Integrating Qualitative Reasoning and Human-Robot Interaction for Domestic Service Robots

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Stefan Schiffer

aus Aachen

Berichter: Universitätsprofessor Gerhard Lakemeyer, Ph.D.
Universitätsprofessorin Dr. Maren Bennewitz

Tag der mündlichen Prüfung: 17. Dezember 2014

Abstract

The last decade has seen an increasing interest in domestic service robots. Particular challenges for such robots especially when performing complex tasks are deliberation, robust execution of actions, and flexible human-robot interaction. Despite progress in qualitative reasoning and human-robot interaction their integration is an open issue.

In this thesis, we build on an existing cognitive mobile robot platform and make a series of contributions to integrate qualitative representations, high-level reasoning and human-robot interaction for an intelligent domestic service robot. We start by introducing the domestic service robotics domain and parts of the RoboCup@Home methodology that we contributed to. Before we can actually turn to our main focus, we equip the system with a set of basic capabilities that are required for a service robot in human environments. As a bridge between perception and symbolic reasoning we provide a semantic mapping scheme that allows to centrally manage information about the environment. With a novel hierarchical object recognition method we are further able to classify even yet unseen objects.

Then we move on to the main contributions of this thesis. First, we extend the robot with important modes for human-robot interaction by adding components for speech, face, and gesture recognition as well as for speech synthesis and a virtual facial display. For the speech input we proceed with a simple form of natural language understanding that allows a limited form of error recovery. Second, we introduce qualitative representations and control to our high-level control system. After integrating a general account for qualitative information based on fuzzy sets into our high-level language we also add means to specify and use fuzzy controllers for behaviour specification. Then we focus on spatial data and provide a formalization that allows for representing and reasoning with qualitative positional information in our high-level language. Lastly, we increase the robustness of the robot against internal errors and add to the flexibility in dealing with possibly faulty external input. We integrate a basic form of self-maintenance that allows the robot to recover from internal errors by itself.

Zusammenfassung

Innerhalb des letzten Jahrzehnts ist das Interesse an Servicerobotern für Haushaltsumgebungen stetig gestiegen. Einige der Herausforderungen für solche Roboter, insbesondere bei der Bewältigung von komplexen Aufgaben, bestehen in der Deliberation, der robusten Ausführung von Aktionen und der flexiblen Mensch-Roboter-Interaktion. Ungeachtet der Fortschritte auf den Gebieten des qualitativen Schließens und in der Mensch-Roboter-Interaktion ist die geeignete Integration dieser beiden Aspekte eine noch offene Fragestellung.

Aufbauend auf einem existierenden Robotersystem liefert diese Dissertation eine Reihe von Beiträgen zur Integration von qualitativen Repräsentationen, high-level Reasoning und Mensch-Roboter-Interaktion für einen intelligenten Haushaltsroboter. Wir beginnen mit einer Vorstellung der Haushaltsrobotik und von Teilen der Methodologie von ROBOCUP@HOME, an der wir mitgewirkt haben. Bevor wir uns den eigentlichen Beiträgen dieser Arbeit widmen können, müssen wir zunächst das Robotersystem mit einigen Basisfertigkeiten ausstatten, die jeder Serviceroboter mitbringen muss, der in menschlichen Umgebungen arbeiten soll. Als Bindeglied zwischen sensorischer Wahrnehmung und symbolischem Schließen stellen wir eine Methode zur semantischen Kartografie vor, die die zentrale Verwaltung von Informationen über die Umgebung des Roboters ermöglicht. Dank eines neuartigen hierarchischen Verfahrens zur Objekterkennung sind wir in der Lage sogar zuvor unbekannte Objekte zu erkennen.

Dann widmen wir uns den Hauptbeiträgen dieser Dissertation. Zuerst erweitern wir unseren Roboter mit wichtigen Fähigkeiten zur Mensch-Roboter-Interaktion, indem wir Komponenten zur Sprach-, Gesichts- und Gestenerkennung sowie zur Sprachsynthese und zur Darstellung eines künstlichen Gesichts integrieren. Auf die Spracherkennung folgt eine einfache Form des Sprachverstehens, welche eine begrenzte Fehlerkorrektur erlaubt. Dann erweitern wir unser high-level Kontrollsystem um Möglichkeiten für qualitative Repräsentationen und zur qualitativen Kontrolle. Nachdem wir diese beiden Aspekte zunächst generisch behandeln, stellen wir eine spezielle Erweiterung zur Repräsentation von und zum Schließen mit qualitativen räumlichen Informationen vor. Schließlich erhöhen wir die Robustheit unseres Systems gegenüber internen Fehlern durch die Einführung einer Selbstwartungsfunktion. Diese versetzt den Roboter in die Lage, gewisse Fehler selbständig zu beheben.

Acknowledgments

Doing my PhD, in general, and writing this thesis, in particular, has been an extensive journey for me, pleasure and pain, hard and long sometimes, exhausting at times, but definitively worth completing. This would not have been possible and it would not have reached the present state without the help of many people whose support I would like to acknowledge and thank for in the following.

First of all, I would like to thank Gerhard Lakemeyer for giving me the opportunity to conduct my research in his group. He let me start exploring the field of domestic service robotics for the Knowledge-Based Systems Group and he let me find my topic without many restrictions, yet subtly pointing out paths worth to explore or better to shun. I am thankful for the freedom to play that he gave me. He also gave particular support on the formal parts of this thesis. I'd like to thank Maren Bennewitz for kindly agreeing to be my second reviewer, as well as Jürgen Giesl and Thomas Seidl for acting as additional examiners in my defense committee.

Even before my PhD studies I received support and mentoring in many different ways from one person who I am greatly indebted to and whom I would like to thank: Alexander Ferrein. He encouraged me more than twelve years back when I did my robot lab course with him as a diploma student, I continued under his guidance as a Hiwi and later I wrote my diploma thesis with him as my advisor. Working with him has always been a pleasure to me and I greatly benefited from his experience. I want to thank Daniel Beck for spending many long evenings in the Mogam and weekends in the office with me, thus largely contributing to the progress of this work. The many discussions with him about our theses and beyond helped me both, in professional and in emotional matters related to this thesis. I thank Christoph Schwering for being an office mate with the right portions of being funny, witty, and serious. I would like to thank all my colleagues at the KBSG, in order of appearance: Alexander Ferrein, Frank Dylla, Jens Claßen, Daniel Beck, Vaishak Belle, Tim Niemüller, Christoph Schwering, and Martin Liebenberg, as well as the people at Informatik 5, in particular the administrative staff, in alphabetical order, Daniele Glöckner, Gabriele Hoeppermanns, Tatjana Liberzon, Reinhard Linde, and Claudia Puhl.

Much of the work presented in this thesis has been implemented, deployed, and tested with the ALLEMANIACs ROBOCUP team which I am proud to be a member of. Credit is due especially to Tim Niemüller for his endless efforts in all robotic and hacking matters. The particular members are, in order of appearance over the years, Alexander Ferrein, Frank Dylla, Jost Wunderlich, Nils Springob, Andreas Strack, Christian Fritz, Stefan Jacobs, Lutz Herrmanns, Tim Niemüller, Philipp Vorst, Vaishak Belle, Masrur Doostdar, Daniel Beck, Bahram Maleki-Fard, Christoph Schwering, and Tobias Baumgartner. I enjoyed many sleepless nights, some of them being very cold (below zero), lots of fun at and around events, creativity, improvisation, team spirit, and much more. I would like to thank my theses students for their contributions to this thesis and beyond. Students were, in order of appearance: Vaishak Belle, Masrur Doostdar, Andreas Wortmann, Tobias Baumgartner, Safoura Rezapour Lakani, and Niklas Hoppe. As part of my efforts in domestic service robotics I dealt with many people from around the world, most notably the members of the ROBOCUP@HOME league's executive and technical committee, in particular, Tijn van der Zant, Thomas Wisspeintner, and Luca Iocchi.

The sometimes very little amount of spare time was made much more enjoyable by people I want to thank for their friendship and support. I thank Philip Schuster for listening to my complaints on our jogging rounds and for listening to and commenting on earlier versions of my defense talk. Thanks also go to Shangning Postel-Heutz especially for her support in the early years of my doctoral studies. I want to express my gratitude to Alexander Ferrein for being more than a room mate, a colleague, and a mentor, for becoming a friend. Daniel Beck has my heartfelt gratefulness for becoming a friend, for being on the same wavelength and I feel honored that we now share so many things. I am indebted to Volker Lingens for always being a friend, for never giving up on me, and also for reading earlier versions of this work.

Last but not least, I do owe an uncountably great deal to my family and to my wife Sandra Geisler (even though, technically, we are not married yet). My family, first and foremost my parents have always supported me with whatever choice I made and I cannot be grateful enough for that. All of my family contributed to how and who I am today. Sandra supported me with her love, she kept cheering me up when my mood was bad and she put up with me and my quirks. Many times, especially in the last month of completing this thesis, she pitched in and she kept me free for finishing my dissertation. Finally, my son Jakob cheered me up with his chuckles and brightened up any day as lousy as it may have seemed to be by just being the great little guy that he is.

To my family

Contents

List of Figures

List of Tables

1 Introduction

The idea of artificial servants and companions is already present in ancient mythology. Homer mentions walking tripods created by Hephaestus (Homer *Iliad*, 18.369) and Plato tells us about life-like statues capable of self-motion made by Daedalus (Plato *Meno*, 97d). Also in real life, the concept of an automaton that today would be referred to as a robot dates back to at least the fourth century before Christ. Aristotle already thought about automatons as a possibility to someday end slavery (Aristotle *Politics*, 1.1253a). In the twelfth century, automatic machines were designed and constructed for kitchen appliances or for entertainment by muslim inventor Al-Jazari (Rosheim, 1994). At the end of the fifteenth century Leonardo da Vinci designed a humanoid automaton, a mechanical knight often referred to as *Leonardo's robot*, that would be able to sit up, to wave its arm, and it would even be able to move its head and its jaw. The mechanical design has been an inspiration of a design for an astronaut-helping robot at NASA (Rosheim, 1997). These are just some examples of early robot-like inventions. The word *robot* was introduced by Čapek (1920) in his play "*R.U.R. – Rossum's Universal Robots*" premiering in 1921 that features artificial people working for humans. *Robotics* in its modern sense was coined by Isaac Asimov, mainly in his formulation of the *Three Laws of Robotics* (Asimov, 1950).
Robotics has been an active field of research for the last five decades. While very often robotics is used to refer to machines used in industrial automation, in this thesis we are concerned with mobile robots and, more precisely, with autonomous mobile robots that are endowed with some form of intelligence. According to (Murphy, 2000) an intelligent robot is "*a mechanical creature which can function autonomously.*" *Autonomy* according to (Bekey, 2005) "*refers to systems capable of operating in real-world environments without any form of external control for extended periods of time.*" The first robot that can be attributed with these qualities perhaps is Shakey (Nilsson, 1984). It is a general purpose mobile robot developed from 1966–1972 that uses a world model and that is able to perform tasks which, among others, require planning.

1.1 Domestic Robots – At Your Service?

Thrun (2004) distinguishes three kinds of robots: industrial robots, professional service robots, and personal service robots. *Industrial robots* are the most widely spread, they work in factories and perform automation tasks such as welding. They usually do not require much intelligence and they do not interact with people. *Professional service robots* help people in pursuing their professional goals, however, unlike industrial robots

not in industry applications. Examples are robots in mining, robots transporting items in hospitals, or robots for search and rescue missions. *Personal service robots* are used for entertaining or assisting people and they are deployed for recreational activities and in domestic settings. An example for the former are robotic toys such as the Sony Aibo[1] while examples for the latter are vacuum cleaning or lawn mowing robots. Thrun (2004) mentions the widely acknowledged shift from industrial to service robots in 2004 already and discusses research challenges implied by this shift. Most of these challenges concern human-robot interaction and robot autonomy.
Two advertisements titled *"We're building a dream, one robot at a time."* (Honda, 2002, 2003) feature the humanoid robot ASIMO[2] standing in front of a typical American suburban house among its owners, acting as if it was just a normal member of the family (at least as prominent as the dog depicted in the same image). Although the ads were printed more than ten years ago, we do not see humanoid robots as family members in our daily life yet. Actually, we do not see commercially available humanoid robots for personal use deployed anywhere. This is probably also due to the complexity that robots face in real-world scenarios, when they leave the more restricted conditions one usually finds in development labs. In his 2007 article in *Scientific American*, Bill Gates sketches his vision of every home having at least one robot (Gates, 2007). We are already very much on our way for this to become reality. Yet, current domestic robots usually are very specialized on one particular task like vacuum cleaning or lawn mowing. Our aim however, is to work towards an intelligent mobile robot that is not fitted for one particular task only, but that is able to perform several tasks and that is also able to solve complex problems in complex settings in domestic environments.
The research areas in robotics are as diverse as the possible applications of robots. Some efforts are concerned with the mechanical design, other research covers development and use of sensors and effectors, still others focus on a particular low- or mid-level capability of a robot such as navigation or manipulation. For service robots also aspects in human-robot interaction (HRI) are important topics of interest for research. Here, we are concerned with research in the field of *cognitive robotics* and our application domain is domestic service robotics. Hence, human-robot interaction in cognitive robotics is an integral topic for us as well. Cognitive robotics as introduced by the late Ray Reiter is to be understood as "the study of the knowledge representation and reasoning problems faced by an autonomous robot (or agent) in a dynamic and incompletely known world" (Levesque and Lakemeyer, 2008). A central effort of Reiter's vision "is to develop an understanding of the relationship between the knowledge, the perception, and the action of such a robot" (Levesque and Reiter, 1998).
Mobile robots have advanced from pure repetitive tasks to work in human populated environments and also with humans in the last decades. They get better at any particular thing they do and there are also more things they are able to do. Besides increased capabilities in a particular task at hand, autonomous robots in general and those that perform assistive tasks in complex scenarios in particular benefit from high-level control mechanisms such as planning (see for example (Pineau et al., 2003)). In this thesis we work towards an integrated system that extends an existing autonomous mobile robot system with means to successfully operate in complex domestic service robotics

[1]The Aibo product family was discontinued by Sony in 2006.
See, for example, `http://www.sony-aibo.co.uk/` for more information.
[2]`http://asimo.honda.com/`

scenarios. To this end, the robot must be able to perform a basic set of duties in domestic settings and it must be able to engage in communication with the human users around it. This not only demands for extending the capabilities of a mobile robot with methods for human-robot interaction. It also means, that differences in representations between the robot and humans must be overcome. What is more, when the robot should assist humans with complex, cognitively challenging tasks it must be endowed with some form of reasoning that allows to take decisions in complex scenarios. In addition, longer autonomous operation can only be made possible if the robot can handle certain variations and unavoidable errors by itself. Also, it should be flexible in dealing with human fallibility. In this thesis, we want to take a series of steps towards the application of an autonomous cognitive mobile service robot with a logic-based high-level control in interactive domestic domains featuring the above description.

1.2 Goals and Contribution

The goal of this thesis is, first, to extend an existing mobile robot system with means for affective human-robot interaction (HRI). However, it is not an exercise in any of the particular methods for HRI. Rather, we focus on further developing the high-level control of our cognitive robot for successful interaction with humans. We aim to make it fit for applications in domestic domains. Second, to bridge the gap between different representations of humans and the robot is one of the key issues we want to look at. We do this with a particular focus on representations for positional information that is dominant in any domestic scenario. Last, we will investigate means to increase both robustness against internal failures and flexibility towards faulty external input. To this end, this work presents a step towards building an intelligent, robust and flexible robotic system that is applicable for complex interactive service tasks in domestic domains. The contributions in particular are as follows.

Basic Components
We tailor our robotic platform towards its application in the domestic service domain. Part of the contribution to a solid basis is a reliable method for local navigation with safe collision avoidance. Also, we provide a semantic mapping mechanism that allows to conveniently adapt to changing environments and to provide the robot system with semantic information about the environment. Moreover, to enable detecting and recognizing unknown or unspecific objects we propose a novel architecture that combines existing methods for visual object recognition with attribute-based descriptions.

Human-Robot Interaction
To enhance the robot's capabilities in interacting with humans we endow our robot with three important means in communication: speech recognition, face detection, recognition and learning as well as gesture recognition. We propose a robust speech recognition system that uses two different kinds of decoders to filter out false-positive recognitions reliably. We contribute a one-step framework for detecting, recognizing and learning faces of human users in real-time using random forests. As a means for gesture recognition we present a modular system that detects human hands and recognizes their posture. Together with the face detection this already allows to detect and interpret

pointing gestures. By tracking the hand position over time and matching the trajectory to known templates we are able to recognize dynamic hand gestures. Furthermore, we increase the affectivity of the robot by providing an artificial face that is capable of displaying basic emotions and supporting the actions of the robot. The integration of these methods for HRI with our existing high-level control to solve a complex task in interactive domestic service robotics is shown in a realistic demonstration scenario.

Qualitative Representations and Reasoning
In order to close the gap between the representations that humans and robots use to refer to things in the real world we realize qualitative representations and behaviour control in our logic-based high-level framework. The proper formal integration allows for a transparent use of human-like notions in agent programs and in human-robot interaction. We show the general applicability of both approaches for different tasks in the domestic service robot domain. Furthermore, we present a particular extension of the system to the spatial domain to represent and to reason with qualitative positional information. After reiterating an existing approach to qualitative spatial information we give a formalization of qualitative positional fluents that includes a frame of reference to account for contextual information. Together with a unified reasoning mechanism we provide a powerful framework for seamlessly using qualitative positional information in our high-level control.

Robustness and Flexibility
Finally, we attend to the problem of increasing the robustness and the flexibility of the system. This is especially important when the robot works with human laymen and over extended periods of time. Specifically, we tackle two types of resilience against faults and errors. For one, we care for internal errors by endowing the robot with a form of self-maintenance. This enables the robot to recover from certain errors regarding states of its internal components on its own. For another, we account for human fallibility in instructing the robot via natural spoken language. We present an action theory in the situation calculus for natural language interpretation that shows the applicability of our high-level control for implementing cognitive functions of a domestic service robot.

1.3 Outline

The remainder of this thesis is organized as follows.

Chapter 2
We introduce the challenges in domestic service robotics as an application domain where autonomous mobile robots work and interact with humans in a domestic setting in Chapter 2. We present the RoboCup@Home initiative, a testbed and a benchmark for assistive robot technology in a complex naturalistic environment. Afterwards, we review related work in personal service robotics. Other than to discuss efforts in benchmarking robots and to give an overview of diverse efforts in developing personal domestic service robots we also take a look at high-level decision making approaches for service robots, in particular in human-robot scenarios. Then we present our robotic platform Caesar that we develop and use as a service robot in domestic environments. We detail

one of the most fundamental modules, the local navigation and collision avoidance. Afterwards, we briefly sketch selected software components that extend our robot with semantic mapping, object recognition and mobile manipulation. Parts of this chapter have previously appeared in (Schiffer et al., 2006a; Wisspeintner et al., 2009, 2010; Jacobs et al., 2009; Niemueller et al., 2013).

Chapter 3
We lay out the mathematical foundations for the subsequent chapters of this thesis in Chapter 3. We start with a brief review of decision trees before we discuss random forests that we use for image classification, namely in our face and our gesture recognition modules. Also, we introduce Markov Decision Processes as a model to describe stochastic dynamic systems. Then we give an introduction to fuzzy logic as a means to use qualitative notions in representation and control. For our effort in enabling a qualitative description of spatial information we then review a method for representing qualitative positional information that we build upon. Lastly, we introduce the situation calculus, GOLOG and READYLOG which build the foundations of the high-level control of our robot.

Chapter 4
We extend our robot with methods for three important modes in human-robot interaction in Chapter 4. We begin with a presentation of our approach to robust speech recognition in restricted domains. Next, we explain our real-time approach to one-step face detection, recognition and learning using random forests. Further, we propose a modular architecture for gesture recognition that works in a filter-and-refine fashion. Also, we describe additional components for HRI such as speech synthesis as well as a virtual facial display and information panel. Finally, we showcase the application of the above methods for human-robot interaction and our logic-based high-level control for a domestic service task which illustrates the applicability and the benefit of deliberation. The contributions from this chapter were partly published in (Doostdar et al., 2008; Belle et al., 2008; Schiffer et al., 2009, 2011a, 2012a).

Chapter 5
We present our approaches to bridge the gap between the representations of the human users and the domestic service robot with a focus on spatial information Chapter 5. First, we specify a semantics for qualitative fluents based on fuzzy sets. Then, we introduce fuzzy controllers in our high-level language for qualitative control for our robot. We illustrate both approaches with applications in domestic service robotics tasks. Lastly, we present an extension of our qualitative fluents for positional information that allows for seamless reasoning in our high-level framework. The approaches presented in this chapter were first presented in (Ferrein et al., 2008, 2009; Schiffer et al., 2010a, 2011b, 2012c).

Chapter 6
How to increase the robustness and flexibility of our system with respect to internal and external sources of errors is discussed in Chapter 6. First we review an approach to self-maintenance that uses explicitly formulated dependencies between actions on the

task-level and the states of internal components of the robot to increase the resilience against failures in the inner workings of the system. Then we talk about our method of natural language processing of human inputs where we cast the interpretation of spoken user commands as a decision-theoretic planning process. Parts of this chapter were discussed in (Schiffer et al., 2010c,b, 2012d, 2013a) before.

Chapter 7
We conclude this thesis in Chapter 7, where we summarize, draw conclusions and give an outlook on future work.

Large parts of the approaches and the development in this thesis were implemented, deployed and tested with the AllemaniACs RoboCup team. The beginning of the endeavour in the @Home league is reported on in (Schiffer et al., 2006a). The team's participation is also documented in a series of team description papers (Ferrein et al., 2006; Schiffer et al., 2007; Schiffer and Lakemeyer, 2008; Schiffer et al., 2009; Schiffer and Lakemeyer, 2011). Some of the material covered in this thesis was partly presented in Bachelor's, Diploma, and Master's theses before (Belle, 2008; Wortmann, 2010; Doostdar, 2011; Hoppe, 2011; Baumgartner, 2011; Lakani, 2013). I would like to take the opportunity to thank the people involved for their support and contribution.

2 Domestic Service Robotics

Due to the ageing society in western countries, home care of elderly people becomes more and more important. Living in your well-known and familiar home for as long as possible, keeping independence and dignity and participating in social life are invaluable benefits for living quality.

Besides relatives and professional caregivers caring for elderly people, assistance systems for facilitating every-day tasks are becoming more important. Gates (2007) talks about his vision of a not so distant future where robots are common household items helping with our daily activities. In fact, we already see this becoming a reality with specialized robotic devices for vacuum cleaning or lawn mowing available at favourable prices. Our aim in this thesis is a robot not tailored towards performing one particular task only, but we envision a versatile autonomous mobile robot that can help with a lot of things around the house. For this to become reality, our robot needs to fulfil very complex tasks and requires sophisticated techniques, many of them from the field of Artificial Intelligence research. A robot living together with human beings helping them in their daily lives needs to be robust, fail-safe, understand the human and it needs to exhibit intelligent behaviour. For the robot applications we have in mind this means that a lot of different capabilities need to be integrated into the robot system. As the robot is operated by laymen and not by robot experts, an intuitive human-machine interface is of utter importance. Of course, the robot needs basic abilities such as to localize itself in the domestic environment, it needs to navigate safely, and it needs to perform the given tasks in an intelligent way.

In this chapter we introduce the domestic service robotics domain and our robot platform. We present some of the contributions that are not in the focus of the thesis but that are still essential for a working domestic service robot system. We start by characterizing the challenges posed to an intelligent service robot in domestic domains and the efforts that are being made in this regard in Section 2.1. This includes the RoboCup@Home initiative and its benchmarking methodology that we contributed to. We review existing work in the field of domestic service robots, both, in developing but also in benchmarking the same in Section 2.2. After we present our domestic service robot platform Caesar in Section 2.3 we detail one of the most fundamental components of our mobile robot – the collision avoidance – in Section 2.4. Then in Section 2.5 we review selected software of Caesar. We only briefly discuss some of the components that were developed in this thesis and which are needed for our service robot but which are not in our primary focus.

2.1 Autonomous Service Robots in Domestic Domains

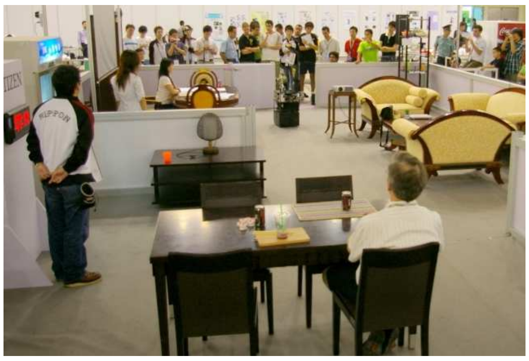
In this section we introduce the domestic service robotics domain. Further, we present the ROBOCUP@HOME initiative as a testbed and as a benchmark for service robotics in domestic environments.

2.1.1 Challenges

Consider an elderly person living at home alone. With increasing age such a person could suffer from increasing health issues such as difficulties walking and from problems with daily chores. Yet he or she might not want to leave his or her accustomed surrounding. People in these kinds of situations could certainly benefit from support given by an autonomous robot in their daily life tasks. The robot can help around the house, for example, it can carry things that are too heavy or it can help fetch things if it is burdensome for the human to move around too much. A domestic service robot, that is to say, a robot providing service around the house could enable elderly persons to stay in their familiar environment much longer.

A robot for such a scenario, for one, needs to be equipped with a set of components that provide low-level and mid-level capabilities. That is, it needs to have means to navigate and to localize itself, and it should have methods in place for object detection and recognition and it should have means for mobile manipulation of objects also. Apart from these common requirements for an autonomous mobile robot, its application in domestic scenarios with humans around places additional demands. A robot in a domestic environment needs to be able to detect and to recognize people and it should be able to track their positions over time. It must be able to communicate with the humans around it and it needs to interpret the commands these humans use to instruct the robot. What is more, the robot also needs a powerful and flexible high-level control that can figure out an applicable course of action for complex tasks that require an intelligent combination of its basic capabilities. This is even more important when the robot should lessen the cognitive load of a task for an elderly or disabled person.

Hence, two very important issues in domestic service robotics are intelligent behaviour and human-robot interaction. Both should further be flexible and robust. When humans and a robot share the same habitat and are working together, the robot must exhibit some form of intelligent behaviour to be helpful for extended periods of time. At the same time it needs to be robust against various types of variations and errors in its environment, not only those caused by the humans. As an example for the former kind of variations consider that humans are messy (from a robot's perspective) and tend to leave things in different places. They move around items frequently so that the environment is not as predictable as, say, with an industrial setting. For the latter kind of variations and errors, notice that a robotic system for complex tasks is a complex system itself. Modules might crash and components might get stuck in certain situations. Robustness against those problems allows for enduring autonomy which is crucial when the robot needs to assist a person over extended periods of time. A domestic service robot has to meet the cognitive demands that arise in daily chores, where complex problems sometimes require deliberation. Strictly predefined behaviours are prone to errors and tend to fail since they cannot account for more or less subtle variations and the uncertainty inherent in real-world scenarios all the time. From a human-robot


(a) Arena from the 2008 RoboCup@Home competition with CAESAR performing the *WhoIsWho* test in it looking for people

(b) The arena from the 2009 RoboCup@Home competition in Graz

Figure 2.1: Examples of arenas in the RoboCup@Home competition

interaction perspective, robots need to be operable by laymen and they need to account for imprecision and fallibility of humans, in general, and of elderly people, in particular.

2.1.2 The RoboCup@Home Initiative

In order to bring forward the development of domestic robots as described above, there exist a number of efforts. One among them is the ROBOCUP@HOME initiative (van der Zant and Wisspeintner, 2005, 2007), which particularly focuses on domestic robot applications. ROBOCUP@HOME was established as a distinguished league inside the ROBOCUP initiative (Kitano et al., 1997a) in 2006. The motivation was to provide a testbed and a benchmark for domestic service robotic systems that brings such robots out of their confined lab conditions into the real world. A robot system in the competition has to fulfil a number of preset challenges such as seeking lost objects or being a robot butler recognizing people and seating them in the living room. ROBOCUP@HOME is designed to be both, a scientific competition and a benchmark for domestic service robots (Wisspeintner et al., 2009). It is an effort to test individual components of DSR systems as well as the integration of the system as a whole.
The general idea in the @HOME competition is to set up a home-like scenario that is as realistic as possible and to let robots perform a set of tests in that environment. Examples of the arena are depicted in Figure 2.1. The underlying concept is to test and evaluate robots in scenarios that are as complex as the real world.
The @HOME competition is organized in so-called tests which check for different abilities of the robot. The tasks in those tests are oriented towards real-life applications of domestic service robotics. For example, the robot has to act as a host at a party by locating and serving people, or it has to move around in the apartment (or a supermarket) to find and to fetch certain objects for its human user. The tests change over time to integrate new scenarios and to adjust the relative priority of specific robot abilities. For the 2014 version of the rules we refer to (Nardi et al., 2014). The tests in @HOME require working solutions to specific (sub)tasks such as localization, navigation, manipulation, face recognition and others. What is even more important, a successful system has to fully integrate those capabilities to a complete system that also has means of deliberation to purposefully work towards achieving a particular mission in a home-

like scenario. The more complex the task, the more does deliberation pay off. Our team, the ALLEMANIACS, participated quite successfully in these competitions since they were established in ROBOCUP@HOME. We were able to win the world championship in 2006 and 2007, and came in second in 2008. We won the RoboCup German Open competition in the @HOME league in 2007 and 2008. We did not participate the RoboCup world championship after 2009 and we did not participate the RoboCup German Open competition after 2008. However, we used the RoboCup German Open 2011 event to perform the robOCD demonstration that is presented in Section 4.5.

Key Features As argued in (Wisspeintner et al., 2009) a successful domestic robot system needs to possess a set of *key features*, that is, certain abilities and properties. The key features can be divided in two groups: (1) *functional abilities*, that are abilities required by the robot to solve a particular task within a test, and (2) *system properties*, that are generally required by the robot to accomplish tasks.
Functional abilities include specific functionality that must be implemented on the robot in order to perform decently in the tests. Each test requires a certain subset of these abilities, as they are also directly represented in the score system. Teams must thus decide which of these abilities to implement and what degree of performance to achieve, depending on their background and the kind of tests they intend to participate in. Functional abilities currently are

- *Navigation*: the task of path planning and safely navigating to a specific target position in the environment, avoiding (dynamic) obstacles
- *Mapping*: autonomously building a representation of a partially known or unknown environment on-line
- *Person recognition*: detecting and recognizing a person
- *Person tracking*: tracking the position of a person over time
- *Object recognition*: detecting and recognizing (known or unknown) objects in the environment
- *Object manipulation*: grasping or moving an object
- *Speech recognition*: recognizing and interpreting spoken user commands (speaker dependent and speaker independent)
- *Gesture recognition*: recognizing and interpreting human gestures.

System properties include demands on the entire robotic system that we consider of general importance for any domestic service robot. They can be described as “soft skills” which must be implemented for effective system integration and successful participation in the @HOME competition. Initial system properties are

- *Ease of use*: Laymen should be able to operate the system intuitively and within little amount of time.
- *Fast calibration and setup*: Simple and efficient setup and calibration procedures for the system.
- *Natural and multi-modal interaction*: Using natural modes of communication and interaction such as natural language, gestures or intuitive input devices like touch screens.

