298. (Cited on page 75.)

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In Andrew W. Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, volume 7576 of *Lecture Notes in Computer Science*, pages 746–760. Springer, 2012. doi: 10.1007/978-3-642-33715-4_54. (Cited on pages 47, 48, and 75.)

Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007. doi: 10.1016/j.websem.2007.03.004. (Cited on page 31.)

Richard Socher, Brody Huval, Bharath Putta Bath, Christopher D. Manning, and Andrew Y. Ng. Convolutional-recursive deep learning for 3D object classification. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 665–673, 2012. (Cited on page 47.)

Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 567–576. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298655. (Cited on pages 47 and 75.)

Sebastian Stock. Hierarchical hybrid planning for mobile robots. *KI*, 31(4):373–376, 2017a. doi: 10.1007/s13218-017-0507-7. (Cited on pages 13 and 41.)

Sebastian Stock. *Hierarchische hybride Planung für mobile Roboter*. PhD thesis, Universität Osnabrück, 2017b. URL `https://repositorium.ub.uni-osnabrueck.de/handle/urn:nbn:de:gbv:700-2017031715643`. (Cited on pages 13 and 41.)

Sebastian Stock, Martin Günther, and Joachim Hertzberg. Generating and executing hierarchical mobile manipulation plans. In *Proceedings of ISR/Robotik 2014; 41st International Symposium on Robotics*, pages 1–6. VDE, 2014. (Cited on pages 12 and 13.)

J. Stückler, N. Biresev, and S. Behnke. Semantic mapping using object-class segmentation of RGB-D images. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3005–3010, October 2012. doi: 10.1109/IROS.2012.6385983. (Cited on page 47.)

Jörg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from RGB-D video. *J. Real-Time Image Processing*, 10(4):599–609, 2015. doi: 10.1007/s11554-013-0379-5. (Cited on pages 28 and 48.)

Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Robot. Auton. Syst.*, 45(3-4):181–198, 2003. doi: 10.1016/j.robot.2003.09.004. (Cited on pages 87, 88, and 89.)

The RACE project. Description of work, September 2011. (Cited on page 11.)

Akshaya Thippur, Chris Burbridge, Lars Kunze, Marina Alberti, John Folkesson, Patric Jensfelt, and Nick Hawes. A comparison of qualitative and metric spatial relation models for scene understanding. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1632–1640. AAAI Press, 2015. (Cited on pages 43 and 46.)

Antonio B. Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, pages 273–280. IEEE Computer Society, 2003. ISBN 0-7695-1950-4. (Cited on pages 43 and 47.)

Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, volume abs/1809.10790, pages 306–316, 2018. (Cited on page 47.)

Vladyslav Usenko, Florian Seidel, Zoltan-Csaba Marton, Dejan Pangercic, and Michael Beetz. Furniture classification using WWW CAD models. In *IROS 2012 Workshop on Active Semantic Perception (ASP2012)*, Vilamoura, Portugal, 2012. (Cited on page 28.)

J.P.C. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P.H.S. Torr. Mesh based semantic modelling for indoor and outdoor scenes. In *CVPR*, pages 2067–2074, June 2013. doi: 10.1109/CVPR.2013.269. (Cited on pages 43 and 45.)

Tijn van der Zant and Thomas Wisspeintner. RoboCup X: A proposal for a new league where RoboCup goes real world. In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 166–172. Springer, 2005. doi: 10.1007/11780519_15. (Cited on page 1.)

Tiago S. Veiga, Pedro Miraldo, Rodrigo Ventura, and Pedro U. Lima. Efficient object search for mobile robots in dynamic environments: Semantic map as an input for the decision maker. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pages 2745–2750. IEEE, 2016. doi: 10.1109/IROS.2016.7759426. (Cited on page 87.)

Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 2018. doi: 10.1016/j.mechatronics.2018.02.009. Available online 2 March 2018. In Press, Corrected Proof. (Cited on page 1.)

Thomas Wiemann, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. A toolkit for automatic generation of polygonal maps – Las Vegas Reconstruction. In *Proc. ROBOTIK-12*, pages 446–415, 2012. (Cited on page 30.)

Thomas Wiemann, Hendrik Annuth, Kai Lingemann, and Joachim Hertzberg. An evaluation of open source surface reconstruction software for robotic applications. In *Advanced Robotics (ICAR), 2013 16$^{th}$ International Conference on*. IEEE, 2013. (Cited on page 37.)

T. Wisspeintner, T. van der Zant, L. Iocchi, and S. Schiffer. RoboCup@Home: Scientific competition and benchmarking for domestic service robots. *Interact. Stud.*, 10(3):392–426, 2009. ISSN 1572-0373. URL `http://www.dis.uniroma1.it/~iocchi/publications/AtHome-IS09.pdf`. (Cited on page 1.)

Walter Wohlkinger, Aitor Aldoma, Radu Bogdan Rusu, and Markus Vincze. 3DNet: Large-scale object class recognition from CAD models. In *ICRA*, pages 5384–5391. IEEE, 2012. ISBN 978-1-4673-1403-9. doi: 10.1109/ICRA.2012.6225116. (Cited on page 28.)

Yu Xiang, Xiangdong Zhou, Zuotao Liu, Tat-Seng Chua, and Chong-Wah Ngo. Semantic context modeling with maximal margin conditional random fields for automatic image annotation. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pages 3368–3375. IEEE Computer Society, 2010. doi: 10.1109/CVPR.2010.5540015. (Cited on pages 43 and 47.)

Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *CoRR*, abs/1711.00199, 2017. URL `http://arxiv.org/abs/1711.00199`. (Cited on page 47.)

Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1625–1632. IEEE Computer Society, 2013. doi: 10.1109/ICCV.2013.458. (Cited on page 75.)

Xuehan Xiong and Daniel Huber. Using context to create semantic 3D models of indoor environments. In *British Machine Vision Conference (BMVC)*, September 2010. (Cited on pages 43 and 45.)

B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, July 1997. doi: 10.1109/CIRA.1997.613851. (Cited on page 87.)

Yiming Ye and John K. Tsotsos. Sensor planning for 3D object search. *Comput. Vis. Image Und.*, 73(2):145–168, 1999. ISSN 1077-3142. doi: 10.1006/cviu.1998.0736. (Cited on pages 87, 88, and 95.)

Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In *2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May - 3 June 2017*. IEEE, 2017. doi: 10.1109/ICRA.2017.7989165. (Cited on page 47.)

Liwei Zhang, Martin Günther, Masoumeh Mansouri, Federico Pecora, and Luís Seabra Lopes. Deliverable D5.3 – Year-2 Demonstrator. The RACE project (FP7-ICT-2011-7-287752). Public Deliverable, December 2013. (Cited on page 23.)

# List of Figures

# List of Tables

# List of Algorithms

# List of Listings

# Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any resources or means other than those indicated.

Osnabrück, April 2021