- *Appeal and ergonomics*: General appearance, quality of movement, speech, articulation or HRI.
- *Adaptivity / General intelligence*: Dealing with uncertainty, problem solving, online learning, planning, reasoning.
- *Robustness*: System stability and fault tolerance.
- *General applicability*: Solving a multitude of different realistic tasks.

In RoboCup@Home, the annual competition is used to benchmark domestic service robot systems. Benchmarking can be distinguished in *system benchmarking* where the robotic system is evaluated as a whole and *component benchmarking* where single functionality is evaluated. While the latter is important to compare different solutions to a specific problem the former assesses the general performance of a complete robot system. To measure the advances of participating teams, the results of the competition are compared from year to year.

Since every test is associated with a set of the key features, the scoring system allows to relate the performance of the competitors to the desired abilities. To improve on itself, the performance over the years with respect to those abilities has to be analysed. Then, by adjusting the representation of each ability in the test sets, the focus can be shifted towards those abilities that more emphasis should be put on in the further development. As an example, to put more attention on the development of the intelligence of the robots, the *general purpose service robot test* was added to the set of tests (van der Zant and Iocchi, 2011) for the 2010 competition. In this test, the robot has to understand and execute a series of up to three actions. The actions have not been determined before the test starts. Also the parameters to those actions will only be selected during the test. Actions to choose from as well as objects and location that function as parameters are taken from the other tests in the competition. The focus here is on high-level cognitive capabilities of the robot.

While the functional abilities get checked in the particular task to solve in a test, the system properties are monitored by the overall requirements and the general rules. For example, the operation of the robots by uninstructed laymen requires ease of use, restrictions in setup time and calibration procedures assert that systems are *ready-to-use*. Moreover, human-robot interaction is integrated in that only natural modes such as speech and gestures are allowed for the interaction with a robot. The progress of the league is reflected in the changes in the rules[1] over the years from the first competition (Wisspeintner and van der Zant, 2006) to the most recent one (Nardi et al., 2014).

2.2 State of the Art and Related Work

In this section we review related work on robotic competitions and benchmarking, on developing personal service robots and on high-level control approaches for personal robots.

2.2.1 Benchmarking and Robotic Competitions

During the development of domestic service robotic systems these also have to be tested and evaluated. Benchmarking has already been recognized as an activity of fundamental

[1]http://www.robocupathome.org/rules/

importance to advance robotic technology (del Pobil, 2006; Sabanovic et al., 2006). Also, competitions have been identified as an appropriate means to do benchmarking (Behnke, 2006). However, beside existing testbeds and benchmarks for rather specific tasks there have only been few efforts to test systems of a complexity as is necessary in naturalistic scenarios reflecting the challenges posed in domestic service robotics. When robots enter human environments, there are also non-technical aspects to evaluate, of course. For instance, a description of (mostly) psychological benchmarks to evaluate socially assistive robots can be found in (Feil-Seifer et al., 2007).
Using competitions as a means to perform benchmarking is useful. Competitions further promote interaction and communication among the participants. However, many existing competitions only evaluate a single or a very limited set of capabilities.

AAAI Mobile Robot Competition The AAAI Mobile Robot Competition was one of the first robot competitions, established in 1992 (Balch and Yanco, 2002). The series offers great visibility within the AI scientific community because the competition is held in conjunction with the AAAI and (sometimes) IJCAI Conferences on Artificial Intelligence and many important scientific and technological achievements were demonstrated during these competitions and HRI has been a topic in the competition, e.g. (Rybski et al., 2007). However, the focus of the competition and the benchmarks change on a yearly basis. This change of focus makes it difficult to approach problems in a continuous and iterative way, although a relevant suite for benchmarking AI and robotics technology with relevance for real-life applications is targeted. Also, building up a community with a long-term goal is difficult.

DARPA Grand Challenge The DARPA Grand Challenges[2] are perhaps the most prominent robotic competitions. GPS navigation together with multi-modal sensor data fusion were commonly used to face the uncertainties and dynamics of real-world application scenarios. In the Urban Challenge, even real traffic rules were applied. Still, the complexity was limited by restricting the environmental conditions. Also, context information was given such as via a predefined map, the route network definition file (RNDF), which consisted of way-points with GPS coordinates, connection types, traffic signs, number of lanes, width of lanes, etc. Of course, participating in these challenges required a lot of effort. The latest event in the series of DARPA Challenges is the DARPA Robotics Challenge[3] where participating teams develop robot systems for assisting humans in disaster response. Robots must complete a series of very challenging tasks relevant in disaster response. The focus is on the performance of robots in unpredictable and unstructured environments. The finals are set to take place in 2015.

Mobile Manipulation Challenge An initiative with a focus on manipulation is the mobile manipulation challenge, held in 2009[4] at IJCAI, in 2010[5] at ICRA, in 2012[6] at ICRA, and in 2013[7] at ICRA. The emphasis of the first edition was "on demonstration, rather

[2]http://archive.darpa.mil/grandchallenge/
[3]http://www.theroboticschallenge.org/
[4]http://www.mobile-manipulation-challenge.net/ dead link as of 2014-08-28!
[5]http://www.willowgarage.com/mmc10
[6]http://www.icra2012.org/program/robotChallenge.php
[7]http://mobilemanipulationchallenge.org/

than comparative experimentation". It is getting a bit more competitive since. However, the aim is also to bring together researchers and commercial groups demonstrating complete platforms that are able to perform mobile manipulation tasks.

Semantic Robot Vision Challenge The Semantic robot vision challenge (Helmer et al., 2009) focuses on vision. The objective for a robot here is to locate a previously unknown object. Information on the object is typically acquired from the Internet. The competition tries to bring together computer vision and robotics research but only tackles one specific topic.

RoboCup Soccer With its ultimate goal "By the year 2050, develop a team of fully autonomous humanoid robots that can play and win against the human world champion soccer team", RoboCup (Kitano et al., 1997b) soccer has proven to provide an efficient means of interaction and communication among research groups. It combines scientific research, competition, benchmarking and reality checks on various concepts. The focus on soccer is a limitation though. Due to the restricted environmental conditions and rules there is an over-specialization of solutions.

RoboCup Rescue The scope of RoboCup broadened to other applications early with robot search and rescue. Rescue competitions started in 2000 within the AAAI Mobile Robot Competition (Schultz, 2001) and since 2001 within the RoboCup Rescue initiative (Kitano and Tadokoro, 2001). RoboCup Rescue competitions have defined standard rescue arenas and tasks for benchmarking robotic search and rescue missions and for measuring an increase in performance of the rescue robotic technology in a standardized abstracted environment. Within the Rescue competitions, also common metrics for HRI have been defined (Steinfeld et al., 2006) and effective evaluation of HRI techniques have been carried out (Yanco et al., 2004; Drury et al., 2005), with a specific focus on the interfaces used by operators to interact remotely with the rescue robots. Nevertheless, this indirect kind of HRI via an operator station involving semi-autonomy and remote control is different to what is required in most DSR tasks, where the focus is on direct and more natural interaction and on full autonomy. Still, one can think of certain DSR applications where such kind of interaction is desired, e.g. to monitor and to communicate with nursing cases remotely.

RoboCup@Home The second league under the roof of RoboCup that is oriented towards a specific application is RoboCup@Home which we presented in Section 2.1.2 already. It was proposed in 2005 (van der Zant and Wisspeintner, 2005) and it was accepted as a league for the 2006 world championship for the first time. It specifically targets the application of autonomous mobile robots as assistive technology in socially relevant tasks in home environments (van der Zant and Wisspeintner, 2007). The general idea is to test robots in a complex environment that is as close to real-world scenarios as possible. Starting with simpler tasks at first, the difficulty is increased step by step, but the environment is preferably kept at the highest level of complexity possible with as many variations to expect and to enforce as possible. It is designed to be both, a scientific competition and a benchmark (Wisspeintner et al., 2009). RoboCup@Home has grown to be the largest benchmarking effort for domestic service robotics since its start in 2006 (van der Zant and Iocchi, 2011).

2.2.2 Developing Personal Service Robots

When developing domestic service robots not only technical aspects are important but the social component has to be considered as well. It has early been found that computers (and robots are embodied computers) are considered as social actors by humans already if they exhibit simple human-like traits (Nass et al., 1994). Social aspects of the interaction itself should also not be underestimated (Severinson-Eklundh et al., 2003). It has been found that people prefer to consider a robot companion to be a servant rather than a friend (Dautenhahn et al., 2005). Also, human-like communication is more important than human-like behaviour and/or appearance. An effort to create a typology of signals and cues that are meaningful in social robotics can be found in (Hegel et al., 2011). Dautenhahn (2013) claims that, to achieve successful and/or natural HRI, domestic service robots do not have to be humanoid, though.
Service robots for personal and domestic applications have been the subject of research efforts in the last millennium already. Hanebeck et al. (1997) discuss design issues and key components of a robot used for automation in domestic and health care applications. Robots were also already used as tour guides in public spaces such as RHINO (Burgard et al., 1999) in the “Deutsches Museum Bonn” and MINERVA (Thrun et al., 2000) in the Smithsonian museum in Washington. They could safely navigate in dynamic human-populated environments and they could successfully engage in short-term interaction with humans. A famous example of a humanoid service robot is Honda’s ASIMO[8] which has been used, for example, to work as a robotic receptionist (Sakagami et al., 2002). Hüttenrauch and Severinson Eklundh (2002) investigate the use of a robot to aid partly motion-impaired persons with fetch-and-carry tasks in an office environment. Multi-modal human-machine interaction has been integrated to a prototypical interaction cycle for a service robot in a home store (Böhme et al., 2003). Falcone et al. (2003) study the design of a personal rover for long-term interaction in a domestic scenario. The development of the Care-O-bot[9] as an intelligent home assistant started with a mobile platform with a touch screen. Then, a manipulator was added on the Care-O-bot II (Graf et al., 2004). Finally, different kinds of interaction were integrated (Parlitz et al., 2007) on the platform Care-O-bot 3 (Graf et al., 2009). DIARC (Schermerhorn et al., 2006a,b) is a robot that took part in the 2006 AAAI Mobile Robot Competition. It is capable of natural human-robot interaction including features such as emotion expression and recognition and natural language understanding (Brick and Scheutz, 2007).
With the goal of driving the development in domestic service robotics, RoboCup@Home attracts many researchers to present and evaluate their approaches in the scope of the competitions. There are too many teams to mention all of them. We can only present a selection to give an overview of the broad scope of research topics that is covered by @Home participants. The UChile Homebreakers (Correa et al., 2014) focus on HRI (Ruiz-del Solar et al., 2010) and evaluate their robot for education and in public spaces as well. Team homer@UniKoblenz (Seib et al., 2014) looks into building a versatile

[8]http://asimo.honda.com/
[9]http://www.care-o-bot.de/

architecture (Thierfelder et al., 2011) and explores hierarchical multi-robot coordination (Seib et al., 2011). They also provide a facial display with lip-syncing (Seib et al., 2013). Team eR@sers (Okada et al., 2014) does research, among others, in combining speech processing and object manipulation (Nakamura et al., 2011) and in creating a simulation for @HOME (Inamura et al., 2014). Team NimbRo (Schwarz et al., 2014) has a slight focus on mobile manipulation (Stückler et al., 2013), however, they also integrate intuitive human-robot interaction. Team b-it-bots (Breuer et al., 2012) researches a control architecture, multi-modal HRI as well as simultaneous localization and mapping (SLAM). Team ToBi (Ziegler et al., 2014) started with building on earlier work on the robot BIRON (Wrede et al., 2006). They are especially interested in human-robot interaction and reusable behaviours (Siepmann et al., 2014). Team WrightEagle@Home (Chen et al., 2014) want to develop high-level cognitive functions for a service robot (Chen et al., 2010). They also look at evaluating domestic robots in simulation (Ji et al., 2013), at extracting knowledge via NLP (Xie et al., 2012) and at using meta-reasoning for HRI (Chen et al., 2013).

Of course, the development of service robots has also continued outside of ROBOCUP-@HOME. For example in the DESIRE project,[10] the *Germany Service Robotics Initiative*, which is a collaborative project to foster development in the field of service robotics in Germany (Plöger et al., 2008). A collection of results are presented in (Prassler et al., 2012). The two mobile manipulation platforms HERB (Srinivasa et al., 2010) and its successor HERB 2.0 (Srinivasa et al., 2012) were developed at Carnegie Mellon University to perform useful tasks for and also with humans in human environments. Beetz et al. (2008) present the *Assistive Kitchen* as a research scenario for technical cognitive systems. The tasks orient towards house keeping and the scenario poses several research challenges for service robotics. Continuing in the application of a robot as a tour guide, Faber et al. (2009) investigate a humanoid museum tour guide with a special focus on intuitive multi-modal interaction. It is the successor of the communication robot Fritz (Bennewitz et al., 2007). Both are humanoid robots that engage in communication with multiple humans, mimicking human ways of interacting like gestures and gaze.

With the ever increasing availability of hard- and especially software (Goth, 2011) personal robots for domestic environments get more and more capable. Examples here are the robot PR2[11] from Willowgarage and the widespread use and contributions to the open source collection of robot software libraries in the robot operating system ROS.[12] Bohren et al. (2011) report on their experiences with using a PR2 as an autonomous robotic butler for fetch-n-carry task. They use hierarchical concurrent state machines for the task-level execution system. Ciocarlie et al. (2012) report on work with a PR2 where humans use the robot for mobile manipulation. The robot's level of autonomous operation could be varied between full teleoperation and autonomous execution of certain sub-modules. The latter proves to be superior to teleoperation in complex and cluttered environments.

Many of the robotic systems presented in this section perform well in their particular application. Some tackle the hard challenges in complex naturalistic or real-world environments even though this can still be problematic. In such scenarios deliberation

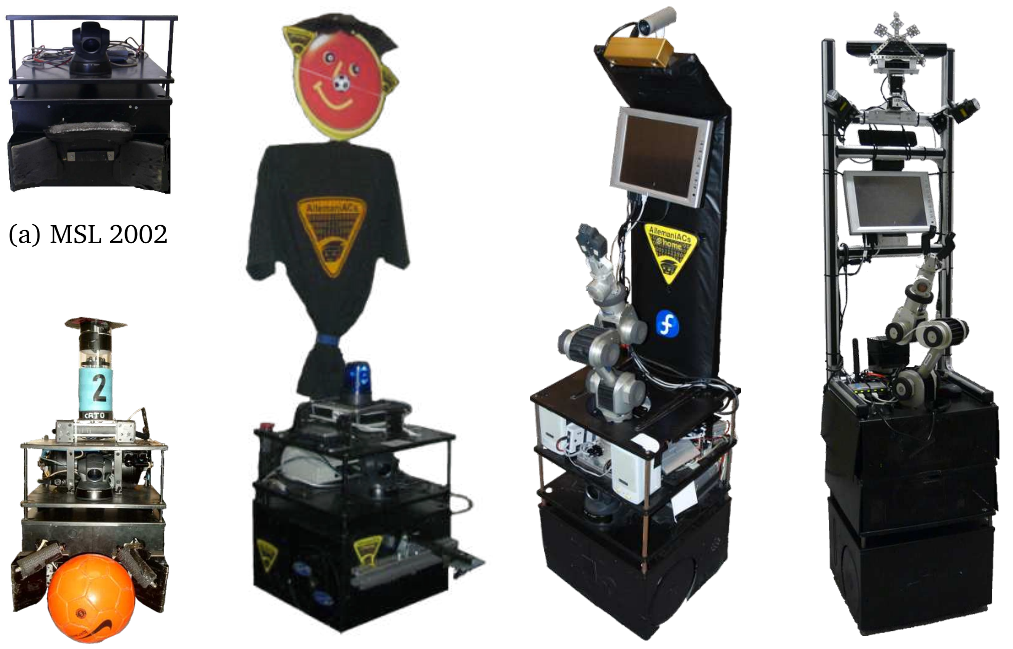
[10] `http://www.service-robotik-initiative.de/uebersicht/?lang=en`
[11] `http://www.willowgarage.com/pages/pr2/overview`
[12] `http://www.ros.org/`

and high-level control comes into play and allows a robot to solve complex missions. In this thesis we aim for a solid base system whose capabilities allow to accomplish the basic tasks in interactive service robotic scenarios. But we think that an assistive service robot in domestic environments can be even more helpful if it is able to take some of the cognitive load off of its human user. This is a particular benefit for assisting elderly or disabled people. This is why this thesis focuses on the high-level control and the reasoning capabilities of our robot and on their integration with human-robot interaction.

2.2.3 High-level Control

There are various ways to achieve intelligent behaviour for autonomous mobile service robots some of which were already mentioned in the previous section. For a lot of specific sub-tasks, statistical methods are used. However, for the high-level control sometimes plan-based approaches come into play.
Some existing work uses formal languages to represent plans and control actions (Beetz, 2001) and to integrate the high-level control with human-robot interaction components (Beetz et al., 2001). Well-founded formal basics allow for open-ended application of personal robots in human environments such as for every-day manipulation tasks (Beetz et al., 2010). The robot can cope with incomplete task specifications by filling the gaps and detailing abstract parts of the task on its own. To be able to cope with execution failures Lemai and Ingrand (2004) propose a framework for interleaving temporal planning and plan execution for robots. They are able to perform online plan repair and execution control based on temporal constraints. Their motivation is that “taking into account run-time failures and timeouts” requires online plan recovery.
Alami et al. (2005) report on how they include the presence of humans in their robot control. Their robot is able to account for humans in the environment in its decision making in order to, for example, not interfere with a human watching TV by driving in the line of sight. The approach reported on in (Brenner, 2007) uses classical AI planning techniques to determine a course of action to execute human user commands with taking into account the current situation. It can also revise its plan depending on updated perceptual input.
Stückler and Behnke (2009) lay out their development of a domestic service robot platform. They present their robot along three main issues: navigation, manipulation and communication. They focus on their robot’s communication skills and do not go into too much detail on their high-level control mechanisms. Breuer et al. (2012) present their service robot for domestic domains and they detail the components of their system. It also features reasoning capabilities using an ontology and HTN planning. Chen et al. (2011) develop a service robot that uses a broad spectrum of AI techniques. To integrate natural language processing with action planning they use answer set programming (Chen et al., 2009). To improve on their robot’s abilities they also use commonsense reasoning (Chen et al., 2010) and non-monotonic reasoning (Ji and Chen, 2011).

The approaches mentioned above present a selection of feasible solutions to particular challenges and some have favourable properties in their integration efforts. In this thesis, we follow a logic-based approach in our high-level control. Building on existing work (Ferrein and Lakemeyer, 2008) we extend our controller by allowing for using human-oriented representations and control. Also, we propose extensions for increasing


(a) MSL 2002

(b) MSL 2006 (c) @Home 2006 (d) @Home 2009 (e) Caesar in 2011

Figure 2.2: Evolution of the hardware platform of Caesar from the initial base platform used in the RoboCup soccer mid-size league (MSL) in 2002 to its later appearance used in RoboCup@Home 2009 and in 2011

robustness by some form of self-maintenance and flexibility in interpreting possibly faulty commands given to the robot by human users.

2.3 The Domestic Service Robot Caesar

The contributions of this thesis are implemented, deployed and tested on a mobile robot platform called Caesar. It is a former soccer playing robot of the AllemaniACs RoboCup team. Starting in 2006, over the years we re-fitted the platform towards domestic service robotic tasks and extended it to participate in the @Home competition. The evolution of Caesar is shown in Figure 2.2.

Hardware

The mobile robot Caesar that we developed for (domestic) service robotics and use in the RoboCup@Home competitions is based on the platform that was initially designed for the AllemaniACs RoboCup Middle Size League Team (Wunderlich and Dylla, 2002) and that was used in the soccer competitions until 2006. It received several improvements dedicated to the specific requirements of domestic service robotics since then.
Caesar's base platform has a size of $40\,\mathrm{cm} \times 40\,\mathrm{cm} \times 60\,\mathrm{cm}$ (Figure 2.2). It is driven by a differential drive, the motors have a total power of $2.4\,\mathrm{kW}$ and are originally developed for electric wheel chairs. The power for the motors is supplied by two $12\,\mathrm{V}$ lead-gel accumulators with $15\,\mathrm{A\,h}$ each. The battery power lasts for approximately one hour

continuous motion at full charge. This power provides us with a top speed of 3 m/s and 1000 °/s at a total weight of approximately 60 kg. Our main sensor for navigation and localization is a 360° laser rangefinder (LRF), a Lase ELD-L-A[13], with a resolution of 1° at a frequency of 10 Hz. For communication a WLAN adapter capable of using IEEE 802.11a/b/g is installed.
Starting with robotic soccer, for which the basic platform (Figure 2.2a) was initially designed, we modified the hardware platform and extended the actuators and computing power to meet the requirements of the domestic environment. One of the major modifications was the installation of an anthropomorphic *robotic arm* called Katana6M180[14] from Neuronics which we use for manipulation tasks since 2007. The Katana is equipped with six motors providing five degrees of freedom. The arm's weight is around 4 kg and it has a maximal payload of 500 g. The arm is mounted on top of the mobile robot platform described above. To provide the arm with the required power, we mounted two additional 12 V NiMH accumulators on the robot.
Together with the arm we added an aluminium rack to mount additional items such as stereo speakers to play sounds and to generate speech output as well as a ten inch touch screen to display information and to receive touch-based commands. The rack increases the total height of CAESAR to 180 cm. After a first version was added in 2007 we completely re-designed the rack for more flexibility in 2011. On the very top of the robot we installed a Microsoft *Kinect* sensor. Replacing a stereo camera for the same purposes, it is used for visual servoing of the arm as well as for face, gesture and object recognition. The camera has a field of view of 57° horizontally and 43° vertically. It can be tilted in a range of ± 27°. The camera provides 32-bit color images with a resolution of 640×480 and 16-bit depth information with a resolution of 320×240 both at 30 frames/s. The depth sensor ranges from 1.2 m to 3.5 m. The Kinect further also provides a 16-bit audio stream at 16 kHz.
To increase flexibility and to better direct the robot's gaze for a particular task, the camera is mounted on a self-assembled *pan-tilt unit (PTU)*. To further improve on our sensing capabilities we additionally installed two URG-04LX-UG01 laser rangefinders from Hokuyo.[15] They provide distance data in a range of up to 5.6 m in a 240° scan window at a resolution of 0.36° with 10 scans per second. To meet the increased demands in computational power for vision, manipulation and deliberation we installed two Intel® Core™2 Duo computers running at 2 GHz with 2 GB RAM each.
All these different components form a complex robotic system. The final assembly and a decomposition with all the wiring is shown in Figure 2.3.

Software

We started off with a software framework called RCSOFT that was used in the ROBOCUP MIDDLE SIZE soccer competition. It features a blackboard architecture where any component can post data to a blackboard that then the other components can read from. Since 2009 we started slowly migrating our software components to the Fawkes[16] robot framework (Niemueller et al., 2010a). Also, we started to port the mid-level behaviours

[13] http://www.lase.de/produkte/2dscanner/eld_l_a/en.html
[14] http://www.neuronics.ch/cms_en/web/index.php?id=244
[15] http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html
[16] http://www.fawkesrobotics.org

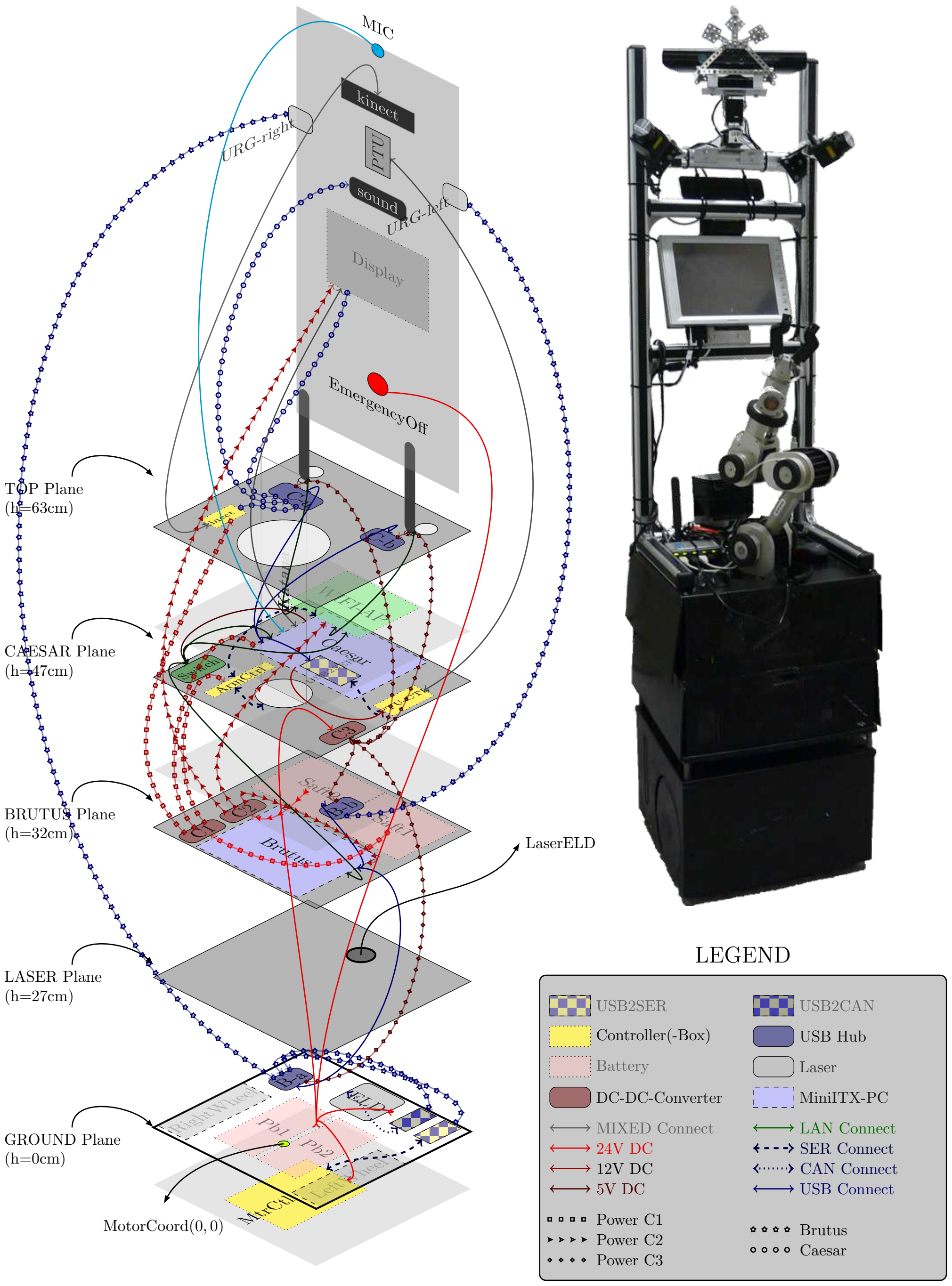


Figure 2.3: Caesar, the AllemaniACs' domestic service robot and its wiring

that were previously implemented as state machines to a Lua-based behaviour engine (Niemueller et al., 2010b).

The primary research focus has been on the high-level behaviour specification. The ALLEMANIACS have been using (variants of) GOLOG for high-level reasoning for the soccer domain (Dylla et al., 2002b). A key feature was to use online decision-theoretic planning (Ferrein et al., 2004) for the high-level control of synthetic and robotic soccer agents (Ferrein et al., 2005). The result of these efforts in logic-based robot control is READYLOG (Ferrein and Lakemeyer, 2008), a GOLOG dialect, which integrates several existing extensions to the original GOLOG language to meet the requirements of dynamic real-time domains such as robotic soccer.

For our efforts in the domestic service robotics domain we stick to the focus on high-level reasoning and we will use READYLOG for the high-level control of our robot CAESAR. However, the new application in interactive domestic domains demands for also extending the low- and mid-level capabilities. For example, feasible means for human-robot interaction have to be established. Also, bridging the gap in the different representations used by human and by robots is our aim, especially with respect to the high-level control. For the remainder of this chapter we first focus on the fundamental component that provides our robot with safe and collision-free navigation. Then, we review a set of further basic components which are needed in domestic service robotics but which are not in the focus of this thesis.

2.4 Collision Avoidance and Navigation

One of the most important and most basic tasks for a mobile robot, especially in domestic domains, is safe navigation with reliable collision avoidance. An earlier approach (Dylla et al., 2002a) was not fully feasible for domestic environments. We present our navigation and collision avoidance algorithm which we successfully deployed in the domestic robot domain. The navigation scheme was one of the key features for our success in the domestic robot competition ROBOCUP@HOME.

Our method relies on a distance measurement sensor from which a local map of the surrounding is constructed. In our case, we make use of the laser rangefinder installed on CAESAR. In this local map, which is in fact a grid representation (Moravec and Elfes, 1985), we search for a path to a target point. We use A^* search (Hart et al., 1968) for this, a best-first-search algorithm for finding least-cost paths that uses a heuristic to estimate the remaining costs to reach a goal. The calculated path serves as an initial solution to the navigation task. To compute the path the robot's kinematic constraints are not taken into account yet. This is decoupled in order to decrease the size of the search space. We integrate it in a second step where we construct the so-called *collision-free triangle*, where the path as it is calculated by A^* serves as the leg of this triangle. In particular, the current setting of the robot's parameters like its speed and its orientation are taken into account. By this, we explicitly take care of the robot's kinematic constraints. In the sequel, we prove that this triangle is obstacle-free and can be traversed safely.

The success of our method is founded on two ideas. (1) The first one is to represent the size of the surrounding obstacles depending on the speed of the robot, i.e. the faster the robot drives the larger the obstacles will become, since the robot needs more time to break in front of them. This way we can represent the robot as a mass point which

simplifies the search for a path. (2) The second idea lies in decoupling the search for a path from its realization. In particular, we propose to construct a collision-free triangle which the robot can traverse safely.
In the past, many different approaches for this fundamental problem have been proposed. Our approach goes beyond other collision avoidance methods in at least three points:

1. We believe that extending the size of the obstacles depending on the speed of the robot is innovative and worth to be mentioned; with this the robot drives only as fast as possible not to collide with any obstacles. In narrow passages it reduces iteratively its speed until it can safely travel through, while in broad areas it will try to reach its maximal speed.
2. With the collision-free triangle we have an area in the motion area of the robot which is guaranteed to be collision-free.
3. Finally, it has proven its applicability in real-world scenarios for many years. We deployed the approach in robotics competitions and it was a key to succeed in the first years of the domestic robot competition RoboCup@Home.

2.4.1 Related Work

Approaches to mobile robot navigation can be categorized along several criteria. Some approaches make use of randomized planning techniques, e.g. (Kavraki et al., 1996; Petti and Fraichard, 2005), other approaches use reactive schemes, for instance (Fox et al., 1997; Khatib et al., 1997; Fiorini and Shiller, 1998), and/or make use of a navigation function to follow a path like (Brock and Khatib, 1999) or plan directly in the velocity space like (Borenstein and Koren, 1991; Ulrich and Borenstein, 2000; Large et al., 2002; Stachniss and Burgard, 2002). Yet other approaches employ a search, some in physical space, some in configuration space.
In their early approach, Borenstein and Koren (1991) proposed to use vector field histograms. The target point exerts an attractive force to the robot while obstacles impose repulsive forces. The trajectory of the robot is then formed by the sum of both these forces. They propose a special wall-following mode to avoid getting stuck in local minima which could otherwise cause oscillating behaviour. The method was tested with robots equipped with sonar sensors driving with an average speed of 0.58 m/s.
Fox et al. (1997) proposed the dynamic window approach. It is directly derived from the motion equations. In the velocity space circular collision-free trajectories are searched. To handle the state space they define a *dynamic window* around the robot to only consider those velocities that the robot can reach in the next time interval. Finally, a trajectory is found by maximizing over the minimal target heading, maximal clearance around the robot, and maximal speed. The method was tested on an RWI B21 robot with a maximum speed of 0.95 m/s. A similar approach except for the dynamic window was proposed in (Simmons, 1996).
Seraji and Howard (2002) describe a behaviour-based navigation scheme. They distinguish between different terrain types such as roughness, slope, and discontinuity. A fuzzy controller selects between traverse-terrain, avoid-obstacles, and seek-goal behaviours. While their method aims at outdoor navigation, it is an example for a local reactive navigation scheme. Another reactive approach is presented in (Minguez and Montano,

2004). The difference to our work is that we use an optimal path as an initial solution to avoid nearby obstacles.

Besides range sensors, imaging sensors are commonly used to navigate a mobile robot. In (Murray and Little, 2000) an approach to build an occupancy map from a stereo camera on an RWI B14 robot is presented. The map is used to navigate through previously unknown environments. We want to note that our method is different in the sense that we here present a reactive local method while the focus of Murray and Little is on vision-based exploration. A large number of other papers deals with navigation approaches using several sensors. Among those, the fusion of imaging and proximity sensors (cameras and LRFs) is popular, see for example (Asensio et al., 1999; Dedieu et al., 2000; Wijesoma et al., 2002).

Koenig and Likhachev (2002) present a search heuristic called Lifelong Planning A*. They propose an incremental search heuristic where only the relevant parts of a path are recalculated. While Lifelong Planning A* is an interesting extension to the basic A* we use in this paper, we here focus on the obstacle representation and the decoupling of path and velocity planning to decrease the dimensionality of the search problem.

The most related research to our approach is the method proposed by Stachniss and Burgard (2002). They use a laser rangefinder as sensor and employ A* to find a shortest trajectory to a given target. The state space used here is a five-dimensional pose-velocity space consisting of the pose x, y, θ and the translational and rotational velocities υ, ω. At first, a trajectory to the target is calculated using A* in the $\langle x, y \rangle$-space. This trajectory serves as the starting point for the search in the pose-velocity space. With a value iteration approach a $70\,\mathrm{cm}$ broad channel around the calculated trajectory is calculated which restricts the state space for the five dimensional search. Stachniss and Burgard tested their approach on a Pioneer I and an RWI B21 both having a maximum speed below $1\,\mathrm{m/s}$. We remark that the approach in (Stachniss and Burgard, 2002) is especially interesting as it does not have problems with U-shaped objects (as opposed to (Borenstein and Koren, 1991)) or narrow doorways (as opposed to (Fox et al., 1997)). In our approach, by restricting the search for velocities to the collision-free triangle which is known to be obstacle-free, we are able to avoid the search in the five dimensional pose-velocity space which leads to a much more reactive navigation and collision avoidance scheme. A similar method applied to holonomic small-size league robots in the robotic soccer domain can be found in (Bruce and Veloso, 2006).

2.4.2 The Navigation Algorithm

In this section, after introducing the kinematics of our robot and reviewing the modelling of the environment, we present the single steps of our navigation algorithm.

2.4.2.1 Kinematics of the Service Robot Platform

We described our hardware platform in Section 2.3 already. We assume accelerated motion with our approach. Thus, the connection between pose, velocity and acceleration is given by $x(t) = \frac{1}{2} \cdot \ddot{x}(t) \cdot t^2 + \dot{x}(t) \cdot t + x_0$, $\dot{x}(t) = \ddot{x}(t) \cdot t + v_0$, and $\ddot{x}(t) = const.$ As usual, $\ddot{x}(t)$ refers to acceleration, $\dot{x}(t)$ to the velocity, and $x(t)$ to the displacement at any given time t. In the algorithm we present in the following, we assume that at each time instance the robot is accelerated only in the first time instance and from there on is

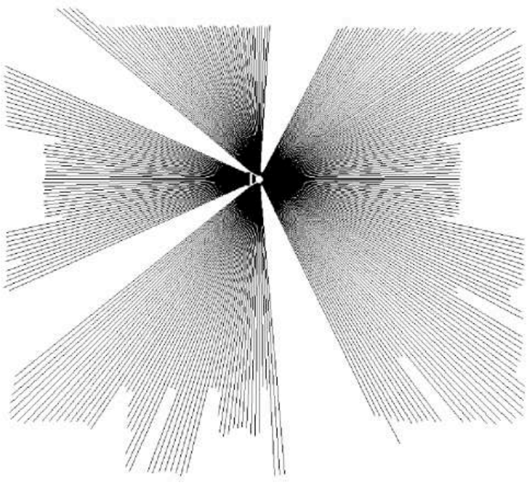


Figure 2.4: The robot's sensor data used for collision avoidance. The four blind regions are due to the rods supporting the platforms above the laser rangefinder.

driving with constant velocity till the next acceleration command is settled. We need this relation because, for efficiency reasons, we decouple the search in the pose-velocity space into a pose space and a velocity space in our algorithm.

2.4.2.2 Representation of the Environment

The task of the navigation algorithm is to steer the robot on a collision-free path from its current location to a given target point. The algorithm we present does not rely on a global map as many other algorithms do but on a local map of its environment. The dimension of the local map corresponds to the area covered by the current reading from the LRF. It is updated every time new laser-readings are received. Although, in our implementation, we integrate new sensor readings into the previous local map if possible, we here assume that the local map is set up from scratch every cycle. In juxtaposition to approaches which rely on a global map a local map has the advantage that it allows to easily account for dynamic obstacles. Moreover, using a local map makes the successful execution of a planned path independent from the localization of the robot. In the following we will give a rough overview of our navigation algorithm and discuss the key aspects in greater detail thereafter.

If the robot is in close proximity to the given target, i.e., the target is reached, it stops. Otherwise the current distance readings from the LRF and the current translational and rotational velocities are obtained. This data is then used to build a (local) map. For this we employ a technique we refer to as *dynamic obstacle extension* (cf. Section 2.4.2.3) which yields a (local) grid-based map of the robot's environment. The cells of this occupancy grid (Moravec and Elfes, 1985) map may either be occupied or free.

With this representation of the surrounding of the robot we search for an initial path to the target first. In a second step, we search for an approximation of the initial path which takes into account the kinematic constraints of the robot. In both steps we employ the A^* search algorithm. Splitting up the path-planning problem into two independent search problems reduces the original problem of dimensionality four to two search problems over two-dimensional search spaces. Namely, finding a path in the xy-plane

Algorithm 1: The navigation algorithm in pseudo-code

```
Input: $\Delta x, \Delta y$ the target point in relative coordinates
while not reached target do
    $d_1, \dots, d_n \leftarrow$ getLaserReadings() ;
    $v_t^{cur}, v_r^{cur} \leftarrow$ getCurrentVelocities() ;
    $map \leftarrow$ extendObstacles($d_1, \dots, d_n, v_t^{cur}, v_r^{cur}$) ;
    $path \leftarrow$ findInitialPath($map$) ;
    if $path.isEmpty()$ then
        sendMotorCommands(0, 0);
        break;
    end
    $v_t, v_r \leftarrow$ findVelocities($path$, $map$) ;
    if no $v_t, v_r$ then
        sendMotorCommands(0, 0);
        break;
    end
    sendMotorCommands($v_t, v_r$) ;
    $\Delta x, \Delta y \leftarrow$ updateTarget($v_t, v_r$) ;
end
sendMotorCommands(0, 0) ;
```

and appropriate velocities v_t, v_r. This is only possible since the search for an appropriate approximation of the initial path is restricted to a certain area which is guaranteed to be free of obstacles. We refer to this area as the *collision-free triangle*. More details on this are given in Section 2.4.2.4.

2.4.2.3 Dynamic Obstacle Extension

A technique often referred to as *obstacle growing* (Meystel et al., 1986) extends the obstacles by the dimensions of the robot. This alleviates the problem of collision detection in the path-planning problem since the robot can now be treated as a mass point. We leapfrog on this idea and additionally extend the obstacles in dependence on their respective imminence of collision which takes into account the current speed of the robot as well as the position of the obstacle relative to the current trajectory of the robot. The intuition behind this is to mark potentially dangerous areas as occupied in the local map and thereby force the search to not consider paths leading through those areas.
The most threatening obstacle for the next step is the obstacle lying on the trajectory defined by the current and the projected position of the robot in the next step. We assume to recompute a new path every iteration and, consequently, it is not necessary to project further into the future then the next step. The next-step trajectory is computed from the current translational velocity v_r, the current rotational velocity v_r and the time between two iterations Δt:

$$v_x = \frac{x(t+1) - x(t)}{\Delta t} \qquad v_y = \frac{y(t+1) - y(t)}{\Delta t} \qquad \alpha = \tan \frac{v_y}{v_x}$$

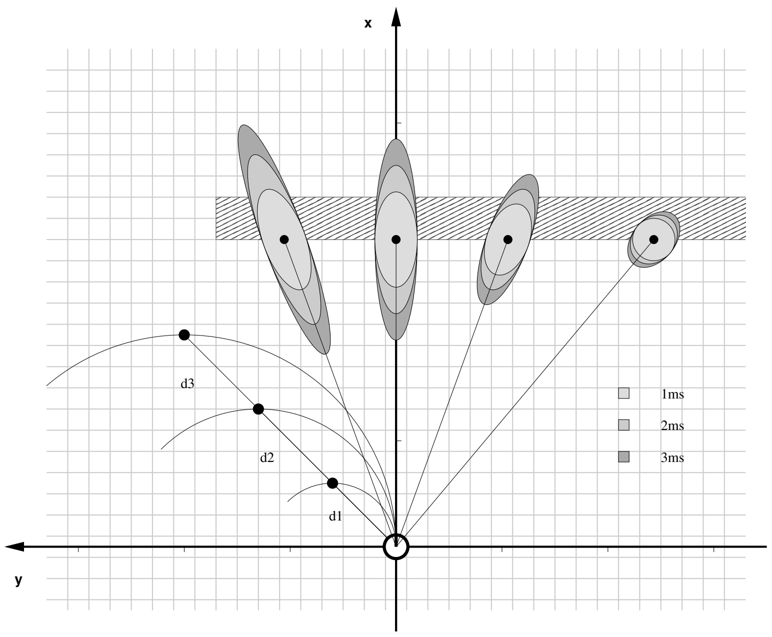


Figure 2.5: In this illustrating example a wall is located in front of the robot. The extension of the obstacles is shown for the obstacles detected at 20°, 0°, -20°, and -40°. The translational velocities are 1 m/s, 2 m/s, and 3 m/s; the rotational velocity remains fixed at 1 rad/s. For illustrating purposes we chose $\Delta t = 0.25$ s and $n = 1$.

For each detected obstacle we place an ellipse centered at the reflection point of the laser beam in the local map and align the axes such that the semi-major axis is parallel to the laser beam. The radius for the semi-major axis r_1 and the radius for the semi-minor axis r_2 are computed as:

$$\begin{aligned} r_1 &= l + l_{sec} + |\cos(\theta - \alpha)| \cdot d \cdot n \\ r_2 &= l + l_{sec} \end{aligned}$$

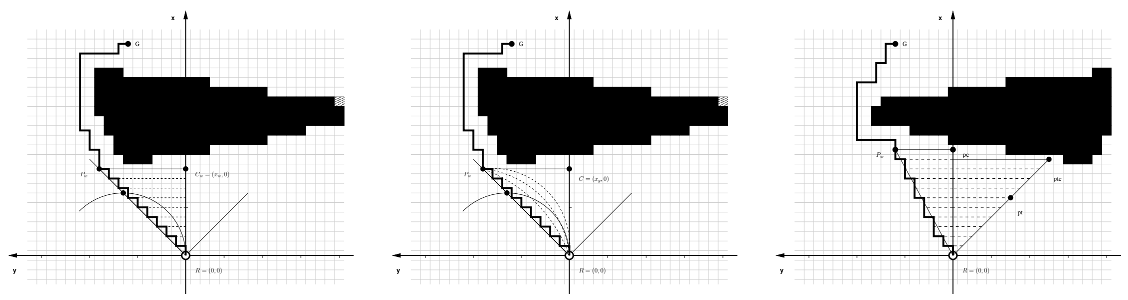
where l is the radial extension of the robot[17], l_{sec} is an additional security distance, θ is the angle of the laser beam that hits the obstacle and d is the euclidean distance between the current position and the position projected for the next step

$$d = \sqrt{(v_x \cdot \Delta t)^2 + (v_y \cdot \Delta t)^2}.$$

Then, the obstacles are extended in such a way that the robot will stay out of the "dangerous area" for the next n steps.

By means of extending the obstacles in such a way we capture the current obstacle configuration as well as the current configuration of the robot (in terms of translational and rotational velocity) in the local map. Figure 2.5 illustrates the extension of the obstacles for different configurations of the robot: the rotational velocity remains fixed whereas the translational velocity is altered between 1 m/s and 3 m/s.

[17]The formula could be further detailed to account for a rectangular shape of the robot, but this is omitted here for clarity reasons.



(a) Triangle construction (b) Finding a trajectory inside the triangle (c) Triangle for opposite rotational velocity

Figure 2.6: Construction of the collision-free triangle

2.4.2.4 The Collision-free Triangle

As we pointed out in the introduction, within the search for an initial path we ignore the kinematic constraints of the robots as well as its current configuration in terms of velocity and orientation. With this, we are in good company with several other search-based methods like Stachniss and Burgard (2002). However, to successfully realize a path on a real robot, the kinematic constraints need to be taken into account, of course. In our algorithm, we do so by performing an A* search on the possible accelerations in translation and rotation making use of the standard motion equations for accelerated motion. The kinematic constraints of the robot are only one part of the velocity planning problem of the robot. The other part is to take the robot's current state into account, i.e. its current translational and rotational velocities. In the following, we therefore present the collision-free triangle. We prove that, by construction, each grid cell inside this triangle is free of obstacles. Poses inside this triangle are thus safe and therefore it can be used to restrict the search for the optimal trajectory the robot should follow.

Definition 2.4.1 (Collision-Free Triangle) *The robot is located in the origin of a Cartesian coordinate system, such that $R = (0, 0)$ is the position of the robot facing the positive x-axis. Let $C_k = (x_k, 0)$ be a grid cell in front of the first obstacle along the x axis. In case there is no obstacle along the x axis, the last cell within the perception range of the robot is taken.*
Let $p = \langle(0, 0), (x_1, y_1), \ldots, (x_{g-1}, y_{g-1}), (x_g, y_g)\rangle$ be the path to the target point G. The path is given as a sequence of grid cells. For each path point $P_i = (x_i, y_i)$, $1 \leq i \leq g$ we ray-trace to the point $C_i = (x_i, 0)$. A path point (x_i, y_i) is called safe *iff the ray $\overline{P_i\, C_i}$ does not hit any obstacle, otherwise it is unsafe. The* collision-free triangle *is now given by the points R, P_w, and C_w with w being the index of the last safe path point.*

Figure 2.6 depicts the construction of the collision-free triangle as described in the definition. Note that the x axis always points into the direction of the robot's orientation as the map is given by a Cartesian coordinate system with the robot in its origin. Hence, the point C denotes the last free cell before the robot would collide with an obstacle if, from now on, it would drive only with translational velocities. Now, for each path point it is checked if the orthogonal projection of a path point onto the segment $|RC|$ will hit an obstacle. The robot's position R, the last safe path point P_w and the point C_w, the projection point of P_w onto $|RC|$, yields the corner points of the triangle.

Figure 2.6c illustrates the situation in which the robot is turning away from the path. In that case, we span an additional triangle by projecting the robot's position according to its current velocity for the next step (cf. point D). We put a straight line from the robot's position through D until we hit an obstacle (cf. point D_i). Then we ray-trace analogous to the original triangle to ensure the additional triangle is also obstacle-free.

Theorem 2.4.1 *Each cell inside the triangle is obstacle-free.*

Proof. Suppose, P_i is the next path point to be considered in the construction of the collision-free triangle as described in the definition. Hence, the triangle given by $\triangle R, P_{i-1}, C_{i-1}$ is collision-free. Now we ray-trace from P_i orthogonal to the segment given by $|RC|$. If the ray hits an obstacle, the collision-free triangle is given by $\triangle R, P_{i-1}, C_{i-1}$, otherwise we conclude that the $\triangle R, P_i, C_i$ is collision-free. □

P_w is the next point for the robot to reach (safely). P_w closes in on G and the robot will thus eventually reach the target point.

2.4.2.5 Computing the Drive Commands

Lastly, a suitable sequence of velocities to reach the intermediate target point P_w needs to be determined. Again we employ A* search to find such a sequence. We restrict the search space by only considering velocity sequences which steer the robot to locations within the collision-free triangle – as soon as the robot leaves the collision-free triangle the search is aborted.
The initial state in the search space is $\langle 0, 0, 0, v_t, v_r \rangle$, i.e., the robot is located at the origin of the coordinate system, its orientation is 0°, and the current translational and rotational velocities are v_t and v_r, respectively. Possible successor states in each step are $\langle x', y', \theta', v'_t, v'_r \rangle$ where $v'_t = v_t + c_t \cdot a_t^{max} \cdot \Delta t$ with $c_t \in \{-1, -\frac{2}{3}, \ldots, 1\}$ and a_t^{max} being the maximal translational acceleration. Analogously for v'_r. (x', y', θ') is the projected pose at time $t + \Delta t$ when sending the drive commands $\langle v'_t, v'_r \rangle$ to the motor controller and the robot is located at $\langle x, y, \theta \rangle$ at time t. The change in position and orientation is computed according to the standard equations for differentially driven robots. The heuristic value for each state is computed as the straight-line distance to the intermediate target; the costs are uniform. A goal state is reached if the distance between the projected position and the intermediate goal is smaller than a certain threshold.
The velocities returned by the function "$\mathit{findVelocities}(\cdot)$" in Algorithm 1 are the first translational and rotational velocities in the sequence of velocities that was determined by the search.

2.4.3 Implementation and Evaluation

Occupancy Grid Although the LRF has a far longer range we limited the local map to a size of $6 \times 6\ \mathrm{m}^2$ for practical reasons. The local map is subdivided into grid-cells with a size of $5 \times 5\,\mathrm{cm}^2$. Consequently, the complete local map is made up of 14400 cells. Recomputing the ellipses that result from extending the obstacles and their respective rasterizations with every update is quite costly. Therefore, we pre-computed a library of rasterized ellipses of various sizes and at various angles. For a given obstacle we look

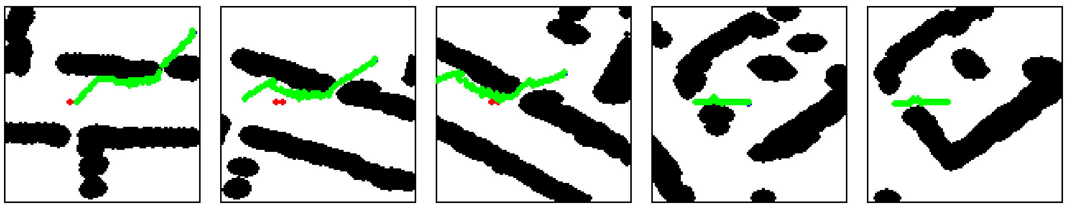


Figure 2.7: Example traces

up the ellipse matching the current velocities of the robot and the angle at which the obstacle is detected and integrate it into the local map.

Searching for the Path In order to avoid that the changed perception of the environment which is due to the movement of the robot leads to an oscillating behaviour we accumulate the sensor readings for a short duration, i.e., we do not only consider the current distance readings but also a number of readings from the past. Each obstacle in this accumulated sensor reading is then extended in the same way described in Section 2.4.2.3. Further, for reasons of efficiency, we try to avoid re-computing a path and proceed with the remainder of the path computed previously, instead. Of course, this is only possible if the “old” path is still a valid path. This means we have to check whether the robot strayed too far from the projected path and whether the collision-free triangle computed for the “old” path is still free of any obstacles. Thus, we still maintain a high degree of reactivity. The implementation of the A^* search algorithm calculates a path of a length up to 300 grid cells (i.e. a path length of $15\,$m) in less than $10\,$ms on the Pentium-III 933 machine on the robot. Given that the frequency of the laser rangefinder is $10\,$Hz and the navigation module runs at $20\,$Hz (not to lose any update from the laser rangefinder) there are about $40\,$ms left for the other steps of the algorithm.

Evaluation The method proposed in this paper was extensively tested during several RoboCup tournaments as well as with indoor demonstrations where the robot had to safely navigate through crowds of people. Figure 2.7 shows the path visualization of a run of a robot in our department hallway. The red dot represents the robot, the green line represents the planned path, the black objects are the walls of the hallway. The robot should navigate from the hallway into a room. Note that the path is calculated in such a way that the shortest possible connection between robot and target is chosen. The second picture in the series shows the robot a few moments later. The calculation of the drive commands and the realization of these commands on the robot have the effect that the robot slightly deviates from the path. We remark that the position of the robot is still inside the collision-free triangle (which is not shown in the figure). In the fourth picture the robot entered the room.
We also tested our navigation algorithm on our B21 robot Carl as shown in Figure 2.8. From the performance point of view we encountered no problems with the algorithm, we tested it with two different maximally allowed velocities (v_{max}). We could observe that Carl reached higher velocities than with the Dynamic Window (DW) approach (Fox et al., 1997) which is part of the original control software of Carl. The DW approach has inherent problems with narrow doorways as well as with relatively sharp turns.

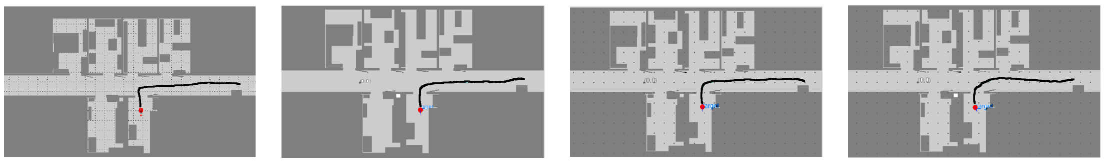

(a) B21 with DWA ($v_{\max} = 0.45$ cm/s) (b) B21 with A* ($v_{\max} = 0.45$ cm/s) (c) B21 with A* ($v_{\max} = 0.95$cm/s) (d) Caesar with A* ($v_{\max} = 3$ m/s)

Figure 2.8: Comparison of the DWA with our method

2.4.4 Summary

We presented a navigation and collision avoidance method for service robots operating in domestic indoor environments. Particularly in these environments, a domestic robot must navigate carefully and be able to drive around obstacles that suddenly cross its path as it is interacting with humans. The core of our method is a grid representation which is generated from the sensory input of a laser rangefinder. Depending on the speed of the robot and several other security parameters, the detected obstacles are extended in the grid representation. This is done to speed up the collision detection when searching for a path to the target. For finding a path, we use A* search on the grid. Next, we construct a so-called collision-free triangle from the obstacle configuration, the current parameters of the robot (its actual speed) and the desired path. For each grid cell inside this triangle we can guarantee that it is collision-free. In a second step, we use this triangle to calculate the drive parameter for the next time step. Again, we employ A* for this task, this time we search for accelerations which keep the robot inside the triangle. The collision-free triangle relates the search for a path to the search for drive commands and allows to decouple both. Positions inside the triangle are safe and therefore feasible. This decreases the complexity of the search problem tremendously. This method allows for fast and adaptive navigation and was deployed for RoboCup@Home competitions over several years without ever colliding with the furniture or humans. For example, we were able to solve the Lost&Found task in 25 seconds while driving through a large part of the apartment looking for an object.

2.5 Further Selected Software Components

In this section we describe a selection of software components that contribute to Caesar being an intelligent service robot for domestic domains but that we will not treat in further detail. This is because discussing every module on the robot would quickly go beyond the scope of the thesis. Modules for human-robot interaction will be detailed in Chapter 4, efforts to bridge the gap in human and robot representations will be the subject of analysis of Chapter 5, and two approaches towards more robustness and flexibility will be presented in Chapter 6.

Localization, Mapping and Path Planning

Beside collision free local navigation it is important that the robot knows where it is. That is, it must have means of global localization given a map of the environment as a reference. Further, path planning is needed for global navigation.

2.5.1 Localization

CAESAR uses a Monte Carlo Localization (MCL) algorithm (Dellaert et al., 1999) which was originally developed for the soccer domain (Strack et al., 2006). It works very well and yet better for indoor navigation even in large environments, though. This is because maps in those domains contain a larger number of structural features that can be used for localization. It works by approximating the position estimation by a set of weighted samples: $\mathbf{P}(l_t) \sim \{(l_{1,t}, w_{1,t}), \ldots, (l_{N,t}, w_{N,t})\} = \mathbf{S}_t$. Each sample represents one hypothesis for the pose of the robot. Roughly, the Monte Carlo Localization algorithm chooses the most likely hypothesis given the previous estimate, the current sensor input, the current motor commands, and a map of the environment. To be able to localize robustly with the laser rangefinder Strack et al. (2006) modified the Monte Carlo approach slightly. To integrate a whole sweep from the LRF we use a heuristic perception model. With this we are able to localize with high accuracy in the ROBOCUP environment. By additionally employing a Novelty filter (Fox et al., 1998) on the laser data in conjunction with some simple clustering methods we can detect dynamic obstacles in the environment, that is, objects which do not belong to the map. These dynamic objects can later be used in other modules, for instance to detect people.

2.5.2 Semantic Mapping

In order to localize itself the robot needs a map of the environment it is to operate in. This map can be constructed by driving around and recording sensor and odometry data. An overview of mapping techniques can be found in (Thrun, 2002). We usually use the mapping tools available in CARMEN (Montemerlo et al., 2003). However, the recording process takes a non-negligible amount of time and it needs to be repeated whenever the environment changes.

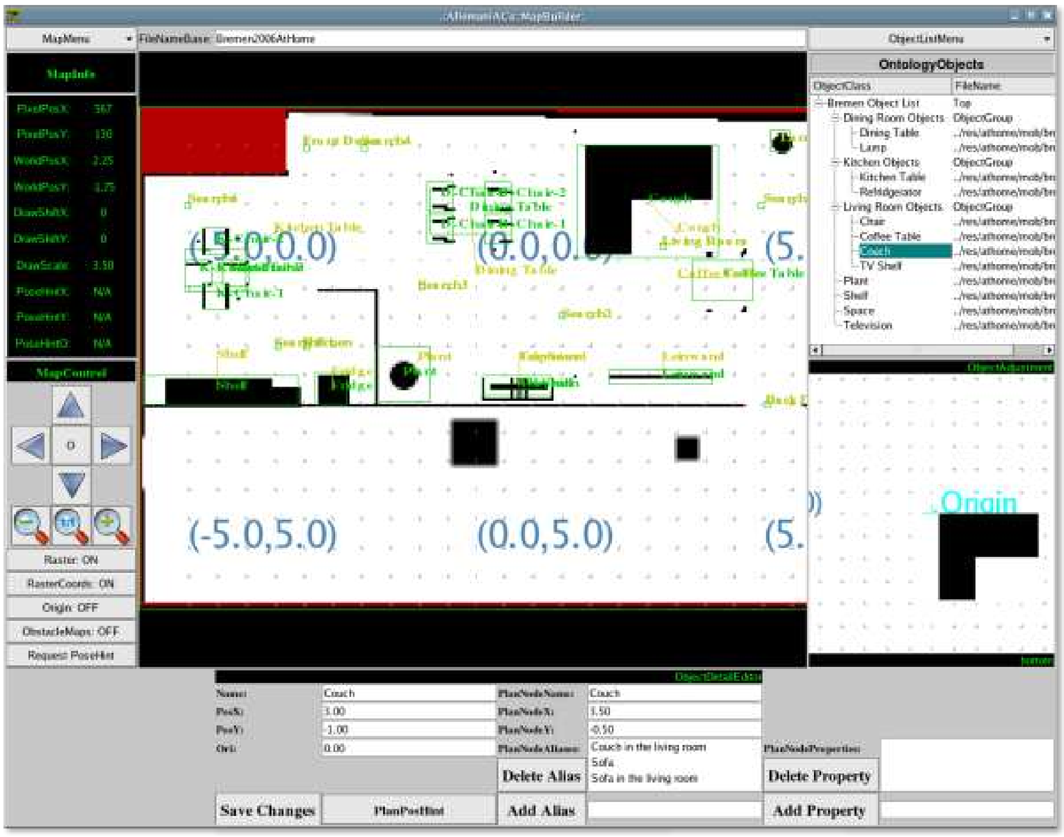
In order to be able to efficiently adapt to the frequent changes which are immanent in a home-like environment we developed a *semantic map building* application. It allows us to update the robot's world representation to the current situation very quickly. Our map builder uses a collection of semantically annotated objects that can be dragged and dropped to their specific location in a base-map. This simplifies the map building process to some few clicks. The objects and the taxonomy that they are organized in are managed in XML-based files. That is, there is a data type definition (DTD) which defines all relevant properties that are associated with an object. A sample definition is depicted in Figure 2.9. In our case it includes a signature of the object as seen by the laser rangefinder, the area to be used in the obstacle map (which we introduce in the next paragraph), and a name along with some common aliases. If additionally a vision system is used one could also include sample pictures of the respective object. The particular information for each object has to be provided beforehand, e.g. the signature of an object as seen by the laser rangefinder has to be drawn or recorded and pictures need to be taken and associated with the object.

Figure 2.10 shows a screenshot of the tool we developed for the map building task. Objects present in the current configuration of the environment can be dragged and dropped to their respective location. When the environment has been set up completely

```
<!ELEMENT MapObject ( Name,
                      MapFileName,
                      ObsFileName,
                      Position,
                      PlanNodeName,
                      PlanNodeAlias* ) >
```

```
<!ELEMENT SEMap ( Name,
                  Location,
                  BaseMapFileName,
                  Room*,
                  MapObject* ) >
```

Figure 2.9: Excerpts from the DTDs of a map object and an overall map file.



FileNameBase: Bremen2006AtHome

PixelPosX:	367
PixelPosY:	130
WorldPosX:	2.25
WorldPosY:	1.75
DrawShiftX:	0
DrawShiftY:	0
DrawScale:	3.50
PoseHintX:	N/A
PoseHintY:	N/A
PoseHintO:	N/A

Raster: ON
RasterCoords: ON
Origin: OFF
ObstacleMaps: OFF
Request PoseHint

OntologyObjects

ObjectClass	FileName
Bremen Object List	Top
Dining Room Objects	ObjectGroup
Dining Table	../res/athome/mob/br
Lamp	../res/athome/mob/br
Kitchen Objects	ObjectGroup
Kitchen Table	../res/athome/mob/br
Refridgerator	../res/athome/mob/br
Living Room Objects	ObjectGroup
Chair	../res/athome/mob/br
Coffee Table	../res/athome/mob/br
Couch	../res/athome/mob/br
TV Shelf	../res/athome/mob/br
Plant	../res/athome/mob/br
Shelf	../res/athome/mob/br
Space	../res/athome/mob/br
Television	../res/athome/mob/br

Name:	Couch	PlanNodeName:	Couch
PosX:	3.00	PlanNodeX:	3.50
PosY:	-1.00	PlanNodeY:	-0.50
Ori:	0.00	PlanNodeAliases:	Couch in the living room Sofa Sofa in the living room

PlanNodeProperties:

Figure 2.10: A screenshot of our semantic map building tool. It shows the creation of the map of the home environment at RoboCup 2006 in Bremen.

several low-level data files are generated, each with all the information required for the task at hand. That is, for localization with a laser rangefinder, for example, the signature of every object is integrated into the occupancy grid used for localization at the corresponding position. If necessary, a corresponding area is added to the obstacle map. Additionally, a node is added to a *topological map* which is needed for path planning (cf. Section 2.5.3). The items in the different low-level data structures are inter-referenced by their name. This way, each module can refer to an object or place by its name in human terminology.
The collision avoidance uses the laser rangefinder as its main sensor. However, the laser can only "see" things at a certain height. This is why we use a so-called *obstacle map* (mentioned earlier) which contains areas that the robot is not allowed to enter but that its sensors cannot detect. Using the robot's position as it is provided by the global localization and this obstacle map, the robot can avoid the areas labelled as obstacles. The idea was borrowed from Burgard et al. (1999).
By providing the robot with semantically enriched objects it is able to make use of any particular part of the information associated with an object. Thus, interaction with humans in the environment can be achieved in a transparent fashion. For instance, when a human specifies a target location for the robot to reach, the name recognized by the speech recognition module is passed to the path planning module which in turn instructs the navigation module to drive to the associated coordinate within the metric map.

2.5.3 Path Planning

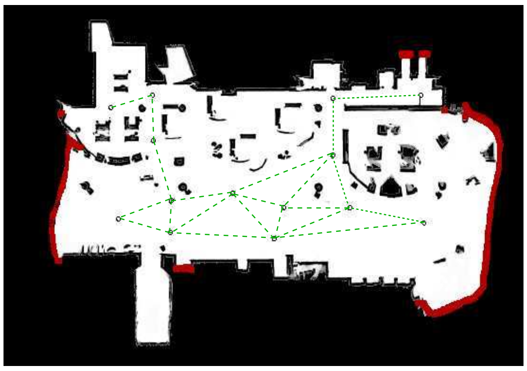
We already mentioned that, on top of the metrical map that is used for localization, we additionally keep a *topological map* that stores information about the positions of and the connections between semantic places like a room or a door. This map can be generated by the semantic mapping tool described in the previous paragraph. To plan a path from one place to another we perform an A^*-search on the topological graph. This yields a list of way-points along which the robot is to navigate to the target place. The local navigation between those way-points is then performed by our collision avoidance method described above. The topological graph in an example scenario is shown in Figure 2.11.
Note that the collision avoidance method only works on a local map that is constructed from the laser data in each cycle. Therefore, collision avoidance works in completely unknown environments while the path planning requires global localization which in turn requires a map of the environment.

Perception and Manipulation

Apart from moving around without collisions and localizing itself a domestic service robot needs to also perceive and manipulate things in its environment.

2.5.4 Object Recognition

In household robotics it is a crucial ability for a robot to be able to learn, recognize and handle various objects. Depending on the sensors available on the robot and the object


(a) Nodes in the environment and connections (dashed green) indicating driveable paths

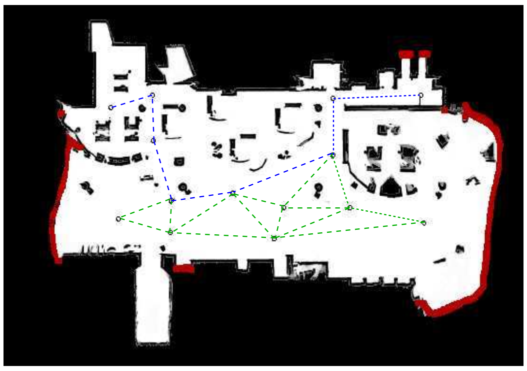

(b) Path planned (dashed blue) from the top left-most node to the top right-most node

Figure 2.11: A map of a large indoor environment with nodes used for path planning

there are different established methods to detect and recognize that object. However, some methods are better fitted for certain types of objects than others, e.g. keypoint-based methods like the *Scale-Invariant Feature Transform* (SIFT) (Lowe, 2004) and *Speeded-Up Robust Features* (SURF) (Bay et al., 2008) mostly only work well on textured objects because they need edges to find interest points and color-based classifiers obviously need objects to be of distinct colors to be distinguishable. If perception is three-dimensional, the *Viewpoint Feature Histogram* (VFH) method (Rusu et al., 2010) might be used. As long as the objects are known, we can simply pick the most appropriate type of classifier to detect and recognize the respective object. To this end, finding a general method to perform object recognition for very heterogeneous data sets is not an easy task.

Traditional object recognition systems use a set of individual objects for training and the task in the recognition phase then is to find one particular individual. Other traditional systems oriented towards classification have a restricted set of classes and the task then is to assign the right class label to an object under consideration. Another possibility that has gained interest recently is to use attributes to describe objects and object classes (Farhadi et al., 2009). These attributes may be properties such as color or material but it may also be components that constitute an object like keys in a remote control or it may even be semantic features such a edibility. This allows to both use properties that several objects share and to still appropriately specify a particular object (if it has enough distinct features). It also enables creating more categories and creating categories more flexibly by simply combining attributes. Moreover, specifying objects in such a way is very much closer to how humans refer to objects in their everyday communication.

To be able to detect and recognize objects specified by a set of attributes we developed a method for flexible object recognition (Schiffer et al., 2013b) a first prototype of which was implemented in (Lakani, 2013). Details of the storage system with regard to this object perception system are described in (Niemueller et al., 2013).

Instead of developing a new (or modifying an existing) method for the purpose of recognizing objects we take a set of existing methods and orchestrate them in a way that yields a flexible system. The underlying idea is pretty simple: To build a system that is strong in the overall recognition we combine the benefits of several existing object recognition methods. We do so by using the method (or a combination of a few such

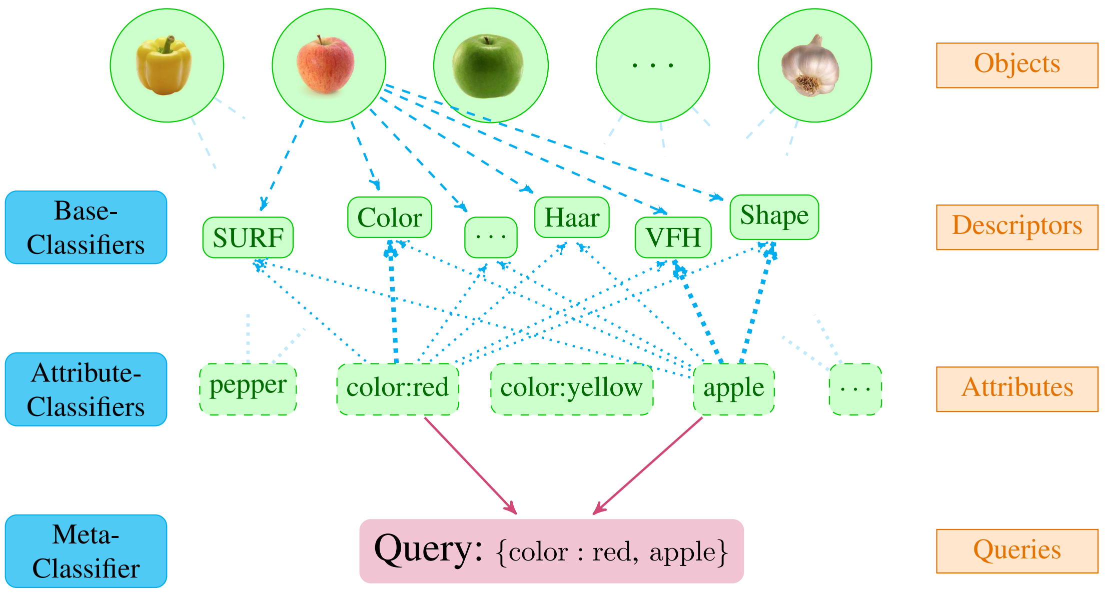


Figure 2.12: Hierarchical composition of classifiers for object recognition. Base-classifiers are built for every descriptor, attribute-classifiers for every attribute, and meta-classifiers are built for certain queries. Although all connections exist between any two successive levels in the hierarchy we reduced the number of connections drawn for legibility.

methods) for every class of objects that is/are able to recognize that particular class best. In our system we organize the classifiers hierarchically. At the first level, we combine individual object descriptors to so-called *base-classifiers*. At a second level, we create classifiers for every attribute by combining all base-classifiers to so-called *single-attribute classifiers* (SAC). At the top most level, we combine the SACs to form what we call a *meta-classifier*, which is built for queries consisting of a set of attributes. The hierarchical composition of classifiers is outlined in Figure 2.12.
In the system like the one sketched above we are able to respond to queries very flexibly. While the base-classifiers and the single-attribute-classifiers are build offline, the meta-classifiers are constructed on-the-fly. This way we are able to answer ad-hoc queries. In order to improve and speed up repeated queries, we are also able to generate improved meta-classifiers offline for queries that occur frequently. A more detailed discussion would go beyond the scope of this thesis and we refer to (Schiffer et al., 2013b) and (Niemueller et al., 2013) instead.

2.5.5 Mobile Manipulation

Manipulation, although important for a domestic service robot, is not our primary concern and it will not be discussed in greater detail in this thesis. In our proof of concept system CAESAR we make use of open source components that are (almost) readily available.
For physical interaction with the world CAESAR can not only move around, but it can also manipulate objects with its robotic arm. To do so, it perceives its surrounding with its RGB-D camera. From the depth information a local model of the scene is generated that is used in the motion planning for the arm. We use OpenRAVE (Diankov and

Kuffner, 2008) to plan collision free motion trajectories for the manipulator to make CAESAR pickup objects and to place them somewhere else. For the inverse kinematics OpenRAVE pre-computes a database so that finding an inverse kinematics reduces to a look-up at run-time. The internal path-planning uses bi-directional *rapidly growing random trees* (RRTs). For the manipulation tasks we encounter in our domestic scenarios, for example for picking up a cup from a table with other objects on it also, the planning of a collision-free trajectory for the arm is done in below five seconds.

2.5.6 High-level Control

The high-level control of our robot uses the logic-based programming and planning language READYLOG (Ferrein and Lakemeyer, 2008), which is a dialect of GOLOG (Levesque et al., 1997). A nice feature of GOLOG and READYLOG is that its semantics is based on Reiter's version of the situation calculus (McCarthy, 1963; Reiter, 2001), a sorted logical language for reasoning about action and change. That means that both languages have a formal semantics and properties of programs can be proved formally. We introduce READYLOG in Chapter 3.

2.6 Discussion

We started this chapter with a description of the domestic service robotics domain and its challenges. Then we presented the ROBOCUP@HOME initiative as a benchmarking effort to evaluate assistive robot technology in domestic environments. We briefly sketched the methodology we helped to create. After reviewing related work we introduced our service robot platform CAESAR. To develop an assistive service robot for domestic environments requires a serious effort in a lot of sub-fields many of which are not the primary focus of our research and this thesis but that still have to be present for a working robotic system. We detailed some of our contributions in this regard in this chapter. For instance, our approach to local navigation is essential since it allows for safely moving around in human environments while avoiding collisions reliably. The semantic mapping is a convenient method to build and to maintain representations of the environment that the robot has to operate in. Due to our novel architecture for object recognition the robot is capable to detect and to recognize yet unseen objects described by their attributes. All this makes for a solid base system.
The focus of this thesis, however, is on the high-level control of the robot. More precisely, we aim to integrate qualitative reasoning and human-robot interaction to enable a successful application of our service robot for complex tasks in human environments. In the next chapter we present the foundations for later chapters of this thesis. Some of the readers may prefer to skip on to the subsequent chapters that cover our main contributions and only consult Chapter 3 if necessary. The integration of new modules for human-robot interaction into the robot systems and their application with the high-level control in READYLOG is shown in Chapter 4. To bridge the gap between the different representations used by humans and by robots, especially for positional information, we present an extension to our high-level control in Chapter 5. Also, we propose another extension to increase the robustness of our robot against internal faults and we apply READYLOG for flexible command interpretation in Chapter 6.

3 Foundations

In this chapter we lay out the foundations that the remaining chapters of this thesis build upon. Some readers may prefer to skip on to the next chapter and come back to the foundations later if needed.

First, we give the mathematical foundations that underlie the low- and mid-level components of our domestic service robot in Section 3.1. That is, in particular, the concept of random forests, a form of decision trees with randomized selection criteria. We use random forests for both face and gesture recognition later. We also introduce Markov Decision Processes which are used for decision-theoretic planning in our high-level control. For our qualitative spatial representations we make use of fuzzy logic. Hence, we introduce fuzzy sets and fuzzy controllers in Section 3.2. Also, we build on existing work on spatial representations and reasoning. That is why we present the corresponding method in Section 3.3, respectively. Lastly, the high-level control of our service robot builds upon a powerful logic-based framework that we introduce in Section 3.4.

3.1 Mathematical Foundations

In Chapter 4 we extend our robot's mid-level system with means to detect and recognize faces and to detect hands and recognize gestures. Detection and recognition are *classification* tasks, binary for detection and multi-class for recognition. We give some of the mathematical foundations used in our approaches now.

3.1.1 Decision Trees

A popular method for reactive decision making are decision trees (Russell and Norvig, 2010). A *decision tree* is a function that maps a vector of attributes to a single output value, the decision. The decision is reached by a sequence of tests that are performed while descending a tree-like structure of tests. At each node, one of the attributes is tested and the node branches into child nodes for the given values of the attribute.

Decision trees are often created by *supervised learning*. Given a set of example input-output pairs *learning* is done using induction of decision trees (Quinlan, 1986). C4.5 (Quinlan, 1993) as well as its predecessor ID3 select the next attribute to test by looking at the *information gain* measured in terms of *entropy* (Shannon and Weaver, 1949) of the split at that node. The entropy of a discrete random variable X with n possible values $x_1, \ldots, x_n$ is given as

$$H(X) = -\sum_{k=1}^{n} p(x_k) \cdot \log_2 p(x_k)$$

where $p(x_k)$ is the probability of X taking on value x_k. The information gain of a split with a certain attribute then is calculated as the expected reduction in entropy. For a Boolean random variable which is *true* with probability q we can define the entropy of the variable as

$$B(q) = -(q \log_2 q + (1-q)\log_2(1-q)).$$

Now consider an attribute A that has d distinct values $a_1, \ldots, a_d$. It divides the training set T into subsets $T_1, \ldots, T_d$ where each subset T_k has p_k positive and n_k negative examples. Going along the kth branch requires $B(p_k/(p_k + n_k))$ bits of information to answer the overall decision question. If we choose an example from the training set at random, it has the kth value for the attribute A with probability $(p_k + n_k)/(p + n)$. The expected entropy that remains after testing A can be computed as

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} \cdot B(\frac{p_k}{p_k + n_k}).$$

The information gain as the expected reduction in entropy is

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A).$$

A serious limitation of decision trees is that of *overfitting* (Russell and Norvig, 2010). Overfitting refers to instances of decision trees that are able to predict the training set but that did not actually learn any pattern in the data and hence fail to classify unseen data. *Pruning* is a common technique against overfitting. It reduces redundancy in decision trees by removing nodes with little or no information gain. Besides increasing the generalization of a tree by reducing its size, pruning also often reduces the classification accuracy of a tree.

3.1.2 Random Forests

With classical approaches to decision trees it is not possible to increase generalization and accuracy at the same time. Another way to counteract overfitting is to grow multiple trees with randomly selected sub-spaces of the original feature space (Ho, 1995), so-called *random trees*.
Each of the trees is trained on the complete training set but in each tree a random subset of the feature vector is chosen. For an n dimensional feature vector there are 2^n possible sub-spaces to sample. A collection of such trees then yields a *random decision forest* (Ho, 1998). Since the trees in the forest are trained independently, the generalization capabilities can be raised without sacrificing the accuracy. With an increasing number of trees in the forest very high accuracy results can be achieved (Ho, 1998). If the feature vector does not consist of Boolean variable only, also the decision criterion can be randomized. For example, for a discrete numerical variable with values between 0 and 10 the split value can be sampled as well.

A thorough mathematical treatment of random forests can be found in (Breiman, 2001). For an ensemble of classifiers, Breiman introduces so-called *margin functions* to model the generalization error of a classifier. This function tells us by how much the classifier returns the precise class compared to the most common wrong class, on average. The generalization error of the ensemble then is bounded by the mean correlation between the single classifiers in an ensemble and their so-called *strength*, which is a function highly dependent on the individual generalization error of a classifier. The generalization error of the ensemble is directly proportional to the amount of randomization and the independence of the individual classifiers in it. Breiman derives that the generalization error converges to a limit as the number of trees tends to infinity.
In summary, random forests yield the following advantages (Breiman, 2001):

- The independent training of trees naturally allows for parallelization.
- The training is fast because potentially costly search for the most discriminant feature is replace with random selection.
- The accuracy is favourable to approaches such as bagging or boosting (Quinlan, 1996).
- The generalization error tends to a limit as the number of trees grows.

We employ random forests in two of our approaches, namely for face recognition and for gesture recognition. Both are image processing tasks that use a feature-based method for recognizing classes of objects, that is faces of different persons and different poses of a hand, respectively. The features that we use for the classification are local features in the image where the position and the size of the feature can be varied. Object recognition with local features has several advantages itself (Matas and Obdržálek, 2004) which come even more into effect when combined with random forests. These are as follows:

- Learning is straightforward because the random sampling does not imply any explicit model of the objects to recognize. Instead, the model can automatically be created in the learning process.
- There is a better classification with partial occlusions because not all classifiers need to be accurate in a forest. That is, if some of the trees were trained on parts of the object that are occluded in the instance to recognize, the overall voting in the forest can make up for that.
- Variations are better accounted for due to the randomization that is straightforward for our local features.

Hence, using random forests with local features for image processing, in general, and for object recognition, in particular, is a particularly promising combination.

3.1.3 Markov Decision Processes

Markov Decision Processes (MDPs) are a model to describe dynamic stochastic systems, a thorough discussion can be found in (Puterman, 1994). We present a sub-class of MDPs here, namely that of *fully observable* MDPs in stochastic environments. A system modelled by an MDP is in exactly one of a set of states which are used to represent the environment. An agent in such a system is able to take decisions at certain stages. In our limited treatment here we consider decisions being possible at discrete so-called decision epochs only. With each decision, the agent takes an action that results in a state transition, possibly ending the same state as before. A *transition model* describes the

outcome of each action in every state. The actions may be stochastic, that is to say, there are multiple outcomes possible leading to different states. The probability $P(s'|s,a)$ of reaching state s' with action a from s is specified in the transition model. MDPs make use of the Markov assumption: the probability of reaching state s' from state s with an action a only depends on s and a and not on any earlier state. In addition to the above, there is a *utility function* that that rates a sequence of state-action transitions. This is done using a *reward function* $R(s)$ that assesses the desirability of a state s. The utility then is computed by the rewards along the history of states.
Formally, an MDP is defined as a tuple $M =< S, A, T, R >$ with S being a set of states, A a finite set of actions, T a transition model, and R a real-valued reward function. The transition model with stochastic actions as described above is denoted with a *transition function* $T : S \times A \times S \rightarrow [0, 1]$ where $T(s_i, a, s_j)$ is the probability of reaching state s_j from state s_i by taking action a. If the number of stages at which to take decisions is finite, the MDP is called having a finite horizon, if the number is infinite it is called an infinite horizon MDP. If the number of stages needed to reach a definite terminal stage is not known in advance, the MDP has indefinite horizon.
The general goal with MDPs is to create a so-called *policy* π, which is a conditional plan, that maximizes the expected benefit with the environment. The policy recommends an action $\pi(s)$ for the agent to take for every state s. An *optimal policy* is a policy that maximizes the expected utility of possible histories in the environment and is usually denoted π^*. If the advice for what to do in a certain state is independent of the current stage, the policy is called *stationary*, otherwise it is called *non-stationary*. For stationary policies we can represent the transition model as one fixed matrix while a non-stationary policy has one such matrix for every epoch.
The *value* of a policy in an MDP is the (discounted) sum of the expected rewards when following the policy π:

$$V(s) = E_\pi \left[\sum_{t=0}^{T} R(s_t | s_0 = s) \right], \qquad V(s) = E_\pi \left[\gamma \cdot \sum_{t=0}^{\infty} R(s_t | s_0 = s) \right]$$

where t is the epoch and γ is a discount factor with $0 \leq \gamma < 1$. The discount factor is used to prevent the value from growing unboundedly. The above equation for the discounted value can be written as

$$V^\pi(s_i) = R(s_i) + \gamma \sum_{s_j \in S} T(s_i, \pi(s_i), s_j) \cdot V^\pi(s_j)$$

which is known as the Bellman equation (Bellman, 1957).
There are two classical methods to derive an optimal policy for an MDP if the model is known: *policy iteration* and *value iteration*.
Policy iteration (Howard, 1960) directly modifies the policy. It starts with an arbitrary policy π_0 and then iterates two steps: *policy evaluation* and *policy improvement*. The former calculates a value $V^{\pi_i}(s)$ for every state s. The latter calculates the expected value at every state for every action and replace the current action in the policy with an action yielding a better value if such an action is found. This process is iterated until it converges to the optimal policy.
*Value iteration*value iteration uses dynamic programming to solve a definite equation system. From an optimality principle for finite horizon MDP problems (Bellman, 1957)

by generalization to infinite horizon Howard (1960) showed that there always is a stationary policy for such problems whose optimal value function satisfies

$$V^*(s_i) = R(s_i) + \max_{a \in A} \gamma \sum_{s_j \in S} T(s_i, a, s_j) \cdot V^*(s_j).$$

With this as a basis a system of equations is derived and solved iteratively. Taking the optimal action in each state then results in the optimal policy.
In Section 3.4.2.2 we present a third method to find an optimal policy for finite horizon MDPs. This is used for decision-theoretic planning in our high-level agent controller. For a thorough treatment and discussion of MDPs and decision-theoretic planning we refer to (Puterman, 1994; Boutilier et al., 1999).

3.2 Fuzzy Logic and Fuzzy Control

In Chapter 5 we use *fuzzy logic* to bridge the gap between human and robot representations and to allow for a more natural specification of robot controllers in domestic domains. We do so by establishing a semantics for qualitative fluents based on *fuzzy sets* (Zadeh, 1965) in the situation calculus. We later also integrate *fuzzy controllers* into our GOLOG dialect.

3.2.1 Fuzzy Logic

Fuzzy logic is form of multi-valued logic for approximate reasoning. It is based on the theory of fuzzy sets which describes classes of objects in terms of the degree of membership of an object to a class. This allows for unsharp boundaries of the classes.
In an earlier work, Sugeno and Yasukawa (1993) discussed the adequacy of fuzzy logic-based approaches for qualitative modelling. In their work they describe the process of generating a qualitative model based on fuzzy representations from a set of sample input-output data describing the system behaviour. Based on heuristics they identify the input data which influence the output. Then, they approximate the number of fuzzy rules needed to describe the output data. The result is a fuzzy controller, i.e. a set of fuzzy rules, which describes the mapping from the given input values to the output values. Later, Tikk et al. (2002) improved the original idea and proposed several algorithms for building trapezoid approximations of membership functions and for rule base reduction. Here, fuzzy logic is used for approximating a given system in a qualitative way.
In (Bolloju, 1996), Bolloju uses qualitative variables based on fuzzy sets to facilitate decision making. While the approach of Bolloju is similar to ours, his approach is very limited in the way decisions can be made. Similar lines as to use fuzzy qualitative variables are followed in (Nordvik et al., 1988; Dutta, 1990). In our approach however, we can make use of the full expressiveness of the situation calculus for taking decisions.
A closely related approach is (Yen and Lee, 1993). They propose a fuzzy test-score semantics for soft constraints. They propose a function fholds which evaluates the result of combinations of soft constraints in allusion to the holds predicate known from action logics. Although they state to use the situation calculus, besides the function fholds they do not give a full axiomatization.

3.2.1.1 Fuzzy Set Theory

The underlying concept in fuzzy logic is that of *linguistic variables* as introduced by Zadeh in (Zadeh, 1975). These linguistic terms can be used for representation as well as for control. According to Dubois and Prade (1998), "fuzzy sets and possibility theory offer a unified framework for taking into account the gradual or flexible nature of many predicates, requirements, and the representation of incomplete information". It is suited to represent human-like taxonomies in pattern classification, or lexical imprecision of natural language. To better understand the concept of fuzzy sets we first give a definition of crisp sets.

3.2.1.2 Crisp Sets

A *crisp set* A over a universe of discourse U can be defined by $A = \{x|x$ meets some condition$\}$ and be stated with a subset relation if any $x \in A \subset U$; or alternatively by defining a characteristic function μ_A as $\mu_A = 1$, if $x \in A$, and $\mu_A = 0$, otherwise. For two sets $A, B \subset U$, the union $A \cup B$ is defined as $\mu_{A\cup B}(x) = 1$ if $x \in A$ or $x \in B$ and $\mu_{A\cup B}(x) = 0$ if $x \notin A$ and $x \notin B$. For the intersection $A \cap B$ it holds: $\mu_{A\cap B} = 1$ if $x \in A$ and $x \in B$, and $\mu_{A\cap B} = 0$ if $x \notin A$ or $x \notin B$. The complement $\bar{A}$ is defined such that $\mu_{\bar{A}}(x) = 1$ if $x \notin A$; $\mu_{\bar{A}}(x) = 0$ if $x \in A$. Following (Mendel, 1995), it can be shown that

$$(A \cup B) \supset \mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)] \tag{3.1}$$

$$(A \cap B) \supset \mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)] \tag{3.2}$$

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{3.3}$$

Crisp set operations enjoy the property of being commutative, associative, and distributive. Further, De Morgan's laws as well as the Law of Contradiction and Excluded Middle hold.

3.2.1.3 Membership Functions and Fuzzy Sets

A fuzzy set F with a universe of discourse U, on the other hand, can only be characterized with the help of a *membership function* $\mu_F : U \to [0, 1]$. The membership function provides a measure of the degree of similarity of an element in U to the fuzzy set. A fuzzy set F in U can be represented as a set of ordered pairs of a generic element x and its grade of membership: $F = \{(x, \mu_F(x))|x \in U\}$. Union, intersection, and complement can be defined the same way as for crisp sets (Equations 3.1–3.3). Note that unlike for crisp sets, the Law of Contradiction and the Excluded Middle do *not* hold, that is $A \cup \bar{A} \neq U$ and $A \cap \bar{A} \neq \emptyset$. In the definitions of intersection and union we used the "max" and "min" operators. But these are not the only possibilities to define fuzzy union and fuzzy intersection. In general, for set intersection several different so-called t-norms have been proposed, usually written as $\mu_{R\cap S}(x) = \mu_R(x) \star \mu_S(x)$, for set union t-conorms or s-norms, written as $\mu_{R\cup S}(x) = \mu_R(x) \oplus \mu_S(x)$ were formulated (see (Dubois and Prade, 1998) for examples). Here, we rely on the min t-norm (Equation 3.1), the max t-conorm (Equation 3.2) and Equation 3.3 for the complement.
To be able to conclude something useful with fuzzy variables, we need inference rules which define how to reason with variables of this kind. In the following, we give several examples of the type of inference possible by stating examples from (Zadeh, 1989):

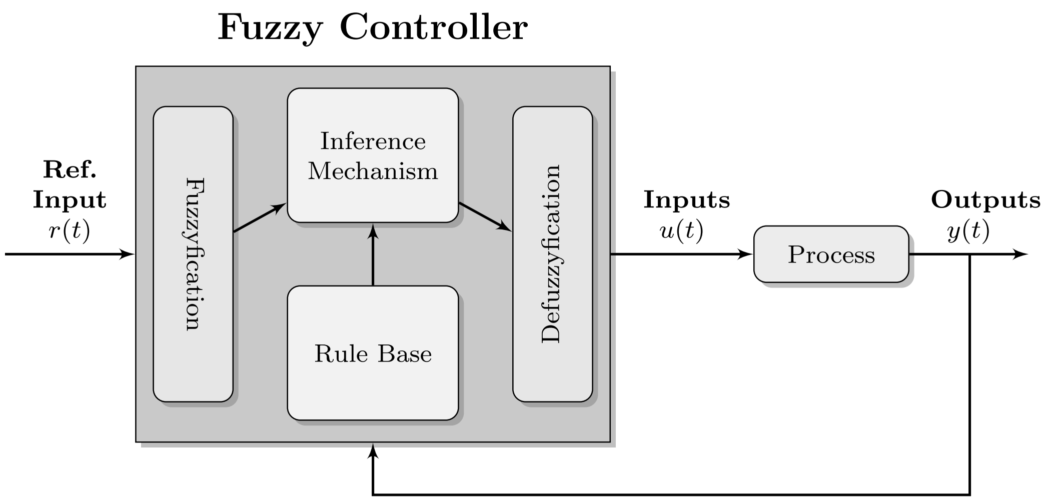


Figure 3.1: General architecture of a fuzzy controller from (Passino and Yurkovich, 1998)

(1) *Categorical rules* like *X is small*; (2) *Entailment rules* like *Mary is very young* and *very young implies young* implies *Mary is young*; (3) *Conjunction/Disjunction rules* like *pressure is not very high* and/or *pressure is not very low* implies *pressure is not very high and/or not very low*; (4) *Compositional rules* like X *is much larger than* Y and Y *is large* implies X *is much larger* $\circ$ *large* (where $\circ$ denotes the composition operator); (5) *Negation rules* like *not(Mary is young)* implies *Mary is not young*; (6) *Extension principle* like X *is small* implies X^2 *is* ^{2}*small* with ^{2}small meaning *very small*. Another way to see these rules is by interpreting X and Y as decision variables and A as a soft constraint, allowing for a degree of membership. If there are only two membership values (true or false), then these constraints can be regarded as hard constraints (see e.g. (Dubois and Prade, 1998)).

3.2.2 Fuzzy Control

The fuzzy membership function can be seen as a degree of similarity, degree of preference, or degree of plausibility. From the first interpretation the rough set theory evolved, while most engineering applications make use of fuzzy rules of the form *if* X_1 *is* A_1 *and* $\ldots$ X_n *is* A_n *then* Y *is* B. These rules are often used in fuzzy controllers (Passino and Yurkovich, 1998). The common application of a fuzzy controller is that some process should be controlled by a set of if-then rules. The process has some output and further takes a certain control input. The general architecture of a fuzzy controller is shown in Figure 3.1. The quantitative sensor values $y(t)$, together with some reference input $r(t)$, which describes the vital state of the system, need to be *fuzzified*, i.e., the membership to a certain linguistic class needs to be determined. The *inference mechanism* uses these fuzzified input values together with a *rule base* of fuzzy rules to select the appropriate control output. The output as such uses fuzzy categories and thus must be *defuzzified* to serve as an input $u(t)$ for the real world. The output of the real world process serves as the sensor input for the next control step.
So we basically have four components that make up a fuzzy controller: (1) a rule base, (2) fuzzyfication, (3) an inference mechanism, and (4) defuzzyfication. With this, fuzzy

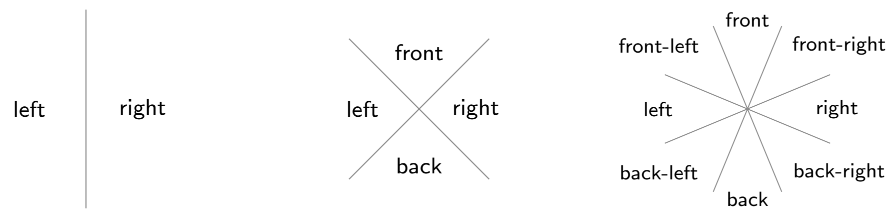
controllers allow for representing the heuristic knowledge that is commonly available to humans on how to control a system. We make this human-like form of control available in our high-level language for tasks where such reactive behaviour is appropriate.

3.3 Qualitative Positional Information

In our effort to bridge the gap in representing positional information with humans and robots we strongly build upon an existing approach to qualitative spatial representations and reasoning which we detail in the following.

3.3.1 Overview

First, Hernandez (1991) introduces an approach to *qualitative orientation relation*. Instead of using geometric models which are very precise, Hernandez establishes a cognitive model of space. He argues that cognitive spatial concepts are qualitative in nature and preciseness is normally not needed in cognitive models. Although the representation might correspond to many '*real*' situations, it avoids falsifying effects of an exact geometric approach which are likely due to the common limited acuity of perception. Hernandez states that the direct modelling of qualitative statements allows for a more efficient way to handle partial and uncertain spatial information. The orientation relation is meant to represent where objects are placed relative to each other. The framework proposes to model qualitative orientation with angular intervals. The number of intervals, that is, the number of distinctions is determined by a *level of granularity*. An orientation relation states where a *primary object* is located in relation to a *reference object*. Further, there is a *reference frame* which determines the direction in which the primary object is located in relation to the reference object. Hernandez presents methods for changing the reference frame as well as methods for composing relations.
Then, a basic method for *qualitative distances* was discussed by Hernandez et al. (1995). Three elements are needed to establish a distance relation, namely a *primary object*, a *reference object*, and a *frame of reference*. The distance $\mathrm{dist}(A, B)$ between the reference object A and the primary object B is expressed as one out of a set of distance relations. These relations are formed by partitioning the plane into regions. These partitions represent the distinctions being made. The context in which the distinctions are made is represented by the frame of reference. It accounts for contextual information such as type and scale of the distance relations as well as for the *distance system*, which contains the distance relations and a set of structure relations describing how the distance relations relate to each other. Among their representation of distances through geometric intervals, they describe basic inference mechanisms such as composition and switching between different frames of reference.
Finally, Clementini et al. (1997) present a unified framework for qualitative representation of positional information in two-dimensional space. In order to represent positional information they combine the orientation and distance relations from above. The position of a *primary object* then is represented by a pair of distance and orientation relations with respect to a *reference object*. Both relations depend on a so-called *frame of reference* which accounts for several factors such as the size of objects, different points of view,



(a) level 1 (b) level 2 (c) level 3

Figure 3.2: Different levels of granularity for the orientation relation

and so on. The framework features basic reasoning capabilities such as the composition of spatial relations as well as switching between different frames of reference.
Since our representation of positional information is largely based on the approach presented in (Clementini et al., 1997) we detail it now.

3.3.2 Orientation Relation

Orientation relations are used to describe where objects are placed relatively to each other. Based on the fundamental observation of how three points in the plane relate to each other an orientation relation can be defined in terms of three basic concepts: the *primary object*, the *reference object*, and the *frame of reference* which contains the *point of view*.
The point of view and the reference object are connected by a straight line. The view direction is then determined by a vector from the point of view to the reference object. The location of a primary object is expressed with regard to the view direction as one of a set of relations. The number of distinctions made is determined by the *level of granularity*. We are going to detail this in the following.

Level of granularity

There are different levels of granularity for orientation relations. On the first level the point of view and the reference object are connected by a straight line such that the primary object can be to the left, to the right, or on that line. Thus the first level partitions the plane into two half-planes. On the second level there would be four partitions, the third level would have eight, and so on. Figure 3.2 depicts the first three levels of granularity for the orientation relation.
A well known example of qualitative directions are the points of a compass which, in the simple case, correspond to the second level: a place can be *north*, *east*, *south*, or *west*. In some cases, we notice even finer distinctions in terms of intermediate directions such as *north-east* or *south-west*. This setting of finer distinctions corresponds to the third level in the above model.

Frames of reference

We already mentioned the term *frame of reference*. It accounts for contextual aspects such as the *front* side of the reference object. Different frames of reference can be

classified into three basic types: they are called *intrinsic* if the orientation is given by some inherent property of the reference object, *extrinsic* if a particular orientation is imposed by external factors such as motion, or *deictic* if the orientation is given by the point of view from which the reference object is seen. Based on the frame of reference there is a 'front' side of the reference object. Independent from the level of granularity there is a uniform circular neighbouring structure. In general, at a level of granularity k the set $\{\alpha_0, \alpha_1, \dots, \alpha_n\}$ denotes the $n+1$ orientation relations where $n = 2^k - 1$. Having a reference object A and a primary object B the orientation of B with respect to A is denoted by $\theta_{AB} = \theta(A, B)$. It can take any of the values mentioned in the set above.
Each orientation has a *successor* such that $succ(\alpha_0) = \alpha_1$, $succ(\alpha_1) = \alpha_2$, …, $succ(\alpha_n) = \alpha_0$ and a *predecessor* with $pred(\alpha_0) = \alpha_n$, $pred(\alpha_1) = \alpha_0$, …, $pred(\alpha_n) = \alpha_{n-1}$. In addition each orientation α_i has an *opposite* orientation which is obtained by applying $(n+1)/2$ times the function $succ$ to α_i. The minimal number of steps needed to get from orientation α_i to α_j along the circular neigh-boring structure is called *range*. The range between opposite orientations is $(n+1)/2$. Two orientations are called *orthogonal* if the range between them is $(n+1)/4$ (except for the basic level 1). Each orientation has two orthogonal orientations to it.

3.3.3 Distance Relation

Analogous to the orientation relation establishing a *distance relation* requires three elements: the primary object, the reference object, and the frame of reference.
First of all, we like to describe commonly used distance systems. In a metric space, the distance between points is defined by the following three axioms:

$$dist(P_1, P_1) = 0 \qquad \text{(Reflexivity)}$$
$$dist(P_1, P_2) = dist(P_2, P_1) \qquad \text{(Symmetry)}$$
$$dist(P_1, P_2) + dist(P_2, P_3) \leq dist(P_1, P_3) \qquad \text{(Triangle Inequality)}$$

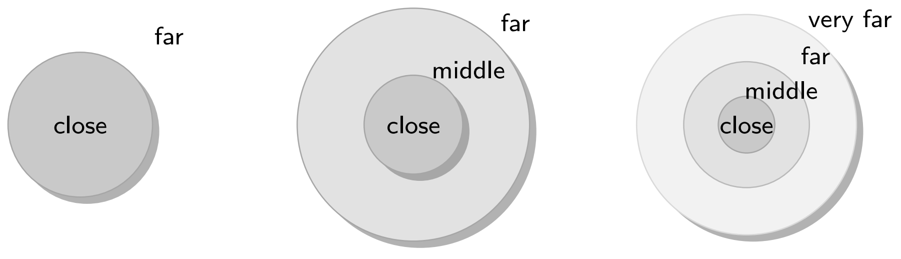
Conventional concepts of distance normally rely on coordinates. The distance between points $P_i = (x_{i1}, x_{i2}, \dots, x_{in})$ in a n-dimensional vector space can be expressed in terms of the Minkowsky L_p-metric (Preparata and Shamos, 1985):

$$d_p(P_1, P_2) = \Big(\sum_{j=1}^{n} |x_{1j} - x_{2j}|^p\Big)^{1/p}$$

Well known examples are, for instance, the *Manhattan distance* which is defined by the L_1-metric or the *Euclidean distance* which is defined by the L_2-metric.
If we want to use a qualitative concept of distance instead, we usually do not have any coordinate system. The distance of two points expressed in a qualitative way often depends not only on their absolute positions but also on cultural and experimental factors and on the frame of reference.

Level of granularity

Similar to the orientation relation we can distinguish distances at various levels of granularity. An arbitrary level n of granularity with $n+1$ distinctions yields to the set $Q = \{q_0, q_1, \dots, q_n\}$ of qualitative distances. Given a reference object RO these



(a) level 1 (b) level 2 (c) level 3

Figure 3.3: Different levels of granularity for the distance relation

distances partition the space around RO such that q_0 is the distance closest to RO and q_n the one farthest away.
The qualitative distance between the reference object RO and a primary object PO both belonging to a set O of objects is a function $d : O \times O \to Q$. Let us consider A as the reference object and B as the primary object then $d_{AB} = d(A, B)$ denotes the distance from A to B. There is a total order of magnitude for the elements of Q saying $(q_0 < q_1 < \ldots < q_n)$ which allows us to define a *successor* being the next symbol in the above list, that is $succ(q_i) = q_{i+1}$ for each $i < n$ and $succ(q_n) = q_n$. Accordingly, there is a *predecessor* giving the previous symbol of the list, that is $pred(q_i) = q_{i-1}$ for each $i > 0$ and $pred(q_0) = q_0$. Furthermore we can introduce a function *ordinal* defined as follows: $ord : Q \to \{1 \ldots n+1\}$, such that $ord(q_i) = i + 1$, and $ord^{-1}(i) = q_{i-1}$, except $ord^{-1} = q_n$ for $i > n$, $ord^{-1}(i) = q_0$ for $i \leq 1$.

Distance system

In order to be able to compare the magnitude of distances, i.e. how they relate to each other, we need to specify several structure relations. They are organized in a so-called *distance system* D defined as:

$$D = (Q, \mathcal{A}, \mathcal{I})$$

where

- Q is the totally ordered set of distance relations;
- $\mathcal{A}$ is an *acceptance function* defined as $\mathcal{A}$: $Q \times O \to I$, such that, given a reference object RO, $\mathcal{A}(q_i, RO)$ returns the geometric interval $\delta_i \in I$ corresponding to the distance relation q_i;
- $\mathcal{I}$ is an algebraic structure with operations and order relations defined over a set of intervals I. $\mathcal{I}$ defines the *structure relations* between intervals.

Each distance relation can be associated to an *acceptance area*. These areas can be uniquely identified with a series of consecutive intervals $\delta_0, \delta_1, \ldots, \delta_n$ which are called *distance ranges*.
There are various interpretations of the intervals $\delta_0, \delta_1, \ldots, \delta_n$. In the case of a strict interpretation we have sharp boundaries and therefore there are points $a_1, a_2, \ldots, a_n \in \mathcal{R}^+$ that make up the intervals $[0, a_1], [a_1, a_2], \ldots, [a_{2n}, +\infty]$. However, cognitive considerations suggest that this is not always the case since intervals may have indeterminate boundaries. The intervals can, then, either be *overlapping* or *non-exhaustive*.

For the overlapping case two consecutive intervals share a common region. This can be represented by as $[0, a_1], [a_2, a_3], \ldots, [a_{2n}, \infty]$ with $a_{2i} \leq a_{2i-1}$ and $a_{2i} > a_{2i-2}$. For the non-exhaustive case, where there is a void space between the acceptance areas of two consecutive distance relations, the intervals would be represented by $[a_0, a_1], [a_2, a_3], \ldots, [a_{2n}, +\infty]$ with $a_0, \ldots, a_{2n} \in \mathcal{R}^+$ and $a_{2i} > a_{2i-1}$ for $i \geq 1$. Any of the above interpretations is possible with the model presented so far as it only uses the notion of consecutive intervals inherited from the total order of $\mathcal{Q}$.
In the following, we briefly sketch an exemplary algebraic structure $\mathcal{I}$. For a detailed discussion of algebraic structures $\mathcal{I}$ we refer to Clementini et al. (1997). Consider $\mathcal{I} = (\mathcal{I}, +, \leq, \ll)$ which defines a sum operation $(+)$ and two order relations $(\leq, \ll)$ over the set I of closed intervals over $\mathcal{R}^+$.
Given two intervals the binary operator $+$ returns the minimal interval that contains both given intervals. Given two intervals $[a, b]$ and $[c, d]$ the operation $+ : I \times I \rightarrow I$ is defined by:

$$[a, b] + [c, d] = [\min(a, c), \max(b, d)]$$

For comparing intervals we have two order relations. Let $\|i\|$ be the function returning the length of an interval i. Then the order relation $\leq$ is defined as follows:

$$i_1 \leq i_2 \Leftrightarrow \|i_1\| \leq \|i_2\|, \forall i_1, i_2 \in I$$

If two intervals have the same length, we say they are *congruent* $(\cong)$:

$$i_1 \cong i_2 \Leftrightarrow \|i_1\| = \|i_2\|, \forall i_1, i_2 \in I$$

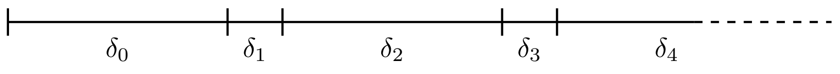
Let us consider an indistinguishability relation $(\approx)$ between the length of intervals which states that no relevant predicate distinguishes between them. We indicate it with $\|i_1\| \approx \|i_2\|$. It allows us to define an order relation *much less than* between lengths:

$$\|i_1\| + \|i_2\| \approx \|i_2\| \Leftarrow \|i_1\| \ll \|i_2\|$$

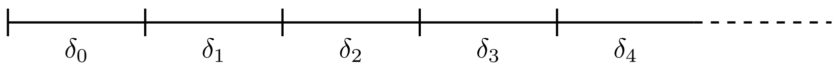
Then, the order relation *much less than* $(\ll)$ between intervals can be defined as follows:

$$i_1 \ll i_2 \Leftrightarrow \|i_1\| \ll \|i_2\|, \forall i_1, i_2 \in I$$

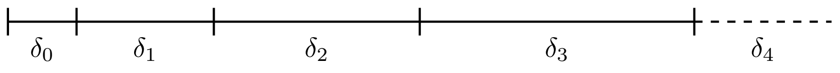
We can also say that the bigger interval *absorbs* the smaller one.
We are now able to define *structure relations* over the domain of intervals. By applying the sum operation to all possible combinations of intervals we obtain a set of intervals Δ which is a subset of I. $\Delta_{h..i}$ is the sum of consecutive intervals δ_k from δ_h to δ_i. If the sum starts from the origin, that is $h = 0$, we abbreviate this to Δ_i which is the distance range from the origin up to and including δ_i. The structure relations of the distance system are the order relations holding between all intervals in Δ.
If the structure relations of the distance domain follow a recurrent pattern they are called *homogeneous*. Examples for distance systems with different homogeneous structure relations are depicted in Figure 3.4.
The structure relations are arbitrary, in general. Yet, we present a restricted set of properties that seems appropriate for our context of domestic domains. By applying several restrictions to the structure of intervals we can achieve fewer indeterminacy in the composition of relations. The constrained properties are motivated by cognitive considerations.



(a) alternating bigger and smaller ranges



(b) equally spaced ranges



(c) increasingly bigger ranges

Figure 3.4: Different distance ranges with homogeneous properties

It is a common phenomenon to make finer distinction in the neighbourhood of the reference object than in the periphery. Therefore, a *monotonicity* restriction is established. It constraints the distance domain to increasingly bigger ranges so that any given interval is bigger or equal than the previous one:

$$\delta_0 \leq \delta_1 \leq \delta_0 \leq \ldots \leq \delta_n$$

It is also possible to exclude equally spaced intervals and demand more distant relations to correspond to considerably bigger ranges. This can be achieved by a *range restriction* which states that any given interval is bigger than the entire range from the origin to the previous interval (denoted by Δ):

$$\delta_i \geq \Delta_{i-1} \forall i > 0$$

This can be further emphasized to obtain differences in the *order of magnitude*. It allows us to disregard smaller intervals δ_i with respect to much bigger intervals δ_j for $i < j$. For a given difference p between the orders of two distance relations q_i and q_j, with $i \leq p \leq (j - i)$, the following holds:

$$(\mathrm{ord}(q_j) - \mathrm{ord}(q_i)) \geq p \Rightarrow \delta_j \gg \delta_i, \forall i, j \geq 0, i < j$$

Thus, a given interval absorbs all the smaller intervals including its immediate predecessor.

Frames of reference

As in the description of qualitative orientation where the frame of reference was meant to fix the 'front' side of the reference object we need to attend frames of reference for the distance relation, too. According to Clementini et al. (1997) we make use of a more general concept of a frame of reference defined as follows. The distance frame of reference is made up of three components:

$$\mathsf{FofR} = (D, S, T)$$

where

- D is a *distance system*:
  We already talked about distance systems in the previous section. The distance system contains contextual information which determine relevant distinctions and their structure.
- S is a *scale*:
  Scale can be understood not only as the proportional relationship of representation to the things it represents but also as proportion between different spatial extensions. The former is the case, for example, with legends like they are used in maps.
- T is a type:
  Analogously to orientation the type can either be
  - *intrinsic*:
    The distance is determined by an inherent characteristics of the reference object. A characteristics can be, for example, its size, shape, or topology.
  - *extrinsic*:
    The distance is determined by some external factor. This could be, for example, the arrangement of objects, the travelling time, or the costs involved.
  - *deictic*:
    The distance is determined by an external point of view. Often this is the case if there is an observer perceiving objects. This is not always meant in the sense of sight but can also be the case for how an individual builds a mental map of the space regarded.

Although each component, in principle, can be combined with any of the other ones all three components are somehow related to one another. Often the type of FofR influences the scale which itself modifies the distance system.
For any domain that we want to apply our qualitative distance to, we have to fix any of the open choices mentioned so far. With our application domain in this thesis being domestic service robotics we describe our choices in the following.
For one, we have to choose an appropriate distance system for our domain. Please, recall that the distance system contains several structure relations in order to enable us to describe how the distance relations relate to each other. We choose a *homogeneous* partitioning, that is, the distance domain follows a recurrent pattern. Instead of using equally spaced distance intervals we think it might be more appropriate to make more distant relations correspond to a bigger range. That is not only because the perception of objects and situations farther away is less accurate but also because of a temporal aspect. Since the time needed to get to a place far away is longer than the time needed to get to place close to one's current position there is more time for properties in the environment to change.
For another, we also need to make some remarks about scale. In interactions between a domestic service robot and a human user the scale is likely to be dependent on the context. Distances given with respect to a particular room certainly have a far bigger scale than distances on a table. Also, the connection between the scale and the type becomes apparent. Depending on the influencing factor, the distance and in turn the scale is determined. A common measure that we want to exploit here, is that the context usually entails a reasonable maximal distance. We use this maximal distance to infer the categories of our distance relation.

Following our above considerations let us examine a way to represent distance in some more detail. Motivated by the cognitively plausible fact that it is common to make finer distinctions in the neighbourhood we want to achieve monotonicity. Therefore, we could determine our distance intervals as follows. Let base be a scaling factor. Each distance interval is then greater by factor base than its direct predecessor. As we already stated we assume there is a maximal distance $\mathsf{dist}_{\mathsf{max}}$ due to some property of the context like the size of a room. We further parameterize our distance system with the *level of granularity* n. At level n there are $n+1$ distance relations. You may notice that we include an additional interval representing the distance relation out_of_reach which corresponds to all distances greater than the maximal distance $\mathsf{dist}_{\mathsf{max}}$.
The maximal distance can then be represented by

$$\mathsf{dist}_{\mathsf{max}} = \|\delta_0\| + \|\delta_1\| + \ldots + \|\delta_{n-1}\|$$

Now, we have to subdivide the maximal distance into n intervals δ_i satisfying a monotonicity restriction. We formulated our monotonicity restriction by requiring that each interval is bigger than the preceding interval by factor base. Starting with the first interval δ_0 the length of each subsequent interval δ_i is acquired by multiplying the length of the preceding interval δ_{i-1} with factor base. This can also be expressed as

$$\|\delta_i\| = \|\delta_0\| \cdot \mathrm{base}^i$$

As a result, we can write the above formula as

$$\mathsf{dist}_{\mathsf{max}} = \|\delta_0\| \cdot \mathsf{base}^0 + \|\delta_0\| \cdot \mathsf{base}^1 + \ldots + \|\delta_0\| \cdot \mathsf{base}^{n-1}$$

Unfortunately, we do not know the length of any of the intervals yet. Therefore, we compute the *sum of powers*, that is the sum of the base to the power of the distance relation's index over all distance relations (excluding the additional relation).

$$\mathsf{SumOfPowers} = \sum_{i=0}^{i<n-1} \mathsf{base}^i$$

With this value we can now obtain the length of the first distance interval by

$$\|\delta_0\| = \frac{\mathsf{dist}_{\mathsf{max}}}{\mathsf{SumOfPow}}$$

We can then compute the length of all subsequent intervals as described above.
For simplicity we choose a strict interpretation of our distance relations. This leads to sharp boundaries for our distance intervals.[1] Thus, we need exactly n values to represent number of distances many intervals δ_i. These intervals are then constituted as follows:

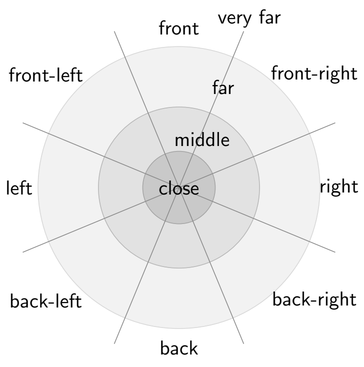
$$[0, a_1], [a_1, a_2], \ldots, [a_{n-1}, a_n], [a_n, +\infty]$$

To obtain the qualitative distance for a quantitative distance value we simply determine to which of the intervals it belongs.

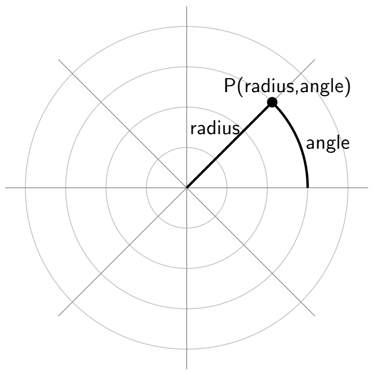
[1]We can soften the boundaries with our approach using fuzzy sets later, but defer any details to the presentation of our approach in Chapter 5.

3.3.4 Combination of Orientation and Distance Relation

In order to describe (and reason about) positional information we now investigate the combination of the distance and the orientation relation. Putting together the two relations presented above allows us to represent the location of a primary object B relatively to a reference object A.
From a quantitative point of view, the combined description of a position can be seen as the representation of a point in polar coordinates. The two dimensional polar coordinate system involves the distance from the origin and an angle. A point p in polar coordinates is defined by the distance r from the origin to the point and the angle φ measured from the horizontal x-axis to the line from the origin to p in the counterclockwise direction. Thus, the position of a point p is described as (r, φ). This description corresponds to a combination of the distance relation and the orientation relation which we presented in the previous sections. We depict an illustration in Figure 3.5.



(a) The combination of the distance and the orientation relation



(b) A point p defined in polar coordinates

Figure 3.5: The combination of distance and orientation relation compared to the polar coordinate system

Most of the time, we use the Cartesian coordinate system to represent a position. As we have just seen, the combination of our qualitative distance and orientation relation corresponds to the definition of a point in polar coordinates. Fortunately, there is an easy way to switch between the two systems. If we choose a Cartesian coordinate system with the same origin as with the polar coordinates and if we have the x-axis in direction of the polar coordinates, we have the following formulas for the transformation between the two systems

$$x = r \cdot \cos(\varphi) \qquad y = r \cdot \sin(\varphi)$$

and

$$r = \sqrt{x^2 + y^2} \qquad \varphi = \arctan \frac{y}{x} + \pi \cdot u_0(-x) \cdot \operatorname{sgn}(y)$$

where u_0 is the *Heaviside function*[2] with $u_0(0) = 0$ and sgn is the *signum function*. Here,

[2]The Heaviside function, sometimes called the unit step function, is a discontinuous function whose value is zero for negative arguments and one for positive arguments. The value of $u(0)$ can be defined freely; it is often indicated as an index to u, that is u_0 in our case.

we use the functions u_0 and sgn as logical switches instead of a distinction of different cases. Most mathematical libraries have a bivariate arcus tangent function $\operatorname{atan2}(y, x)$ that returns the correct value of φ for any x and y.
With the above formulas we are able to transform positional information given in terms of distance and orientation relations to positions in the Cartesian coordinate system. This correspondence is one of the important components of our approach to reasoning with qualitative spatial information presented in Chapter 5.

3.4 Logic-based Reasoning about Actions and Change

Among many different action formalisms, the *situation calculus* (McCarthy, 1963; Levesque et al., 1998) is a popular one; it is a powerful calculus for reasoning about actions and change, but it is also popular because the successful robot programming and planning language GOLOG (Levesque et al., 1997) is based on it. The high-level control of our domestic service robot is based on a dialect of GOLOG called READYLOG (Ferrein and Lakemeyer, 2008). We show the application of READYLOG for interactive robotics in domestic domains in Chapter 4 and we extend READYLOG with means to represent and reason with qualitative positional information in Chapter 5. Further, we integrate a form of self-maintenance in Chapter 6. That is why we now present the situation calculus, GOLOG and READYLOG in some more detail.
In the following logical formulas we use the standard logical notations along with some abbreviations. We use the logical connectives $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\supset$ (implication), $\equiv$ (equivalence), $\forall$ (universal quantification), and $\exists$ (existential quantification). When we omit brackets, the operator precedence in decreasing order is $\neg$, $\wedge$, $\vee$, $\exists$, $\forall$, $\supset$, $\equiv$. We abbreviate $\exists x_1.\exists x_2.\ldots\exists x_n.P(x_1,\ldots,x_n)$ and $\forall x_1.\forall x_2.\ldots\forall x_n.P(x_1,\ldots,x_n)$ with $\exists x_1,\ldots,x_n.P(x_1,\ldots,x_n)$ and $\forall x_1,\ldots,x_n.P(x_1,\ldots,x_n)$, respectively. Also, we use $\vec{x}$ to denote a sequence of pairwise different variables $x_1,\ldots,x_n$ and we write $\vec{x}=\vec{y}$ for $x_1 = y_1 \wedge \ldots x_n = y_n$ accordingly. All variables appearing free in sentences are implicitly universally quantified unless stated otherwise.

3.4.1 The Situation Calculus

Reiter's version of the situation calculus (McCarthy, 1963; McCarthy and Hayes, 1969; Levesque et al., 1998; Reiter, 2001) is a sorted first-order logical language[3] with equality which allows for reasoning about actions and their effects.
The language of the situation calculus $\mathcal{L}_{sitcalc}$ distinguishes three sorts: *actions*, *situations*, and domain dependent *objects*. The world evolves from an *initial situation* only due to *primitive actions*. A situation then is a world history, represented by a sequence of actions. There is a special binary function symbol $do : action \times situation \rightarrow situation$, where $do(a, s)$ denotes the situation that arises after performing action a in situation s. The initial situation is denoted by the constant S_0, that is the situation where no actions have occurred yet. We abbreviate the expression $do(a_n, \cdots do(a_1, S_0)\cdots)$ with $do([a_1,\ldots,a_n], S_0)$.

[3]with a second-order axiom

The state of the world is characterized by functions and relations which have a situation as their last argument. They are called *functional* and *relational fluents*, respectively. The elements of the sort *action* in the situation calculus are characterized by unique names. For each action one has to specify a *precondition axiom* stating under which conditions it is possible to perform the respective action and an *effect axiom* formulating how the action changes the world in terms of the specified fluents. An action precondition axiom has the form

$$Poss(a(\vec{x}), s) \equiv \Phi(\vec{x}, s)$$

where the binary predicate $Poss : action \times situation$ denotes when an action can be executed, and $\vec{x}$ stands for the arguments of action a.
Let us consider the following example of a robot navigating in a domestic environment. An important aspect of the world state is the robot's location $robotLoc(s)$. Suppose, in the initial situation S_0 the robot is in a room labelled *Kitchen*. Now, when the robot moves to a room labelled *Parlour*, its position changes to

$$robotLoc(do(goto(Parlour), S_0)) = Parlour.$$

$goto(Parlour)$ denotes the robot's action of moving from *Kitchen* to *Parlour*. The resulting situation that the world is in after performing that action is described by $s_1 = do(goto(Parlour), S_0)$. The value of the functional fluent $robotLoc(s_1)$ equals *Parlour*. The precondition axiom for our *move* action may be

$$Poss(goto(room), s) \equiv robotLoc(s) \neq room.$$

3.4.1.1 Successor State Axioms

After we have specified when it is possible to physically perform an action we still need to detail how the respective action changes the world. We do this by specifying negative and positive effects in terms of the relational fluent F as

$$\varphi_F^-(\vec{x}, a, s) \supset \neg F(\vec{x}, do(a, s)) \qquad \text{and} \qquad \varphi_F^+(\vec{x}, a, s) \supset F(\vec{x}, do(a, s)), \tag{3.4}$$

respectively. The effect axiom for a functional fluent f then is

$$\varphi_f(\vec{x}, y, a, s) \supset f(\vec{x}, do(a, s)) = y. \tag{3.5}$$

The problem with using an effect axiom for f, however, is that it describes the positive and negative effects on f but it does not describe anything about those effects that do not change the fluent. The axioms that describe the non-effects of an action are called *frame axioms*. The problem of describing these non-effects action is referred to as the *frame problem*. The number of frame axioms we would need to describe all the non-effects is very large. For relational fluents there exist $2 \cdot |\mathcal{A}| \cdot |\mathcal{F}|$ many frame axioms, when $\mathcal{A}$ is the set of actions, and $\mathcal{F}$ is the set of relational fluents. McCarthy and Hayes (1969) were the first to mention this problem.
A solution to the problem was proposed by Reiter (1991) with so-called *successor state axioms*. The trick is to collect all of the negative and positive effect axioms in one place as

$$\left(\Phi_{\neg F}^{(1)} \vee \ldots \vee \Phi_{\neg F}^{(k)}\right) \supset \neg F(\vec{x}, do(a, s)) \quad \text{and} \quad \left(\Phi_{F}^{(1)} \vee \ldots \vee \Phi_{F}^{(k)}\right) \supset F(\vec{x}, do(a, s))$$

and to put them in a normal form (Equation 3.4). Then, so-called *explanation closure* is applied as

$$F(\vec{x}, s) \land \neg F(\vec{x}, do(a, s)) \supset \varphi_F^-(\vec{x}, s) \quad \text{and} \quad \neg F(\vec{x}, s) \land F(\vec{x}, do(a, s)) \supset \varphi_F^+(\vec{x}, s)$$

with the idea behind it that, if the truth value of F changes from true to false from situation s to situation $do(a, s)$, then $\varphi_F^-(\vec{x}, a, s)$ must have been true. Similarly, if the fluent value changes from false to true, the second axiom $\varphi_F^+(\vec{x}, a, s)$ must have been true. This is where we need a unique name assumption for actions to be sure that differently named actions are in fact not the same action. Reiter shows that, under consistency assumptions for fluents together with the explanation closure axioms, the normal form axioms for the fluent F are logically equivalent to

$$F(\vec{x}, do(a, s)) \equiv \varphi_F^+(\vec{x}, a, s) \lor F(\vec{x}, a, s) \land \neg\varphi_F^-(\vec{x}, a, s). \tag{3.6}$$

The above formula is called *successor state axiom for the relational fluent* F.
The successor state axiom for the functional fluent f has the form (Reiter, 2001):

$$\begin{aligned} &f(\vec{x}, do(a, s)) = y \equiv \\ &\qquad \varphi_f(\vec{x}, y, a, s) \lor f(\vec{x}, s) = y \land \neg\exists y'.\varphi_f(\vec{x}, y', a, s) \end{aligned} \tag{3.7}$$

To ensure the consistency of the successor state axioms, the background theory must entail the properties

$$\neg\exists\vec{x}, a, s.\varphi_F^+(\vec{x}, a, s) \land \varphi_F^-(\vec{x}, a, s) \tag{3.8}$$

$$\neg\exists\vec{x}, y, y', a, s.\varphi_f(\vec{x}, y, a, s) \land \varphi_f(\vec{x}, y', a, s) \land y \neq y'. \tag{3.9}$$

The number of $|\mathcal{F}|$ successor state axioms plus $|\mathcal{A}|$ action precondition axioms plus the unique names axioms is far less than the $2 \cdot |\mathcal{F}| \cdot |\mathcal{A}|$ explicit frame axioms that would be needed otherwise.

3.4.1.2 Basic Action Theories

To do reasoning in the situation calculus as a background theory we use a so-called *basic action theory* (BAT), which is a set of sentences $\mathcal{D}$ consisting of

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0},$$

where

- Σ is the set of *foundational axioms* for situations ensuring, for instance, that no action can be performed before S_0. We refer to Pirri and Reiter (1999) and Reiter (2001) for details.
- $\mathcal{D}_{ssa}$ is a set of successor state axioms for functional and relational fluents, one for each fluent as given in Equation 3.6 for relational fluents, and in Equation 3.7 for functional fluents (together with the consistency properties in Equation 3.8 and Equation 3.9).
- $\mathcal{D}_{ap}$ is a set of action precondition axioms, one for each action. The set $\mathcal{D}_{ap}$ is the set of precondition axioms of the form $Poss(a(\vec{x}), s)$ as described above.
- $\mathcal{D}_{una}$ is the set of unique names axioms for all actions.

- $\mathcal{D}_{S_0}$ is a set of first order sentences that are uniform in S_0 and describe the fluent values in the initial situation.[4]

When we want to do reasoning using such a basic action theory we come across what is called the *projection problem*: Given a sequence of actions and a formula, determine whether this formula holds after executing the actions, or more precisely, determine whether the formula is true in the situation that results from performing the sequence of actions.

Regression

To address the projection problem a *regression* mechanism is used in the situation calculus. The idea is to successively substitute fluents in a sentence W by their successor state axiom. So if W mentions a relational fluent $F(\vec{t}, do(\alpha, \sigma))$ (with $F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s)$ being F's successor state axiom) $\mathcal{R}$ substitutes $\Phi_F(\vec{t}, \alpha, \sigma)$ for $F(\vec{t}, do(\alpha, \sigma))$. That way, the regression operator $\mathcal{R}$ determines a logically equivalent formula W' that mentions no situation term other than S_0. After regression has successively been applied until W' is uniform in S_0, the problem of proving if W holds in some complex situation reduces to a theorem proving task w.r.t. the initial database.
Without going into much detail, we state the definition from (Pirri and Reiter, 1999) for regressable formulas.

Definition 3.4.1 (Regressable Formulas) *A formula W of $\mathcal{L}_{sitcalc}$[5] is regressable iff*

1. *Each term of sort situation that is mentioned by W has the syntactic form $Do([\alpha_1, \ldots, \alpha_n], S_0)$ for some $n \geq 0$, where $\alpha_1, \ldots, \alpha_n$ are terms of sort action.*
2. *For each atom of the form $Poss(\alpha, \sigma)$ mentioned by W, α has the form $A(t_1, \ldots, t_n)$ for some n-ary action function symbol A of $\mathcal{L}_{sitcalc}$.*
3. *W does not quantify over situations.*
4. *W does not mention the predicate symbol $\sqsubset$, nor does it mention any equality atom $\sigma = \sigma'$ for terms σ, σ' of sort situation.* ■

The regression theorem (Levesque et al., 1998) states that $\mathcal{D} \models W$ iff $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[W]$, with W a regressable sentence of $\mathcal{L}_{sitcalc}$ and $\mathcal{D}$ a basic action theory. This means that the evaluation of regressable sentences can be reduced to a theorem proving task in the initial theory $\mathcal{D}_{S_0}$ together with unique names axioms for actions. No successor state, precondition or foundational axioms are needed for this task, then.

3.4.2 Golog

Based on the situation calculus, Levesque et al. (1997) presented the high-level programming language Golog., It is designed as a specification language for complex behaviour in dynamic domains. Since classical planning as in STRIPS (Fikes and Nilsson, 1971) is known to be computationally demanding in general, deriving complex behaviours with hundreds of actions by planning alone is impractical. That is why Golog tries to find a

[4]Sentences uniform in s are sentences which do not quantify about situations, nor mention $Poss$ or $\sqsubset$. "$\sqsubset$" stands for an ordering relation on situations and is needed in the foundational axioms (see for example (Reiter, 2001)).

[5]We leave out the formal definition of $\mathcal{L}_{sitcalc}$ and refer to (Pirri and Reiter, 1999; Reiter, 2001).

compromise between planning and programming (the name GOLOG derives from "alGOL in LOGic"). The programmer can specify the behaviour just like in ordinary imperative programming languages but she can also leave decision to the agent by using constructs for the non-deterministic choice of action or argument. This is done by providing the robot or agent with a basic action theory. The amount of planning (projection) that is used lies in the hand of the programmer. With this, one has a powerful language for specifying the behaviours of a cognitive robot or agent.
As an example consider a robot with the goal to deliver a thank-you letter to a user $Alex$. A GOLOG program for such a task could use a procedure as the following:

```
proc deliver_letter
  navigate($Post\_office$); pick_up($Letter$);
  if $inOffice(Alex)$ then deliver_letter($Alex$)
  else do_nothing endif
endproc
```

This simplistic program shows some sequences of primitive actions as well as control constructs like if-then-else or procedures.
The semantics of GOLOG is defined by a macro Do that expands to a formula in the situation calculus. Given the program δ which specifies the behaviour of the robot, the interpreter tries to find a proof that the basic action theory $\mathcal{D}$ entails that starting in S_0 the program δ leads to a situation s that satisfies the goals. That can be formulated as

$$\mathcal{D} \models (\exists \delta, s).Do(\delta, S_0, s) \wedge Goals(s)$$

As a side-effect of the proof we get an executable GOLOG program, or more precisely, an action sequence that we need to execute to reach the goals.
Let us consider some examples of the Do-macro.

- Primitive action

$$Do(a, s, s') \doteq Poss(a[s], s) \wedge s' = do(a[s], s).$$

  Note that $a[s]$ denotes the action term with its situation argument restored.

- Sequence

$$Do([\delta_1; \delta_2], s, s') \doteq \exists s''.Do(\delta_1, s, s'') \wedge Do(\delta_2, s'', s').$$

- Non-deterministic choice of action

$$Do((\delta_1 | \delta_2), s, s') \doteq Do(\delta_1, s, s') \vee Do(\delta_2, s, s').$$

- Conditional

$$Do(\textbf{if } \varphi \textbf{ then } \delta_1 \textbf{ else } \delta_2 \textbf{ endif}, s, s') \doteq Do([\varphi?; \delta_1 | \neg\varphi?; \delta_2], s, s').$$

GOLOG has successfully been deployed, for instance to control museum tour guide robots (Burgard et al., 1999), even if making GOLOG actually work on a real robot required some effort first (Hähnel et al., 1998).

The original GOLOG interpreter works in an "off-line" fashion. That is, given a program, the interpreter tries to find a sequence of actions which constitutes a legal execution of the entire program at once. Only then it executes that sequence in the real world. Especially for large programs this can be a problem in dynamic scenarios, because during execution the world might change from what it was when that sequence of action was created. That is why (De Giacomo and Levesque, 1999) propose an incremental interpreter. It features *on-line* execution of programs where any action that is selected is executed immediately. Such programs are evaluated using a *transition semantics*.

3.4.2.1 Transition Semantics

In the evaluation semantics programs macro-expand to formulas in the situation calculus. For example, $Do(\exists x.broken(x), s, s')$ expands to the formula $\exists x.broken(x, s') \wedge s = s'$. This way, we avoid questions like what $broken(x)$ is logically. On the downside, it is not possible to quantify over macros, in particular, over programs. Quantifying over programs however is fundamental in the transition semantics. Therefore, De Giacomo et al. (1997) give up the macro view of programs in favour of encoding them as first-order terms in the logical language. Further, they define $Trans$ and $Final$ predicates (instead of the Do macro from the evaluation semantics) in order to cope with program variables.

The program is interpreted in a step-by-step fashion where a transition relation defines the transformation from one step to another. In this *transition semantics* a program is interpreted from one *configuration* $\langle \delta, s \rangle$, a program σ in a situation s, to another configuration $\langle \delta', s' \rangle$ which results after executing the first action of δ, where δ' is the remaining program and s' is the situation resulting from the execution of the first action of δ. The one-step *transition relation* $Trans$ defines successor configurations for each program construct. In addition, another predicate $Final$ is needed to characterize final configurations, which are those where a program is allowed to legally terminate.

Let us consider the definitions of the same examples as given above.

- Primitive action

$$Trans(\alpha, s, \delta, s') \equiv Poss(\alpha[s], s) \wedge \delta = nil \wedge s' = do(a, s)$$

- Sequence

$$\begin{aligned} Trans([\delta_1; \delta_2], s, \delta, s') \equiv\ & Final(\delta_1, s) \wedge \\ & Trans(\delta_2, s, \delta, s') \vee \exists \delta'.\delta = (\delta'; \delta_2) \wedge Trans(\delta_1, s, \delta', s) \end{aligned}$$

- Non-deterministic choice of action

$$Trans(\delta_1 | \delta_2, s, \delta, s') \equiv Trans(\delta_1, s, \delta, s') \vee Trans(\delta_2, s, \delta, s')$$

- Conditional

$$\begin{aligned} & Trans(\textbf{if } \phi \textbf{ then } \delta_1 \textbf{ else } \delta_2, s, \delta, s') \equiv \\ & \quad \phi[s] \wedge Trans(\delta_1, s, \delta, s') \vee \neg\phi[s] \wedge Trans(\delta_2, s, \delta, s') \end{aligned}$$

The above definitions used a termination condition $Final$ which defines final configurations. Those configurations are final where no further transition is possible and the program terminates. As examples for $Final$ we show the predicates for the empty program and for the constructs above.

- Empty program
$$Final(nil, s) \equiv true$$
- Primitive action
$$Final(\alpha, s) \equiv false$$
- Sequence
$$Final([\delta_1; \delta_2], s) \equiv Final(\delta_1, s) \wedge Final(\delta_2, s)$$
- Non-deterministic choice of action
$$Final(\delta_1 | \delta_2, s) \equiv Final(\delta_1, s) \vee Final(\delta_2, s)$$
- Conditional
$$Final(\textbf{if } \varphi \textbf{ then } \delta_1 \textbf{ else } \delta_2 \textbf{ endif}, s) \equiv \varphi[s] \wedge Final(\delta_1, s) \vee \neg\varphi[s] \wedge Final(\delta_2, s)$$

We only sketched the transition semantics here. $Trans$ and $Final$ predicates exist for all constructs available in GOLOG. For a concise overview of the transition semantics we refer the interested reader for example to (De Giacomo and Levesque, 1999; De Giacomo et al., 2000, 2001).

3.4.2.2 Extensions and GOLOG Dialects

There exist various extensions to and dialects of the original GOLOG interpreter. Another dialect of GOLOG that uses the transition semantics is INDIGOLOG (De Giacomo et al., 2001). Besides allowing for on-line execution and off-line projection it also features notions for sensing actions using so-called guarded action theories. Instead of successor state axioms INDIGOLOG uses two sets called guarded successor state axioms and sensed fluents axioms. The guards in each of the two axioms are used to limit the applicability to certain situations.

Two further extension to achieve more realistic logic-based controllers for robots are CCGOLOG and PGOLOG (Grosskreutz, 2002). CCGOLOG introduces *continuous change*, a notion that allows for implementing *continuous fluents* as fluents whose values change as a function of time. By specifying so-called *t-functions*, the value of a continuous fluent can be determined as a function of time using a function val. An additional action $waitFor$ further allows for condition-bounded actions. The idea is that $waitFor(\tau)$ terminates when the condition τ holds. PGOLOG introduces *probabilistic projection* over programs. That is, using the statement $prob(p, \sigma_1, \sigma_2)$ one can express that with probability p the program σ_1 is executed and with the converse probability $1-p$ program σ_2 is executed. To do this, the $Trans$ predicate is replaced with functions $trans$ over probabilities. These functions return a value in $[0, 1]$ that represents the probability of a certain transition. Further, PGOLOG implements the belief $Bel(\varphi, s) = p$ as it was proposed by Bacchus et al. (1999). It allows to express an agent's belief about the formula φ in situation s. For a detailed account of CCGOLOG and PGOLOG we refer to Grosskreutz (2002).

Off-line Decision-Theoretic Golog In (Boutilier et al., 2000) the dtGolog interpreter is proposed. It features an integration of *decision-theoretic planning* (DTP) (Boutilier et al., 1999) using a *Markov-Decision Process* (MDP). We introduced MDPs in Section 3.1.3. Recall that formally, a fully observable finite horizon MDP $M =< S, A, T, R >$ with S being a set of states, A a finite set of actions, T a transition model, and R a real-valued reward function is specified as follows. In dtGolog, the situation term implicitly defines the set of states of the MDP. The domain axiomatization yields the set of actions and the successor state axioms define the transition model. As part of the optimization theory a reward function has to be specified that evaluates how desirable a situation is. dtGolog further introduces a notion for *stochastic actions* to the Golog language. The intuition here is that an action can have multiple possible outcomes like with stochastic actions in MDPs. When the action is executed in the real world then, nature chooses one of the possible outcomes. These outcomes can be modelled as deterministic actions which is why dtGolog only needs to introduce a $choice$ predicate which specifies the choices that nature has to pick from.
To solve the MDP created like described above Boutilier et al. (1999) use *decision-tree search*. Starting from an initial state s_0 all possible actions $a \in A$ are 'executed'. Since these actions are stochastic, for every possible outcome of every action a corresponding successor state is generated and the edge to that state is labelled with the outcome's probability accordingly. In all those successor states again all possible actions are executed and their possible outcomes are added as child nodes like described above. The leaf nodes of the tree generated with this procedure get assigned with the reward of the particular state. For intermediate nodes, action nodes get their value computed as the child nodes' values weighted with the probability of the corresponding outcome. State nodes are assigned the maximum of their child nodes. The action that leads to the maximal value at every stage is part of the optimal policy then.

The high-level control of the domestic service robot that we develop and describe in this thesis is based on a variant of Golog called Readylog (Ferrein and Lakemeyer, 2008).

3.4.3 The Robot Programming and Plan Language Readylog

Readylog (Ferrein and Lakemeyer, 2008; Ferrein, 2010b) is the variant of Golog that we use for the high-level behaviour specification of our domestic service robot.
The aim of designing the language Readylog was to create a Golog dialect which supports the programming of the high-level control of agents or robots in dynamic real-time domains such as robotic soccer. Readylog makes use of Reiter's BATs as described earlier. It features a transition semantics and it borrows ideas from several Golog dialects. Besides providing the standard constructs from Golog (Levesque et al., 1997), Readylog, among others, provides sensing and on-line execution borrowed from indiGolog (De Giacomo et al., 2001), it allows for passive sensing as proposed in (Grosskreutz and Lakemeyer, 2001) for ccGolog, it supports continuous change (Grosskreutz and Lakemeyer, 2003), it features interrupts and it allows for parallelism in form of interleaved concurrency as available in conGolog (De Giacomo et al., 2000), and it has methods in place for probabilistic projection into the future like pGolog (Grosskreutz, 2000). An overview of the constructs available in Readylog is given in Figure 3.6.

nil	empty program
α	primitive action
$\varphi?$	wait/test action
$waitFor(\tau)$	event-interrupt
$[\sigma_1;\sigma_2]$	sequence
$\textbf{if } \varphi \textbf{ then } \sigma_1 \textbf{ else } \sigma_2 \textbf{ endif}$	conditional
$\textbf{while } \varphi \textbf{ do } \sigma \textbf{ endwhile}$	loop
$\textbf{withCtrl } \varphi \textbf{ do } \sigma \textbf{ endwithCtrl}$	guarded execution
$\sigma_1 \,\|\|\, \sigma_2$	prioritized execution
$\textbf{forever do } \sigma \textbf{ endforever}$	infinite loop
$\textbf{whenever}(\tau, \sigma)$	interrupt triggered by continuous function
$\textbf{withPol}(\sigma_1, \sigma_2)$	prioritized execution until σ_2 ends
$\textbf{prob}(p, \sigma_1, \sigma_2)$	probabilistic execution of either σ_1 or σ_2
$\textbf{interrupt}$	interrupts
$pproj(c, \sigma)$	probabilistic (off-line) projection
$\{\textbf{proc } P_1(\vec{\vartheta}_1)\sigma_1 \textbf{ endproc}; \cdots ; \textbf{proc } P_n(\vec{\vartheta}_n)\sigma_n \textbf{ endproc}\}; \sigma_0$	procedures
$solve(h, f, \sigma)$	initiate decision-theoretic optimization over σ
$\sigma_1 \,\|\, \sigma_2$	non-deterministic (dt) choice of programs
$(\pi\ \vec{x})[\sigma]/pickBest(\vec{x}, \sigma, h)$	non-deterministic (dt) choice of arguments

Figure 3.6: Overview of Readylog constructs

The idea of GOLOG to combine planning with programming was accounted for in READYLOG by integrating an on-line version of decision-theoretic planning in the spirit of (Boutilier et al., 2000); only partially specified programs which leave certain decisions open, which then are taken by the controller based on an optimization theory, are needed.

We do not want to get into the details of the formal semantics here, as only little of it is needed to understand the remainder of this thesis. We refer the interested reader to (Ferrein and Lakemeyer, 2008) for the complete formal definition of the language. GOLOG languages come with run-time interpreters usually programmed in Prolog. Also, a READYLOG implementation is available in Prolog. However, there are efforts in developing non-prolog implementations for GOLOG (Ferrein, 2010a; Ferrein and Steinbauer, 2010).

READYLOG was deployed for tasks ranging from controlling bots in interactive computer games (Jacobs et al., 2005) to cognitive robotics for intelligent soccer robots (Ferrein and Lakemeyer, 2008). We use READYLOG as the high-level controller for our domestic service robot. How the high-level control work together with methods for human-robot interaction and what extensions are need to develop cognitive interactive service robots for domestic domains will be subject of the upcoming chapters.

4 Human-Robot Interaction

In this chapter we extend our robot beyond the baseline of a mobile robot towards a cognitive interactive robot. That is, we equip our mobile platform with means to engage in and successfully perform human-robot interaction (HRI). This chapter is not a general treatment of HRI. Instead, it discusses the specific efforts we made in this regard and it serves as a proof-of-concept for making an autonomous mobile robot capable of natural interaction with the humans around it. For a general overview on the field of socially interactive robots and HRI we refer to (Fong et al., 2003a) and (Goodrich and Schultz, 2007), respectively.
With respect to the four classes (or levels) of social robots as introduced by Breazeal (2003) we stay at what is being referred to as the *social interface*. That is to say, our robot uses social cues and ways of communication that make it natural and intuitive for humans to interact with the robot but it does not go beyond that. A step towards this next level – called socially receptive – would be for the robot to receive and to incorporate input from human, for example with learning motor skills from demonstration. If at all, we slightly touch that level, e.g., when our robot integrates new persons into its knowledge base by learning their faces.

Our domestic service robot is supposed to interact with laymen. Hence, it needs to be operable by such laymen and the interaction between the human and the robot needs to be as *natural* and intuitive as possible. We go into detail about three efforts in this regard, i.e., we present three important human-robot interaction components. Those components were selected since they represent (perhaps the most) important modalities in human-robot interaction, namely Speech recognition (Doostdar et al., 2008), face detection, recognition, and learning (Belle et al., 2008), and gesture recognition (Schiffer et al., 2011a). This is because these modalities make for a natural and human-like behaviour of the robot (Scheutz et al., 2007) and do not pose any restrictions on the human in communicating with the robot. Then we briefly sketch additional modules that our robot is equipped with, namely for speech synthesis, people detection (and tracking) and to display a face (and other information). Finally, we present an application of the robot (Schiffer et al., 2012a) using the aforementioned capabilities in an interactive demo scenario.

4.1 Speech Recognition

Face to face communication between humans is mostly done using speech. Hence, *speech recognition* is a crucial ability for a mobile service robot that should communicate with humans. However, spoken language is a natural and convenient way to instruct a robot only if it is processed reliably. Modern speech recognition systems can achieve high recognition rates, but their accuracy often decreases dramatically in noisy and crowded environments. This is usually dealt with by either requiring an almost noise-free environment or by placing the microphone very close to the speaker's mouth. Although we already assume the latter, all requirements for a sufficiently high accuracy cannot always be met in realistic scenarios.

4.1.1 Speech Recognition in Domestic Service Robotics

As already mentioned, our target applications are in the domestic service robotics domain as it is replicated in the ROBOCUP@HOME league (van der Zant and Wisspeintner, 2005), where robots assist humans with their everyday activities in a home-like environment. Any interaction with a robot has to be done in a natural fashion. That is to say, instructions issued to the robot may only be given by means of gestures or natural spoken language. An important property of the domain, especially at a competition, is the high amount of non-stationary background noise and background speech. A successful speech recognition system in ROBOCUP@HOME must be able to provide robust speaker-independent recognition of mostly command-like sentences. For one, it is important that commands given to the robot are recognized robustly. For another, spoken language not directed to the robot must not be matched to an instruction for the robot. This is a non-trivial task in an environment with a high amount of background noise. That is why most teams use a head mounted microphone for their speech recognition. Still, it is not easy to determine which audio input is actually addressed to the robot and which one is not. This is even more so, since within a competition there usually is a person that describes to the audience what is currently happening in the arena via loudspeakers. The words used for the presentation often are very similar if not even the same used to instruct the robot. This complicates the task of robust speech recognition even more.
We propose and implement an architecture that tackles the problem of robust speech recognition in the above setting. It comprises two steps. First, we use a threshold based close speech detection module to segment utterances targeted at the robot from the continuous audio stream recorded by the microphone. Then, we decode these utterances with two different decoders in parallel, namely one very restrictive decoder based on finite state grammars and a second more lenient decoder using N-grams. We do this to filter out false positive recognitions by comparing the output of the two decoders and rejecting the input if it was not recognized by both decoders.

4.1.2 Statistical Speech Recognition

Most statistical speech recognition systems available today use *hidden Markov models* (HMMs). For a given vector of acoustical data x they choose a sequence of words w_{opt} as best-hypothesis by optimizing

$$w_{opt} = \arg\max_{w} p(w|x) = \arg\max_{w} \frac{p(x|w) \cdot p(w)}{p(x)} = \arg\max_{w} p(x|w) \cdot p(w). \tag{4.1}$$

Here $p(w|x)$ is the posterior probability of w being spoken, the fundamental Bayes' decision rule is applied to it. The constant normalization probability $p(x)$ can be omitted for the maximization. $p(x|w)$ denotes the probability of observing the acoustical data x for the assumption of w being spoken. This is given to the recognizer by the *acoustic model*. $p(w)$ denotes the probability of the particular word-sequence w occurring. This prior probability is provided to the recognizer by the so-called *language model*. The language model can either be represented by a grammar, e.g. a *finite state grammar* (FSG), or by means of a statistical model, mostly in form of so-called N*-grams* that provide probabilities for a word dependent on the previous $(N-1)$ words. Common speech-recognizers use 3-grams, also called *TriGrams*.

Standard statistical speech recognition systems process a given speech utterance time-synchronously. Each time-frame, possible sub-word-units (modelled by HMM-states) and word-ends are scored considering their acoustical probability and their language model probability. Most of the possible hypotheses score considerably worse than the best hypothesis at this time frame and are pruned away. In the search for a best hypothesis information about possible near alternatives can be kept to allow for useful post-processing. For each time-frame, the most probable hypotheses of words ending at that frame are stored along with their acoustical scores. This information can be appropriately represented by a directed, acyclic, weighted graph called *word-graph* or *word-lattice*. Nodes and edges in the graph denote words, their start-frames and their acoustic likelihoods. Any path through the graph, starting at the single start-node and ending in the single end-node, represents a hypothesis for the complete utterance. By combining the acoustical likelihoods of the words contained along this path with the language model probability we obtain the score $p(x|w) \cdot p(w)$. The so-called N*-best list* contains all possible paths through the word-graph that were not pruned in the search, ordered by their score.

The language model used in searching for hypotheses largely influences the performance of a speech recognition system. Thus, it is crucial to choose a model appropriate for the particular target application to achieve sufficiently good results. On the one hand, FSG-based decoders perform good on sentences from their restricted grammar. On the other hand, they get easily confused for input that does not fit the grammar used. This can lead to high false recognition rates. N-gram based language models are good for larger vocabularies, since utterances do not have to follow a strict grammar.

4.1.3 System Overview and Related Work

For our target application we are confronted with a high amount of non-stationary background noise including speech similar to the vocabulary used to instruct the robot. Only using an FSG-based decoder would lead to high false recognition rates. We aim to eliminate false recognitions with a system that exploits the properties of different language models described above. In a first step, we try to segment utterances that are potential speech commands issued to the robot from the continuous audio stream recorded by the robot's microphone. Then, we decode those utterances using two decoders in parallel, one FSG-based and a second TriGram-based one to combine the benefits of both. An overview of our system's architecture is shown in Figure 4.1.

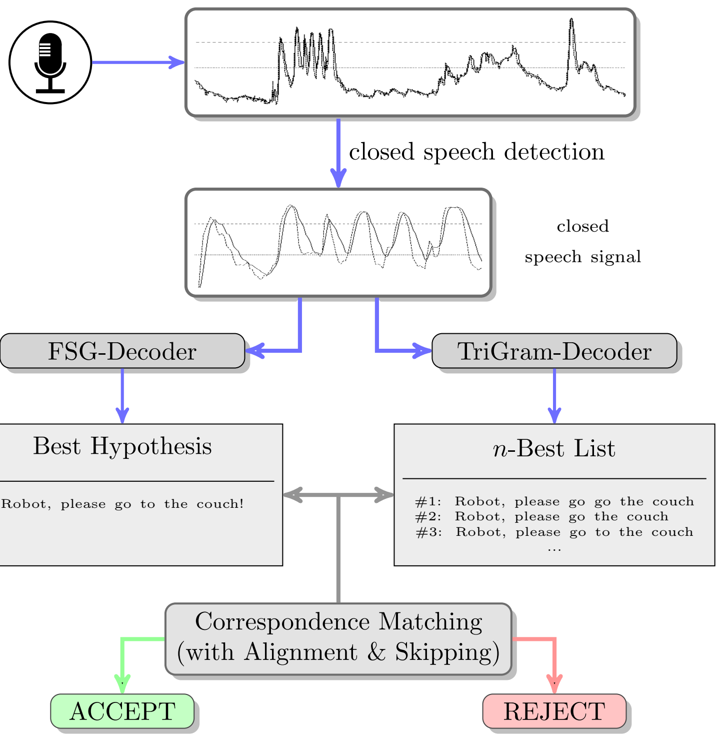

closed speech detection
closed
speech signal
FSG-Decoder
TriGram-Decoder
Best Hypothesis
Robot, please go to the couch!
n-Best List
#1: Robot, please go go the couch
#2: Robot, please go the couch
#3: Robot, please go to the couch
...
Correspondence Matching
(with Alignment & Skipping)
ACCEPT
REJECT

Figure 4.1: Architecture of our dual decoder system

4.1.3.1 Segmentation of close speech sections

We employ a module that is supposed to segment *close speech sections* from the continuous live stream, i.e. sections where the main person speaks closely into the microphone. We call this *close speech detection* (CSD). Doing this provides us with two advantages. First, with the (reasonable) presupposition that the speech to be recognized, we call it *positive speech*, is being carried out close to the microphone, we can discriminate it from other (background) speech events that are not relevant and thus may cause false recognitions. These false recognitions would be wrongly matched to a speech command for the robot. Second, the performance of speech recognition engines like SPHINX (Huang et al., 1993) increases considerably, if the speech input occurs in closed utterances instead of a continuous stream. Furthermore, we are able to reduce the computational demands for speech recognition if we only process input of interest instead of decoding continuously. This is especially useful for mobile robotic platforms with limited computing power.

4.1.3.2 Multiple decoders

As already mentioned, different types of decoders exhibit different properties we would like to combine for our application. For one, we are interested in the high accuracy that very restricted FSG decoders provide. But we cannot afford to accept a high rate of falsely recognized speech that is then probably matched to a legal command. For another, TriGram-based decoders are able to reliably detect words from a larger vocabulary and they can generate appropriate hypotheses for utterances not coming from the grammar, i.e. that are not positive speech. However, that comes with the drawback of an increased error rate in overall sentence recognition. Still, a sentence at least similar to the actual utterance will very likely be contained in the N-best list, the list of the n hypotheses with the highest score. By decoding with an FSG and a TriGram decoder in parallel we seek to eliminate each decoders drawbacks retaining its benefits. We can look for similarities between the FSG's output and the N-best list of the TriGram decoder. This allows detecting and rejecting *false positive recognitions* from the FSG decoder.

In principle, any automatic speech recognition system that provides the ability to use both, FSG and N-gram based decoding with N-best list generation, could be employed within our proposed architecture. We chose to use SPHINX 3 because it is freely available open source software, it is under active development with good support, it is flexible to extend, and it provides techniques for speaker adaption and acoustic model generation. For an overview of an earlier version of the SPHINX system we refer to (Huang et al., 1993).

4.1.3.3 Related work

For speech detection, also referred to as speech activity detection (SAD), endpoint detection, or speech/non-speech classification, speech events have to be detected and preferably also discriminated against non-speech events on various energy levels. There has been work on this problem in the last decades, some of which also employs threshold-approaches like an earlier work by Lamel et al. (1981) detecting energy pulses of isolated words, or work by Macho et al. (2005). One of the main differences to our approach is that they dynamically adapt the threshold to detect speech on various energy levels. For our application, however, it is more preferable to use a static threshold since the environmental conditions may vary but the characteristics of the aural input of interest do not. Furthermore, we dynamically allocate the distance allowed between two spoken words and we apply simple smoothing of a signal's energy-value sequence. For a more general solution to the problem of speech activity detection threshold-based approaches often do not work since they are not robust enough with higher noise levels. More robust approaches use, for example, linear discriminant analysis (LDA) like (Padrell et al., 2005) and (Rentzeperis et al., 2006), or HMMs on Gaussian Mixtures (Ruhi Sarikaya, 1998). Our aim is to use detection of close speech only as a pre-processing step before decoding. That is why we do not want to put up the additional costs for these more sophisticated approaches.

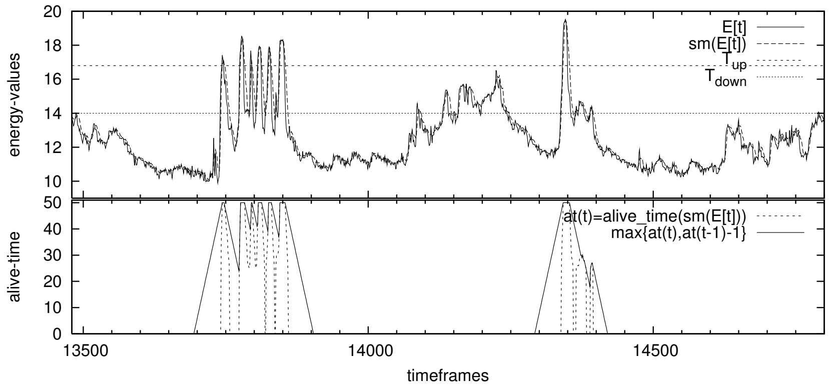
To improve the accuracy of speech-recognition systems on grammar-definable utterances while also rejecting false-positives, usually in-depth knowledge of the low-level HMM-decoding processes is required. There has been work on integrating N-grams and finite state grammars in one decoding process for detecting FSG-definable sentences (Lin et al., 1997). They assume that the sentences to detect are usually surrounded by carrier phrases. The N-gram is aimed to cover those surrounding phrases and the FSG is triggered into the decoding process if start-words of the grammar are hypothesized by the N-gram. To reject an FSG-hypothesis they consult thresholds on acoustical likelihoods of the hypothesized words. Whereas this approach requires integration with low-level decoding processes, our dual-decoder approach only performs some post-processing on the hypotheses of the N-best list. In combination with the CSD front-end we achieve acceptable performance for our application without modifying essential parts of the underlying system. Instead of using two decoders in parallel, a more common method could be to use an N-gram language model in a first pass and to re-score the resulting word-graph or N-best list using an FSG based language model afterwards. However, independently decoding with an FSG-based decoder can be expected to provide higher accuracy for the best hypothesis than the best hypothesis after re-scoring a word-graph with an FSG language model. It would be promising, though, to combine a two-pass approach or our dual-decoder with a method for statistically approximating confidences (Wessel et al., 2001) (in terms of posterior probabilities) of hypothesized words given a word-graph. A reliable confidence measure would provide a good method for rejection with a threshold.

4.1.4 Close Speech Detection

Our approach to detect and segment sections of close speech from a continuous audio stream is quite simple. It makes use of the straightforward idea that sounds being produced close to the microphone exhibit considerably high energy levels. The *energy values* of an audio input are provided when working with speech recognition systems as they extract cepstral coefficients as features from the acoustic signals. The first value of the cepstral coefficients can be understood as the signal's logarithmic energy value. Close speech is detected by first searching for energy values that exceed some upper threshold. Then, we determine the start and the end-point of the segment. Therefore, we look in forward and backward direction for points where the speech's energy values fall below a lower threshold for some time. Note that this straightforward approach can only detect speech carried out close to the microphone. However, this is expressly aimed for in our application since it provides a simple and robust method to discriminate between utterances of the "legal speaker" and other nearby speakers as well as background noise.

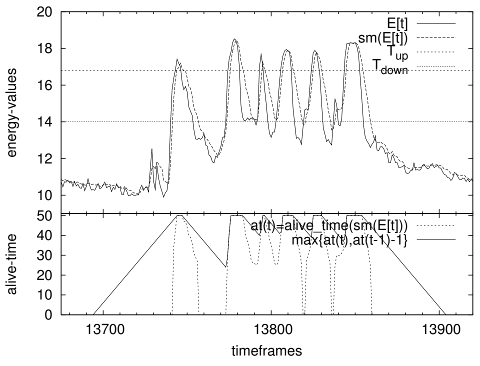
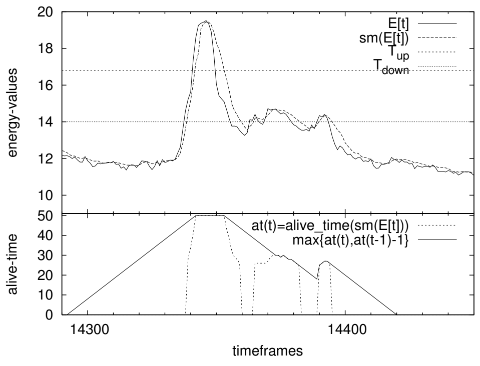
4.1.4.1 Detailed Description

Examining a sequence of energy values, speech segments are characterized by adjacent heaps (see Figure 4.2: E[t]). For our aim of detecting close speech segments we use two thresholds, namely T_{up} and T_{down}. The first threshold T_{up} mainly serves as a criterion for detecting a close-speech-segment when some energy values exceed it. Thus, T_{up} should be chosen so that for close speech segments some of the heaps are expected to exceed T_{up} while other segments do not.
After this initial detection, the beginning and the end of the speech segment have to be determined such that the resulting segment contains all heaps adjacent to the initial peak. Therefore, starting from the detection point, we proceed in forward and backward direction. We search for points where the energy value drops below the second threshold T_{down} and does not recover again within a certain amount of time-frames. We thereby identify the beginning and the end of the speech segment, respectively. T_{down} should be chosen largest so that still all heaps of a close speech section are expected to exceed it and lowest so that the energy-level of the background noise and most background sound events do not go beyond T_{down}. The amount of time-frames given for recovering again represents the maximal distance we allow between two heaps, i.e. between two consecutive words. We call this the *alive-time*. We further enhance this approach by smoothing the sequence of energy levels before processing it and by dynamically allocating the time to recover from dropping below T_{down}.



(a) Continuous stream with two utterances of interest

bg-speech – **Oh,I forgot my cup!** – Should I go an get it for you? – **Yes-please!** – *bg-speech*

(b) First utterance "Oh, I forgot my cup!"

(c) Second utterance "Yes please!"

Figure 4.2: Segmentation of close speech segments

Smoothing the energy values The energy-value-sequence is smoothed (cf. $sm(E,t)$ in Figure 4.2) to prevent punctual variations to take effect on the detection of speech-segments and the determination of the alive-time. We compute the smoothed values by averaging over the current energy-value and the three largest of the previous six energy-values, i.e. for an energy-sequence E and time-frame t we use the smoothing function:

$$sm(E,t) = \frac{1}{4} \cdot \max\{E[t] + E[t_1] + E[t_2] + E[t_3] \mid t_i \in \{t-1, \ldots, t-6\}\}.$$

Start/End-point detection The amount of time-frames given to recover from falling below T_{down} is not fixed but is determined dependent on the height of the last heap's peak and the distance to this peak. Thus, the closer the peak of the last heap is to T_{down}, i.e. the less the confidence is for the last heap being produced by a close speech, the less time-frames we grant before the next heap must occur (cf. alive-time of right-most heap in Figure 4.2c). This helps to prevent that background sounds which intersect with the close-speech segment or directly succeed/precede them and which exceed T_{down} (like a nearby speech) cause the fixation of the start or end point of a close-speech segment to be postponed over and over again. For energy-values greater than T_{up} we assign the alive-time AT_{up}, for the value T_{down} we assign the alive-time AT_{down} and for values between T_{up} and T_{down} we calculate a time-value by linearly interpolating between AT_{up} and AT_{down}:

$$\text{alive\_time}(v) = \begin{cases} AT_{up} & , v \geq T_{up} \\ \frac{AT_{up}-AT_{down}}{T_{up}-T_{down}} \cdot (v - T_{down}) + T_{down} & , T_{down} < v < T_{up} \\ 0 & , v \leq T_{down} \end{cases}$$

After a close-speech segment is detected we proceed in forward/backward direction (see Figure 4.2b and Figure 4.2c). For each time-frame t, we compute the alive-time at as the maximum of the value associated with the smoothed energy-value $at(sm(E,t))$ and the alive-time value chosen in the previous time-frame minus one ($at(t-1)-1$). Obviously, when the energy-value-sequence falls below T_{down} the alive-time decreases each time-frame. If 0 is reached before the values recover again, we determine the start and end point of the close-speech-segment at that frame. As soon as we have determined the start point of a segment, we can start passing the input to the decoders. This drastically increases the reactivity of the system. For now, we manually define the actual thresholds T_{up} and T_{down} based on the environmental conditions at a particular site. We fixed AT_{up} at 50 time-frames ($500\,\text{ms}$) and AT_{down} at 25 time-frames ($250\,\text{ms}$). These values were determined empirically.

4.1.5 Dual Decoding

As mentioned in Section 4.1.2, in statistical SR-systems the optimization of the posterior probabilities $p(w|x)$ is approached by maximizing the scores $p(x|w) \cdot p(w)$ where the likelihood $p(x|w)$ is given by an acoustical model and the prior probability $p(w)$ is provided by a language model. The set of utterances to recognize in our target application per task is quite limited and very structured. It can thus appropriately be defined by a grammar. Consequently, a language model based on a finite state grammar (FSG) seems most suitable. Even though we assume our CSD already filtered out some undesired input, we are confronted with a high rate of false positive recognitions of *out-of-grammar* (OOG) utterances which we cannot afford and have to take care of. Given an OOG utterance x, a restricted FSG-based decoder cannot come around to hypothesize x as an *in-grammar* (IG) sentence w (or prefixes of it), since the word-sequence probability for all other sentences $p(w')$ is 0 because they are not part of the grammar. This holds even if we suppose the acoustical probability $p(x|w)$ for an IG-sentence to be low. The acoustical probability mainly plays a decisive role for discrimination between different utterances w from within the language model. A TriGram-based language model contains many more possible utterances, hence a decoder using such a language model can also hypothesize those other sentences when it is given an utterance that is OOG (with respect to the FSG). Unfortunately, it cannot provide us with an accurate *best hypothesis* reliably enough. That is, the correct sentence w_x for a given IG-utterance x will not be the best hypothesis often enough. Otherwise, we could just stay with a single TriGram-based decoder for recognition. But we can utilize a TriGram language model to help rejecting OOG utterances hypothesized by the FSG-based decoder. For this the TriGram has to comprise a larger vocabulary than the specific grammar and provide not-too-low probabilities for appropriate combinations of these words, i.e. for OOG sentences.

Let us consider a false positive recognition where an OOG-utterance x is falsely recognized as an IG-sentence w by the FSG-based decoder. With an appropriate modelling of the TriGram we can assume it to provide OOG-sentences w' with significant probabilities within its language-model. We can also assume that the acoustical probability $p(x|w')$ for those w' corresponding to the actual utterance exceed the acoustical probability of each falsely hypothesized IG-sentence w considerably. So for the TriGram-based decoding process the comparatively low acoustical probability $p(x|w)$ causes some words of w to be pruned away around the corresponding time-frames they were hypothesized at by the FSG-decoder. Hence, the word-graph and thus the N-best list produced by the TriGram-based decoder will not contain the sequence w. On the other hand, given an IG utterance x and its sentence w_x, the comparatively high acoustical probability $p(x|w)$ (in combination with a still sufficiently high language probability) likely prevents that words of w_x are pruned in the decoding process. Therefore, the N-best-list will still contain w_x.
Consequently, we accept the hypothesis of the FSG-based decoder, if it can be matched with some hypothesis within the N-best list of the TriGram decoder. To not compare utterances from different instances of time, the matching also takes word-start-frames into consideration.

Hypothesis matching

For the matching of the FSG-best hypothesis w_{FSG} with one of the N-best hypotheses of the TriGram w_n, we require that the words of w_{FSG} occur in the same order in w_n. The difference in the start-frames of the matched words shall not exceed a predefined maximal offset. For this, we simply iterate through the word-sequence w_n. If the current word of w_n matches with the current word of w_{FSG} (considering the maximal offset allowed), we proceed to the next word of w_{FSG}. If all words of w_{FSG} are processed, we accept the FSG's hypothesis. Within this matching, we always omit hypothesized filler-words like *SILENCE*. For some cases, we experienced that an additional heuristics can improve the acceptance rate. For example, for longer word-sequences w_{FSG} hypothesized by the FSG-based decoder it can be reasonable not to require that all words in w_{FSG} have to be matched on the N-best hypothesis compared to. There is a trade-off between good acceptance rate and good rejection rate when relaxing the matching. Since we only want to make sure that the FSG's hypothesis is not a false positive we argue that enough evidence is given if the FSG's and the TriGram's hypotheses have been similar enough. Therefore, we allow to skip some words of w_{FSG} during the matching dependent on the number of words of w_{FSG} (e.g. in our application we allow to skip 1 word if $|w_{FSG}| \geq 4$, skip 2 words if $|w_{FSG}| \geq 6$ ...). This can be incorporated very easily in our matching procedure we explained above.

4.1.6 Experimental Evaluation

To evaluate our approach we conducted several experiments on speech input recorded in the RoboCup@Home environment during a competition. We use a freely available speaker independent acoustic model for the Sphinx 3 speech engine build with the WSJ-corpus (Seymore et al., 1998). The FSG decoder was run with the specific grammar for the navigation task shown in Table 4.1.

command	=	[salut] instruct TO THE location | STOP
salut	=	ROBOT [PLEASE]
instruct	=	GO | NAVIGATE | DRIVE | GUIDE ME
location	=	ARM CHAIR | PALM [TREE] | WASTE (BASKET | BIN) | TRASH CAN | UPLIGHT | REFRIGERATOR | FRIDGE | COUCH | SOFA | PLANT | BOOKSHELF | SHELF | (COUCH | SIDE | COFFEE | DINNER | DINNING) TABLE | [FRONT] DOOR | LAMP

Table 4.1: Grammar for the navigation task

The performance of the our dual-decoder systems is influenced by several parameters. We adjusted these in such a way, that the trade-off between higher acceptance-rate of IG-utterances and higher rejection-rates of OOG-utterances tends to a higher acceptance rate. That is because we expect to let pass a fairly low amount of OOG speech-like utterances in the close-speech detection step already.

4.1.6.1 Recognition Accuracy

To assess the overall recognition performance of our dual decoder system compared to a TriGram-only and an FSG-only system, we compiled a set of utterances that are legal commands of a particular task in the RoboCup@Home domain. The FSG decoder is using the corresponding grammar of this specific task. The TriGram decoder in our dual decoder system uses a language model constructed from all tasks (excluding the task used for evaluating the rejection of OOG-utterances) of the RoboCup@Home domain with an additional set of 100 sentences of general purpose English. To achieve best performance the TriGram-only decoder uses a language model constructed from navigation sentences only.
In our particular evaluation setup we fed 723 (20.6 minutes) correct commands from the navigation task (cf. Table 4.1) to all three decoders. Table 4.2 shows the accuracy and rejection results. For our dual-decoder, we consider an utterance successful if it is recognized correctly and accepted. The recognition rates are based on the FSG decoder while the rejection rates are based on the matching between the FSG's hypothesis and the first 25 entries of the TriGram's N-best list. In the TriGram-only case, we take the best hypothesis as the recognition output. The results indicate that using two decoders in parallel yields successful processing. Adding up 13.8% of falsely recognized commands and 8.6% of correctly recognized but rejected commands, we receive a total of 22.4% of unsuccessfully processed utterances in comparison to 30.7% of an TriGram-based decoder. The *sentence-error rate* (SER) is a more meaningful measure than the *word-error rate* (WER) here, because we are interested in the amount of sentences containing errors and not in the number of errors per sentence. The 10.2% of falsely recognized but accepted utterances are critical in the sense that they could have caused a false command to be interpreted. Depending on the application, hypotheses that differ from the reference spoken can still result in the same command, e.g. ”ROBOT PLEASE GO TO THE REFRIGERATOR” yields the same command as “DRIVE TO FRIDGE”. To give an idea about possible proportions, for the dual-decoder on our navigation task half of the potentially critical utterances (10.2%) are matched on the same command. For the TriGram-decoder this is the case for one fifth of the 30.7% of all sentence errors. 24.2% (overall) are OOG-sentences and thus are not matched to commands at all. To compare the processing speed, we also measured the *real-time factor* (RTF), i.e., the time it takes to process a signal of duration 1. We achieved an RTF of 1.16 for our dual-decoder

(a) Dual decoder

	rejected	accepted	
recognized	8.6%	77.6%	86.2%
falsely recognized	3.6%	10.2%	13.8%
	12.2%	87.8%	

(b) word-error rate (WER), sentence-error rate (SER), and real-time factor (RTF) for single decoders

	WER	SER	RTF
TriGram only	9.9%	30.7%	0.99
FSG only	4.1%	13.8%	0.24

Table 4.2: Accuracy and rejection results of our dual decoder for legal commands

system on a Pentium M with 1.6 GHz. This is fast enough for our application, since we are given closed utterances by our CSD front-end and we only decode those. RTFs for the single-decoder systems (with relaxed pruning thresholds for best accuracy) on the same machine are listed in Table 4.2b.

4.1.6.2 Rejection Accuracy

To assess the performance of our dual decoder system in rejecting OOG utterances (with respect to the FSG) we collected a set of utterances that are legitimate commands of the ROBOCUP@HOME domain (all tasks) but that do not belong to the specific task the FSG decoder is using. Please note that this is close to a worst case analysis since not all of the utterances that make it to the decoder stage in a real setup will be legal commands at all. In our particular case we took 1824 commands (44 minutes) from the final demonstration task and the manipulation task and fed those commands to an FSG-only system, a TriGram-only system, and our proposed dual decoder system. All three decoders had the same configuration as in the recognition setup above. The FSG decoder for the single case and within our dual-decoder system was using the navigation task grammar (cf. Table 4.1). As can be seen in Table 4.3, the single FSG decoder setup would have matched over 93% of the false positive utterances to valid robot commands. With our dual decoder approach, on the other hand, the system was able to reject more than 82% of those false utterances. With a TriGram-only decoder we would have been able to reject 84%, but this would have come with a prohibitive error rate of more than 30% for correct commands as shown in Table 4.3 and Table 4.2b already.

4.1.7 Summary

We presented a robust speech recognition system for service-robotics applications. We use off-the-shelf statistical speech recognition technology and combine two decoders with different language models to filter out false recognitions which we cannot afford for a reliable system to instruct a robot. The advantages of our system in comparison to more

Decoder	$\mathrm{FP}_{accepted}$	Error rate on correct commands
Single (FSG only)	93.9%	13.8% (SER)
Single (TriGram only)	16.1%	30.7% (SER)
Dual (FSG+TriGram)	17.7%	13.8% (SER) + 8.6% (falsely rejected)

Table 4.3: Acceptance rates of false positive (FP) utterances and error rates on legal commands. In both columns lower numbers are better.

sophisticated approaches are as follows. It provides sufficiently accurate speech detection results as a front-end for ASR-systems. Our approach is computationally efficient and relatively simple to implement without deeper knowledge about speech recognition interiors and sophisticated classifiers like HMMs, GMMs or LDA. It is therefore valuable for people lacking background knowledge in speech recognition and aiming for a robust speech recognition system in restricted domains.

4.2 Face Detection, Recognition, and Learning

Service robots aim at offering assistance to humans in general and to people with disabilities in particular. Such robots socially interact with human beings, i.e., they respond dynamically to requests and communicate. The interaction can be more natural if the robot can identify persons it encounters. We envision that on encountering unknown identities, the robot may introduce itself and add the new identity to its knowledge base. Therefore, a fast and reliable face recognition system is required, which, in a first step, detects faces and, in a second step, recognizes the persons. This task is complicated by the computational limitations of common mobile robots which typically have only modest computing power that additionally have to be shared with other robot components such as motion control and localization. In the literature, face detection and face recognition, typically, are addressed separately, although they share identical structural foundations (Turk and Pentland, 1991; Jones and Viola, 2003).

4.2.1 Overview and Related Work

We approach a one-step system that addresses both face detection and recognition in an integrated framework using *random forests* (RF). The advantages of RFs have been thoroughly investigated (Breiman, 2001) and it has been shown that RFs are fast and have good generalization capabilities. Additionally, we introduce *identity learning*, as an extension to this framework. A collection of face images for a new identity captured by the robot can be added to the knowledge base in real-time, i.e., the robot learns to recognize new persons from that instant. This is made feasible as a result of a very short training time. Similar to other approaches, we use local descriptors, which are known to be an excellent means for face authentication (Paredes et al., 2001). RFs have previously been used for biological image classification (Marée et al., 2007) where, similar to our approach randomly sampled rectangles are used as test candidates in the training procedure. RFs have also been successfully used for general object/image classification (Bosch et al., 2007). RFs have been used for foreground/background segmentation in a video chat application and systematically compared to boosting and

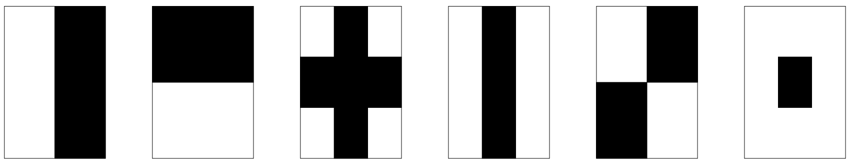


Figure 4.3: The six different Haar features used in our RTs.

bagging classifiers (Yin et al., 2007). An approach most similar to our own has been proposed for real-time gesture recognition. Here, an RF is used for segmentation and subsequently a second RF is used for the recognition (Deselaers et al., 2007).
We compare the detection performance of our approach to the AdaBoost face detector (Jones and Viola, 2003), an excellent implementation of which is freely available in OpenCV.[1] The AdaBoost face detector is often considered a quasi standard (Pham and Cham, 2007) for face detection, comparable to detection with neural networks (Rowley et al., 1998b,a). Results of our recognition are later compared to those achieved with support vector machines (SVM), which have been successfully used for face identification (Guo et al., 2000) before.

4.2.2 Random Forests

As presented in Section 3.1 a random forest is a collection of random trees (RT). Random trees are structurally identical to classical decision trees but are trained differently. During training not an exhaustive search of the possible test candidates is considered but only a randomized subset in order to allow for quickly creating several different and independent RTs. In our approach, we create a large number of randomized test candidates to pick from in the training procedure in each iteration. Here, a randomized test candidate is a local *feature* on a rectangle of random dimensions and at a random location in the training data images.

4.2.2.1 Features

We use Haar features similar to those used in the boosted face detection cascade by Jones and Viola (2003). They allow for fast evaluation and they are known to be good features for face detection. We allow the RT to choose among six different Haar features (depicted in Figure 4.3) and during training, in each iteration a set of these features is generated by choosing the type (one of the six), the size, and the test threshold randomly. A test is successful, if the sum of the pixel grey values in the black area minus the sum of the pixel grey values in the white area is higher than the test threshold. These tests can efficiently be calculated using integral images (Jones and Viola, 2003).

4.2.2.2 Tree Training

An RT is grown in an iterative procedure. In each iteration, a set of L test candidates is randomly generated. The test candidates are applied to all the training samples at a node and the entropy gain is measured on the corresponding split. The best candidate

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

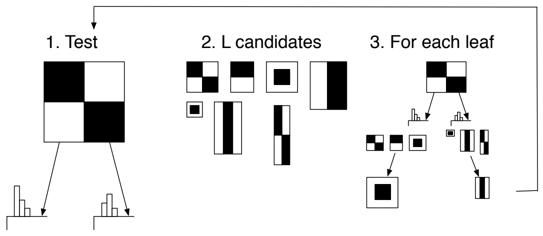


Figure 4.4: Schematic representation of the training method.

is chosen and the left and right branches are appended to the existing node and the procedure continues until the leaf nodes maintain training data of only a single class.
The training data for face detection consists of segmented faces (positives) and patches of background images (negatives). These images are all scaled to a common size and sub-windows (up to a heuristically determined percentage of the image size) are sampled as test candidates. The initialization of the RT is done using a normal training iteration by choosing the candidate among a set of L random test candidates that optimizes the classification according to the entropy gain. To measure the entropy gain we use *class histograms*. The class histogram compares the number of images for each class in a leaf node on a split to the image count in the parent node to calculate an information gain. The candidate with the highest gain is chosen for each node in each iteration. The divided training samples are propagated to the left and right children of a split node. The training process is schematically shown in Figure 4.4. This procedure is repeated until either a predefined threshold is reached or until perfect class separation is reached. RTs are prone to overfitting if trained long enough (i.e., too many nodes added). Averaging classifiers improves the generalization ability and to benefit from both effects, RFs are one possibility (Breiman, 2001). Following the aforementioned ideas, we create an RF by training a set of T RTs simultaneously.

4.2.2.3 Detection

To detect faces in an input image, we use integral images to allow for rapid evaluation of our RTs. Then, each RT is used in a sliding windows manner on an image to determine for each position whether the surrounding area is a face or not. Eventually, in each RT a leaf node is reached delivering a probability (by looking up the relative frequency of the nodes class histogram) for the region to contain a face:

$$\text{FMP} = \frac{\text{number of face images in leaf node}}{\text{total number of images in leaf node}}. \tag{4.2}$$

Each RT is applied in this manner, resulting in T probabilities at each position. These probabilities are fused to determine the absence/presence of a face for each position.
To detect faces of different sizes, either the input image is scaled or the RTs are adapted by scaling the dimensions of tests, which is faster because it does not require for recalculating the integral image.
The fusion of the different RTs at different scales is performed using an aggressive merging step, much simpler than the one proposed by Jones and Viola (2003). Our merging technique finds an *area of interest* by listing neighbours of a detection window. Then, a weighted average over the detected face areas is computed to deliver the final

detection, where we chose the weights from experiments on a preliminary in-house data set.

4.2.2.4 Recognition

For face recognition, we have to discriminate between P known identities with the additional option of classifying a person as unknown. In our integrated joint detection and recognition approach, we use the face/no-face information jointly with the identity labels. Therefore, we first train a normal detection RF as described above, then we add the identity labels and continue training to be able to discriminate among identities (i.e. grow additional leaves until these contain either only one identity or the "*unknown identity*" label).
The detected face is propagated into the leaf nodes of the detection RT, the corresponding class histograms are extended with the additional classes and additional training iterations are applied to discriminate among the identities and the large set of unknown identities. This step is repeated for each RT and a final identity label is obtained using majority voting.
New Identities. This training step only needs few iterations, thus this method allows for adding new identities on the fly in the same way as initially the detection RT is extended to become a joint detection and recognition RT. Similarly, new identities can be added to a detection/recognition tree: A new set of face images are supplied for the new identity. These images are propagated to the leaf nodes of the RT and additional nodes are added until the new identity is separated from other identities.
Another option to achieve detection and recognition is to first create a detection RF and then in a second step to create a recognition RF. A similar setup was presented for the task of hand gesture and object recognition in (Deselaers et al., 2007). This setup is also open to the learning of new identities by rebuilding the recognition forest.

4.2.2.5 Parameters

We provide insights into the parameters modelled and the values experimentally determined to work best for our setup.The *forest size* T is theoretically and empirically related to the classification accuracy. We grow *ten* RTs. The *feature size* limits the dimensions of the rectangles sampled and we find $0.5 \cdot \sqrt{K \times L}$, where $K \times L$ is the dimension in pixels of the training samples, as ideal value. We set $L = 200$ for our evaluations. Note that it has been shown, with as few as 3 or 5 RTs, acceptable performance was achieved (Deselaers et al., 2007).

4.2.3 Experimental Evaluation

We present the results from the experimental evaluation of our proposed methods to face detection, recognition and learning and discuss the integration into our domestic service robot demonstrator (Schiffer et al., 2009).

4.2.3.1 Detection

We compare our detection performance to the AdaBoost face detector (Jones and Viola, 2003) of the OpenCV library. For this purpose, we use the pre-trained model delivered

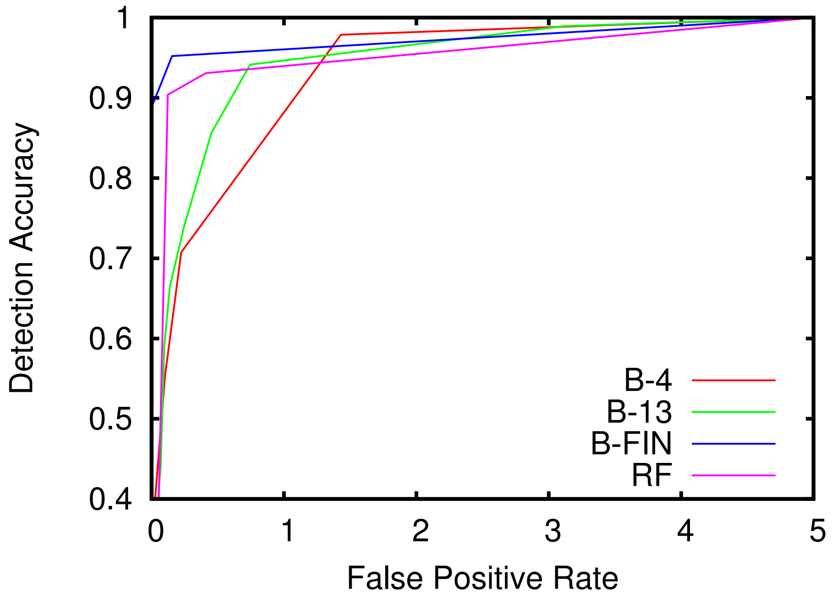


Figure 4.5: ROC curve for the detection results.

with the library and we also compare using a model that we trained using our training data. It is well known that the pre-trained model was very carefully engineered and performs extremely well. Unfortunately, it is unclear which data was used for training and thus comparison with this model is not completely fair.

Our training collection includes a collection of 4000 faces and 4000 background images collected from various sources on the Internet, all scaled to 24×24 pixels. We trained the AdaBoost detector for four and 13 days, denoted as *B-4* and *B-13*, respectively. Training was performed on an Opteron machine with 2.2 GHz. OpenCV ships with a pre-trained detector which we refer to as *B-FIN* in the following.

The RF is trained on the same data, where we allow for up to 8000 nodes (which allows perfect discrimination between all training samples). Here, the training takes approximately 400 s per RT on a 2.0 GHz Intel Core2Duo machine.

ROC curves for the detection results are shown in Figure 4.5 for a combination of the MIT+CMU test collection (Rowley et al., 1998b), Yale Face Database[2] and the BioID Face Database[3]. In the figure, both *B-4* and *B-13* attain a detection accuracy of 90 % with a very high false positive rate, emphasizing the need for weeks of training time for AdaBoost models (Pham and Cham, 2007). RFs outperform *B-4* and *B-13*. Further, RFs perform comparatively to *B-FIN* albeit with a slightly higher false positive rate. We however note that our detections suffer from alignment errors. One option to tune our method would be to incorporate a more carefully engineered filtering/merging step similar to the one proposed by Jones and Viola (2003).

4.2.3.2 Recognition

The recognition is evaluated after detection on the BioID and Yale collections. We cannot use the MIT+CMU test set since identities are not annotated there. We compare

[2]`http://cvc.yale.edu/projects/yalefaces/yalefaces.html`
[3]`http://www.bioid.com/downloads/facedb/`

Table 4.4: Results (error rates) from the recognition.

Test Collection	RF	SVM
Yale[RF]	46.6	69.4
BioID[RF]	86.3	68.63
Yale[B-FIN]	15	7.7
BioID[B-FIN]	13.5	2.7

the performance of our RF classifier to an SVM classifier. SVMs (Hearst et al., 1998) can be used as general classifiers in pattern recognition tasks. The SVM performs a linear classification by finding a hyperplane in the feature space that divides a two class problem with maximum generalization (Boser et al., 1992). This hyperplane is called optimal when it separates without any error and the distance between the data points of the two classes and the hyperplane is maximal (Guo et al., 2000). In face recognition with SVMs the pixel of an image are usually considered directly as a feature vector. For an in-depth treatment of SVMs we refer to (Vapnik, 1995) and (Cortes and Vapnik, 1995). Results are given in Table 4.4. Due to the alignment errors of our detection framework, the error rates in Yale[RF] and BioID[RF] are rather high. If, however, we use the pre-trained OpenCV face detector (*B-FIN*), a drastic performance improvement can be observed in the corresponding Yale[B-FIN] and BioID[B-FIN]. In general, it must be noted that the SVM outperforms the RFs but at the cost of a) a much higher training time, and b) a much higher time required for the classification.
To further analyse the difference in the classification speed, we trained RTs for recognition only and detail the training time. It takes about 7.5 ms to create an RF consisting of ten RTs, each grown up to a 1000 nodes with $L = 50$ on our BioID training collection of 464 face images and six identities. As our Yale training collection is smaller, a total of 40 face images and five identities, we are able to build ten RTs in only 2 ms by letting each RT grow up to 200 nodes and $L = 50$. Clearly, if the number of training images per identity can be approximately estimated in advance, we can optimize the corresponding node count ($2 \cdot$ number of identities $\cdot$ images per identity) and grow an RF extremely rapidly with rank-1 recognition[4] rates comparable to SVM.

4.2.3.3 Realistic Scenario

The method was developed for mobile service robots, we also evaluate it on our in-house lab data set. Images with faces in typical backgrounds that a mobile robot encounters were captured consisting of four identities and up to 30 images per per identity. Sample images are depicted in Figure 4.6. Faces were collected using *B-FIN*. We use 22 images for training and eight to evaluate the performance. We train an RF of ten RTs with up to 200 nodes, training takes 2.5 ms and the error rate is 7.1 %. Table 4.5 gives insights into the duration of RT growth. Here, the number of nodes, test candidates and training data that contribute to the training time are enumerated on the experiments presented in this section. The rank-1 recognition rates correspond to the error rates discussed in the table above and that on our lab test.

[4]The rank-N recognition is the percentage that the correct label of the input image is within the top N suggestions (Jones and Viola, 2003). The rank-1 recognition hence is the result returned as is by the random forests using majority voting.

Table 4.5: Relations between L,N, number of nodes, the training time, and the recognition rate (RR).

Data set	Nodes	L	N	Time	RR [%]
Detection	8000	200	8000	400s	–
BioID	1000	50	464	0.75ms	86.50
Lab	200	50	88	0.25ms	92.86
Yale	200	50	40	0.2ms	85.00

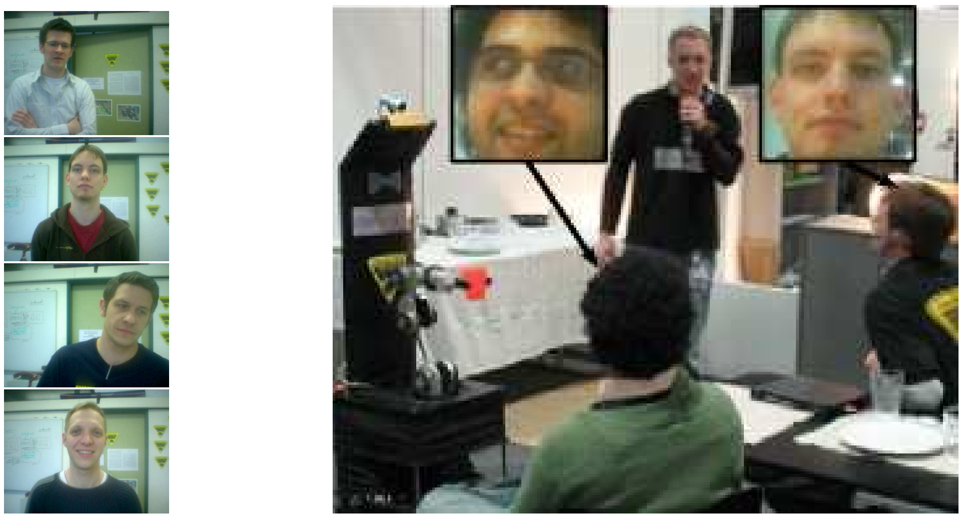


Figure 4.6: Left: examples from the lab test set. Right: detection, segmentation and recognition of faces at the 2008 RoboCup German Open, Hannover, Germany.

The described system was deployed and used on the ALLEMANIACS' service robot CAESAR at the 2008 RoboCup German Open competition[5] and at the 2008 RoboCup World Championship[6] in the RoboCup@Home league. The task included detecting, recognizing and learning faces in the arena (Figure 4.6) to hand over objects of interest to the respective personnel.

4.2.4 Summary

We presented a framework for one-step face detection and recognition with low training time, which is able to learn new identities at any moment on our mobile robotic platform. The problem of one-step detection and recognition is addressed using decision trees which can be trained to discriminate additional classes with only little effort and it is shown that the technique works well.
The proposed method is evaluated on different data sets and compared to the standard AdaBoost detection method and an SVM-based recognition system and it is shown that the new method is faster by several orders of magnitude in the training and testing time with only little deterioration of the accuracy. We plan to improve our detection aggregation technique in a similar way as the AdaBoost cascade does.

[5]http://www.robocup-german-open.de/
[6]http://www.robocup-cn.org/

4.3 Gesture Recognition

So far, we have considered speech as a means for intuitive control and interaction with a domestic service robot. However, a huge part of meaning in communication is also transferred via non-verbal signals (Engleberg and Wynn, 2006). A very important mode of this non-verbal communication is using gestures. This is especially true in interaction with a domestic service robot, since controlling the robot often relates to entities in the world such as objects and places or directions. References to objects can conveniently be made by pointing gestures while other dynamic gestures can be used to indicate directions or commands.

Quite some approaches tend to pose undesirable requirements both, before the start of operation and within the operation itself. For example, they may require a tedious calibration of hand colors or depend on initialization poses of the human. Also, some approaches are quite costly and demand high computational resources. We try to avoid these undesirable properties and aim for a flexible, modular, and robust system that is both, easy to set up and easy to use while minimizing computational demands.

4.3.1 Overview and Related Work

We designed a modular architecture where gesture recognition is decomposed into subtasks orchestrated in a multi-step system. This enables a *filter-and-refine* like processing of the input where the single steps are as independent as possible to allow for an easy replacement of the particular method used for any of them. Further, the output of each step can already be used for specific purposes. That is, for example, pointing gestures can already be inferred from the position (and possibly the posture) of a hand and a face, while recognition of dynamic gestures additionally requires hand tracking to extract a trajectory. What is more, the overall computational demands are kept low because the amount of information to be processed gets reduced in each step.

4.3.1.1 A Multi-step Approach

The process of gesture recognition is subdivided into four main steps: *hand detection*, *posture recognition*, *hand tracking*, and finally *gesture recognition*. Hand detection is the task of finding the position of one or more hands in an image, where we follow a color-based approach. To increase robustness against false detections, we additionally apply a *hand verification* step. Posture recognition then is to determine the shape of the hand, that is to say the configuration of the fingers (e.g., a *fist* or an *open hand*) and the orientation of this *posture*. Both, hand verification and posture recognition are performed with a feature-based classification method. Binary classification is used to decide whether a candidate area actually contains a hand and a multi-class classification determines one of multiple possible postures. Hand tracking refers to recording the position of the hand (and its posture) over a sequence of images. Finally, gesture recognition is understood as identifying a specific dynamic movement of the hand from the trajectory formed over time. A graphical overview of our system's architecture is depicted in Figure 4.7.

Note that the intermediate steps yield useful information already. The hand detection and verification provide us with hand positions and posture recognition adds the posture

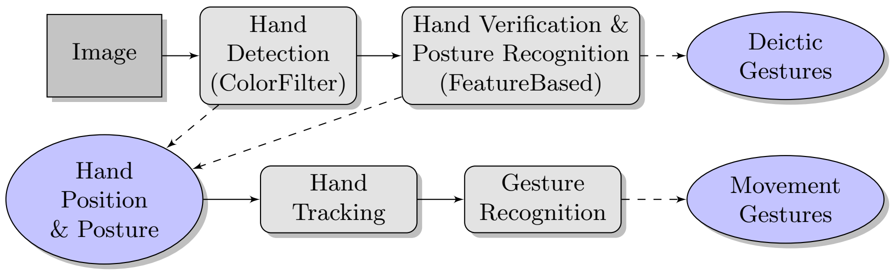


Figure 4.7: Architectural overview of our approach

of a hand detected. Specific commands such as a lifted open hand can already be processed, for example, as an indication of a *stop*-request or the user's request for attention. Taking an existing face detector as a secondary cue also allows for identifying pointing gestures when the posture is, say, a lifted index finger. We refer to gestures at this stage as *deictic gestures*. Also, by iteratively refining the information we can dismiss wrongly detected hands and attach additional information thus enabling increased performance of later steps.

4.3.1.2 Related Work

Hand gesture recognition can be used for various human-machine interaction tasks (Wachs et al., 2011). It is well-suited in human-robot interaction with a domestic service robot because it is a natural means that humans use in their communication to convey information. For instance, Van den Bergh et al. (2011) propose a system for real-time recognition of 3D pointing gestures to specify navigation goals for a robot. We now briefly review some related work for each component of our approach. Further, we also indicate overall approaches throughout this section.

Hand Detection A lot of the existing hand detection approaches are color-based, since skin-colored regions can quite easily be detected in an image (Saxe and Foulds, 1996; Jones and Rehg, 2002). There are two main issues with (such static) color-based hand detection, however. For one, not every skin-colored region in an image is in fact a hand. For another, in dynamic environments the values of skin color may vary over time, for example, because of changes in lighting conditions or due to obstructions like shadows or colored reflections. A combined way to tackle the problems mentioned above is to make use of the assumption that an image containing a hand also contains a face. This does not seem to be too restrictive given a human-robot interaction scenario. If a face is detected, a skin color can be extracted from that face and then be used for hand detection. Such an approach was already presented in (Francke et al., 2007) and (Correa et al., 2010). We follow similar directions but apply some variations. We operate on the HSV color space which, according to (Zhu et al., 2000), seems most appropriate for skin color detection. We then use a modified scheme to dynamically adapt the color values over time. Another way to detect hands is to use features instead of color. Kölsch and Turk (2004b) use a sliding window and apply a classifier on this window to decide on whether this window contains an object of interest (i.e. a hand) or not. It is an extension of a method for object detection presented in (Viola and Jones, 2001), namely a haarcascade, which is a cascade of haar-like visual features. The classifier is built using the AdaBoost method (Freund and Schapire, 1997) for boosting. An alternative to classical boosting could be random forests (Ho, 1995), especially for multi-class classification as we will employ for posture recognition. As described in the previous section, we have successfully applied random forests to face recognition for multiple identities on our mobile robot (Belle et al., 2008) already as shown in the previous section.

Posture Recognition A method to recognize hand postures is followed in (Panin et al., 2009). The idea is to bring the fingers of a hand in a meaningful relation by considering the hand's convex hull and counting the defects of this hull. However, this is only done for planar hands and it is not generally applicable in its current form. Triesch and von der Malsburg (1996) present a system for posture recognition using elastic graph matching. Several points on a hand are used to build a graph which is compared to graphs trained on a set of postures before. Wang and Wang (2008) use SIFT features to discriminate different postures. Although these methods seem appealing and could potentially be plugged into our modular architecture, we opt for a feature-based posture recognition following Kölsch and Turk (2004b) to be able to exploit the similarities in hand verification and posture recognition.

Pointing Gestures Nickel and Stiefelhagen (2007) use the input of a stereo-camera to train a hidden Markov model to recognize pointing gestures. To achieve joint attention in interaction with a service robot Droeschel et al. (2011) recognize pointing gestures with a time-of-flight camera. Van den Bergh et al. (2011) use a Kinect sensor providing depth information. Then Haarlets are used for the gesture recognition. With our modular architecture we are able to recognize pointing gestures with only very little effort, as long as the sensor provides 3D information of the hand and the face. Even with only 2D information we could estimate the pointing by the relative sizes of the face and the hand.

Hand Tracking Tracking can, for example, be realized with a method called "Flocks of Features" (Kölsch and Turk, 2004a). To track, the hand position is initialized by detecting one of a set of six trained postures. Then, features are chosen on this hand and followed with KLT feature tracking (Shi and Tomasi, 1994). Another successful tracking mechanism is CAMSHIFT (Bradski, 1998), a modification of the *mean shift* algorithm. The position of a fixed size window is iteratively computed. However, CAMSHIFT is color-based and it is thus subject to the drawbacks we mentioned earlier. That is why we follow the lines of Shi and Tomasi (1994) in tracking.

Gesture Recognition There is a huge body of work on gesture recognition, a survey can be found in (Mitra and Acharya, 2007), for example. A lot of the approaches use statistical methods like hidden Markov models such as (Axenbeck et al., 2008) or (Park and Lee, 2011). We think HMMs are computationally too demanding for our setting. Also, statistical methods usually rely on (often huge amounts of) training data which the recognition then depends on. Instead, we plan on extending a simple approach to gesture recognition described in (Wobbrock et al., 2007). It is especially intriguing since it does not require any form of training and does not depend on any external library or toolkit either.

4.3.2 Hand Detection and Posture Recognition

We now detail the first steps of our approach, that is, hand detection, hand verification, and posture recognition. First, we employ a color-based approach for hand detection. A so-called *ColorFilter* selects those parts of the image that are skin-colored. The extracted parts are then forwarded to a *hand verification* step which decides whether these areas actually contain a hand or not. All detected and verified hands are subsequently processed by a multi-class classifier determining one of a set of specific postures for each hand. The result of this procedure, that is, the hands in an image and their (possibly undefined) postures are then available for any later module. This multi-step processing tries to reduce the search space with every step. The cheapest, color-based classification is done first. The false positive rate at this stage may be comparatively high, though. However, the subsequent step filters out further false positives, now with a feature-based method which is computationally more costly, but only has to consider fewer candidates. Only those hands are processed in the posture classification that passed the first two steps. This way, we proceed with less effort than it would have taken to do the classification of the different postures on the complete image instead of on candidate regions only.

4.3.2.1 Hand Detection

The maybe most obvious feature of hands in an image is color. Skin color may easily be detected (Saxe and Foulds, 1996; Jones and Rehg, 2002), but especially static color models suffer from dynamic changes in the environment such as lighting, shadows, and inter-person skin color variations. Further, static color models typically require a training which has as many examples for skin color as possible. Any hand exhibiting a color that is not covered by the training set will not be detected by this color model. That is why we opt for a dynamically adapting skin color model with almost no prior information. Instead, we only exploit the following assumption: If the robot is to interact with a person, this person is most likely facing the robot. So, similarly to Francke et al. (2007), we first detect faces in the image to extract parameters for our color model from the face region. For one, the face detection module is active on our robot already, so no additional effort is required. For another, there are properly working face detection methods (Jones and Viola, 2003; Viola and Jones, 2004) readily available in OpenCV.[7] After a sufficiently reliable detection of a face in x subsequent images we compute expectation values σ_c and standard deviation μ_c for each of the HSV color space channels c, i.e., hue, saturation, and value. Pixels in the image with a value of v_c in color channel c are then classified according to the formula $\mid \sigma_c - v_c \mid \leq \alpha \cdot \mu_c$, where α is a positive real-valued factor allowing us to control the adaptivity.
Additionally, we compute two control values for every image that help in determining when and how we need to adapt our model. The *image cover percentage* indicates how many percent of the image are classified to be skin-colored. Analogously, the *face cover percentage* tells us how many pixel of the face region were classified to be of skin color.

[7] http://www.opencv.org/

Table 4.6: Results of Color Evaluation

Cover Percentage[2]		True positive	False positive	Mean iterations
Face	Image			
0.3	0.6	0.83	0.19	0.71
...				
0.5	0.8	0.90	0.19	1.03
...				
0.6	0.8	0.93	0.24	3.42
0.7	0.5	0.96	0.33	11.25
...				
0.9	0.9	0.99	0.44	54.10

If the *face cover percentage* is too low, this indicates that only few pixels were considered to be skin, which obviously is undesirable. In that case we have to increase α to make our skin color classifier more lenient. If, on the other hand, the *image cover percentage* is too high, too much of the overall image is considered to be skin. In that case we need to lower our α for the model to be more restrictive.

4.3.2.2 Evaluation

To determine appropriate values for the *image cover percentage* and *face cover percentage* we constructed a database of around 800 pictures of typical application scenario settings. To do so, we placed faces and hands with variations in lighting and brightness on indoor background images. Faces and corresponding hands were varied in size to mimic a user standing in front of the robot at distances of between two and three meters. For every combination of *image cover percentage* between $[0.5 \ldots 0.9]$ and *face cover percentage* between $[0.3 \ldots 0.9]$ in steps of 0.05 we computed the true positive and false negative rate as well as the number of iterations it took the color model to reach a fixed value as *mean iterations*. Selected results are listed in Table 4.6.[8]
The final ColorFilter then used a threshold of 0.5 for the *face cover percentage* and 0.8 for the *image cover percentage*. We chose to do so because at these values the true positive rate of 90% was reasonably high while the false positive rate of 19% was tolerable, especially since further false detections can still be sorted out in later steps. A mean number of 1.03 iterations also indicates that these values are quite stable and do not require too many adaptations. Combinations beneath the horizontal line after the values 0.6 and 0.8 have a very high number of iterations so that it would take too long to compute color models with these.

4.3.2.3 Hand Verification and Posture Recognition

Since the previous step, the color-based hand detection, results in a non-negligible amount of false positives, the next step is to verify the candidate regions. Also, we want to determine the posture of every hand that could be verified. For both these tasks we choose a feature-based approach. A basic technique in feature-based detection is to

[8]Percentages are given as values between 0.0 for 0% and 1.0 for 100%, respectively.

Table 4.7: Results for *Haarcascade* training in overview

	Training Images		True Positive	False Detections	Stages	Training Time
	Positive	Negative				
All Hands	1000	700	0.15	71.95	11	21h
All Upright	1000	1000	0.36	3.06	18	42h
Open Own	3000	2000	0.98	12.71	12	7h
Open Kumar	1000	700	0.93	2.37	14	9h
Fist	130	100	0.57	0.23	4	1h
Fist + Open	500 + 500	1000	0.28	0	19	41h
L	100	100	0.25	0	10	3h

use a so-called *sliding window*. A window of fixed size is slided over the image and for each position of the window a decision rule (i.e. a classifier) decides whether the window contains an object of interest or not. Since the classifier used with the sliding window can be a binary or a multi-class classifier and the window does not need to be slided over the whole image but can also be only slided over regions of interest (ROIs) we opt to use feature-based classification (with haar-like features) for both, the hand verification and the posture recognition.

4.3.2.4 Data Sets

A major question in building feature-based classifiers in general, and for binary classifiers for hand verification in particular, is the selection of an appropriate training set. In contrast to faces, the variations found with many different hand postures are considerably larger. To determine a good classifier for hand verification we trained and evaluated several haarcascades (using OpenCV) and random forests with different training sets compiled from collections available on the Internet, namely one set with high resolution images of a single posture,[9] a set with lower resolution images of ten different postures[10] and a collection by Ajay Kumar[11] used as training data for a touchless palmprint authentication (Kumar, 2008). Additionally, we constructed our own data set from hand images extracted from video sequences and placed on complex backgrounds. The video sequences were recorded by gesturing in front of our robot. Negative examples were taken from a data set collected by Natoshi Seo.[12]

4.3.2.5 Evaluation

We conducted extensive evaluation series with both, classical haarcascades and random forests. Results from different trainings for haarcascades and random forests are given in Table 4.7 and Table 4.8, respectively. The evaluation suggests that training does not seem suitable to generate a single cascade capable of detecting arbitrary hands with any of the available training sets. An in-depth investigation revealed that the choice of haar-like features available to the classifier influences the recognition results.

[9] http://www2.imm.dtu.dk/~aam/datasets/datasets.html
[10] http://www.idiap.ch/resources/gestures/
[11] http://www.comp.polyu.edu.hk/~csajaykr/IITD/Database_Palm.htm
[12] http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/

Table 4.8: Results for *Random Forest* classifiers

	TP rate	FP rate	Samples	Trees	Time (h:m)
All hands, old feat.	0.73	0.25	2×190	10	0:36
All hands, 4 orient.,	0.79	0.25	5×200	3	0:20
5 classes	0.63	0.03	5×2000	20	16:02
Open	0.73	0.07	2×250	5	0:09
Fist	0.78	0.07	2×250	7	0:21
Open + Fist	0.77	0.21	2×250	9	0:12
Closed	0.60	0.12	2×1000	2	1:18
Victory	0.61	0.16	2×180	1	0:06
Classify Postures	0.19	-	5×150	5	0:31

This finding is supported by results from (Kölsch and Turk, 2004b), where instead of traditional haar-like features the authors use so-called "*four box*" features that are more fine-grained. Instead of simply subtracting grey-values from different regions these allow for almost arbitrarily shaped configurations. We tried to achieve this flexibility by adding new features to our classifiers but still could not replicate the performance of a true positive rate of $92,23\%$ reported in (Kölsch and Turk, 2004b). Despite the unsatisfactory results for both, haarcascades and random forests, it is worth noting, that random forests have considerably lower training times while providing better results. Moreover, it is possible to extend our current implementation of random forests to use the "*four box*" features to improve on our results. Until we finish to do so, we make use of a haarcascade created by Daniel Baggio.[13] Evaluation on our test data yielded a true positive rate of 74% and a false negative rate of $9.1822 * 10^{-8}$, which corresponds to one false detection every six frames. Hence, in the current system it may happen, that the user has to repeat a gesture as a result of false detections.
Besides classifying between "hand" and "no hand" only, an additional dimension is to differentiate between several *hand postures*. As Table 4.7 and Table 4.8 indicate, it is possible to train haarcascades as well as random forests to recognize specific single postures with true positive rates of up to 98% for haarcascades and at least 60% for random forests. Training times for random forests were considerably lower than for haarcascades. However, we did not succeed in building a single random forest classifier that was able to tell apart five different postures with a sufficiently high true positive rate. We hope to eventually achieve more accurate posture classification once we fully integrate the "*four box*" features used in the classifier from Kölsch and Turk (2004b) into our random forest classifiers. Until then, we restrict ourselves to a few postures that we could train sufficiently accurate single classifiers for. Those classifiers can then be applied independently in parallel.

4.3.2.6 Deictic Gestures

As already mentioned, we can make use of the intermediate results after the hand detection and posture recognition steps already. These static gestures are referred to as *deictic gestures*. We trigger detection either when a specific posture (such as a pointing)

[13] `http://code.google.com/p/ehci/wiki/HandTracking`

is recognized or by means of an external trigger such as keywords like “*stop*” or “*there*” spotted by the speech recognition module running on the robot. As an additional cue to extract information from for pointing gestures we make use of the face detection module we used before already. We extract the face’s position and compute a vector from the center of the face position to the center of the hand’s position to identify a pointing target.

4.3.3 Tracking and Gesture Recognition

After the position of a hand has been determined, its trajectory has to be tracked over time to enable gesture recognition. *Tracking* and *gesture recognition* are thus closely connected.

4.3.3.1 Hand Tracking

The general idea with tracking is that a hand’s position will not change too rapidly from one image frame to the next. A naive method to track a hand is to detect all hands in the image and associate the closest hand in the subsequent frame. However, this presupposes a very reliable hand detection. A different technique is applied in (Kölsch and Turk, 2004b), where hands with the same posture are tracked over time. The underlying assumption that the posture of the hand stays the same throughout a gesture does not seem to allow for too natural gestures, though.
In our system, we use a tracking scheme called “Flocks of Features” as proposed in (Kölsch and Turk, 2004a). Components to realize the tracking are readily available in OpenCV. After detecting a hand, features are computed and then tracked using KLT feature tracking (Shi and Tomasi, 1994). In a test series of 20 “tries-to-escape” where the user erratically moved around the hand within the camera’s field of view, the selected tracking scheme proved sufficiently robust. Hands were tracked over periods of between 1.5 and 34.98 seconds with an average of 18.48 seconds. This is sufficient for most gestures in a domestic setting.

4.3.3.2 Gesture Recognition

The final step in our system is to recognize a gesture from following the hand’s position over time. First, the trajectory of the hand is recorded. Then, this trajectory is compared to a set of known gestures. Since we want to keep our system as free from training and as computationally inexpensive as possible, we opt to adapt a method introduced by Wobbrock et al. (2007) initially designed for one-stroke hand-written gestures. It is fast to implement and still yields reliable results. The basic principle is to norm trajectories to a certain scheme and then to compare performed gestures to a set of known gestures by computing their individual points’ distances.
The preparation of trajectories before the comparison to known gesture templates is as follows. Since pairwise comparing all points is too costly only points with the same index are compared. For this to work, the trajectories have to contain the same number of points, which is why every trajectory is filled to have equidistant points. Here we chose 128 as the number of points following results given in (Wobbrock et al., 2007). Afterwards, the trajectory is rotated to a standard orientation, i.e., an orientation of $0°$ between the first point and the centroid, to achieve rotational invariance. Then, the

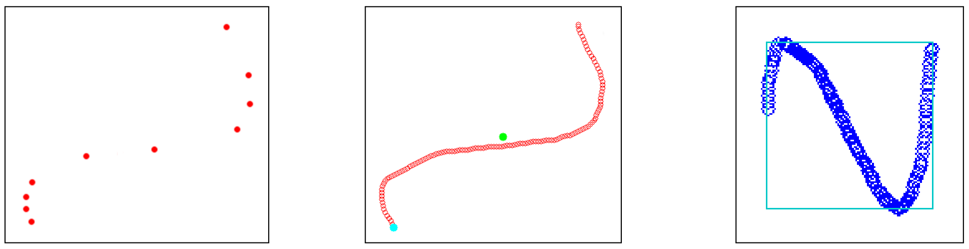


Figure 4.8: Processing steps on a *Snake* trajectory for gesture recognition (Wobbrock et al., 2007).

resulting point set is scaled to fit a box of 100×100 pixels. Preparation steps are shown in Figure 4.8
Basic movements like "left to right" correspond to lines in a trajectory. Unfortunately, the detection of straight horizontal and vertical lines is problematic with the method from (Wobbrock et al., 2007). This is why we need to slightly modify the approach by applying a different scaling mechanism. If the ratio between width and height of the unscaled trajectory is below a certain threshold (empirically determined to be 0.3) we only scale the larger side of the box. This prevents the scaled line-trajectories from being too scattered to be recognizable. We adjust the gesture templates to compare with accordingly.

4.3.3.3 Evaluation

We conducted a separate mouse-based evaluation of the gesture recognition component to yield results independent from the hand detection itself. We recorded a total set of 314 gestures (one of *Line*, *Wave*, *Square*, *Circle*, and *Triangle*) where 85.67% were recognized correctly. More than 50% of the false detections were confusions between *Square* and *Circle* gestures. One reason could be imperfections in the mouse-based input. Preliminary findings in the analysis of human gestures (Grandhi et al., 2010) suggest that humans perform gestures surprisingly precise. If we then assume the trajectories from human gestures to be more precise than the ones created artificially, these confusions might drop down already. Moreover, we additionally applied a corner detection on the trajectory to clear up these confusions. With this extension in effect we could raise the detection rate to above 92%.

4.3.4 Summary

We presented a modular system for visual gesture recognition for interaction with a domestic service robot. The processing pipeline is organized in such a way that it reduces the amount of information to process at every step. We start with an inexpensive on-line adapting color-based method for hand detection, leaving only parts of the image to be processed by the more expensive feature-based posture recognition. The actual gesture recognition adopts a fast approach originally designed for hand-written gesture recognition that does not require any training. Results from intermediate processing steps can already be used for interaction, for example, to react on deictic pointing or stop gestures.

Although our system suffers from shortcomings in the hand verification and posture recognition steps yet, the overall performance is sufficient for it to be applicable in a domestic setting with some minor restrictions. While we successfully used the system at a major robotics competition already, each individual step can still be improved. These improvements, however, can be realized relatively easily due to the modular design. Likewise, anyone replicating the system can exchange any of the proposed components to his or her liking. An important next step is to fully integrate the "*four box*" features from (Kölsch and Turk, 2004b) in our random forest implementation.

4.4 Other Interaction Modalities

So far, we presented components for robust speech recognition, for face detection, recognition and learning, and for gesture recognition. However, those three modules are only a part of the set of interactive capabilities of our domestic service robot. A complementary capability for the speech recognition is speech synthesis. Also, the reverse of using face detection to discover a human could be to display a virtual face as the robot's appearance. Other forms of interaction might be enabled using a touch screen to display information and receive touch-based input. We report on those additional modalities in the following.

4.4.1 Speech Synthesis

During the interaction with a human user the robot may have to give instructions, ask for information or provide data it is being asked for. A convenient mode to do this, especially when speech input has been used towards the robot, is to generate spoken output. We do this using the freely available Festival[14] speech synthesis system (Black and Taylor, 1997; Taylor et al., 1998; Black et al., 2001) or, as an alternative, Flite[15] (Black and Lenzo, 2001). We pass the text that should be spoken to either text-to-speech software which then renders the corresponding audio output. To improve on the naturalness of the speech synthesis output we equipped Caesar with a commercially available voice named Lawrence from Cepstral LLC.[16]

4.4.2 People Detection and Tracking

We already mentioned that face detection is available on our robot. It can be used as a means for detecting a person. However, due to occlusions or simply because a person is not facing the robot this might not always be feasible to detect humans. Therefore, we use additional methods to do so. One way to detect people is by registering their body parts. In our case, we use circle detection in the data available from the laser range finder to identify leg-like structures. Another option is to use sound localization to locate sources that could potentially be humans. By combining those modalities as investigated earlier (Calmes et al., 2007b) human presence can be verified. The additional modalities can be used to track a human over time for periods where the face is not visible to the

[14] `http://www.cstr.ed.ac.uk/projects/festival/`
[15] `http://www.cmuflite.org/`
[16] `http://www.cepstral.com/`

robot. A a tracking method, for instance, the Hungarian method (Kuhn, 1955) can be used.

4.4.3 Interactive Touch Display

To yield affective interaction our robot is equipped with a touch screen monitor that can display things like instructions to the user and which can also be used to receive user input such as pressing a button to select one of multiple options.

Often, the information displayed on screen is mainly provided to backup the speech synthesis in noisy conditions and to provide feedback about the robot's perception. For example, the last utterance recorded by the speech recognition subsystem and the last utterance given to the speech synthesis are printed on the screen. For monitoring purposes the robot also reports on some internal parameters ranging from low level data like the current voltage of its batteries to higher level information such as the skill that is currently being executed.

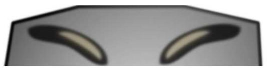
4.4.3.1 Face Display

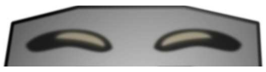
It has been shown that interaction with computers (a robot is considered an embodied computer here) changes when it has a human-like appearances such as a face (Sproull et al., 1996). They attribute personality traits to the machine and are more aroused. This facilitates more natural interaction. Also, people are more likely to engage in interaction with a robot that can convey expressions with a face and that can indicate its direction of attention (Bruce et al., 2002). That is why we endowed our robot with a virtual face that is displayed on the interactive touch screen.

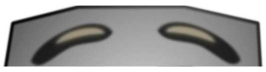
The face is composed of four slices, one for the brows part, one for the eyes, one for the nose and the cheeks, and one for the mouth. Every slice is available in different configurations, e.g., the eye brows may be normal, lifted, or frowning, and the eyes may be looking straight, left, right, up, or down. The whole face is then assembled by taken one image for each of the four slices. The modular composition yields a very flexible display allowing for a total of 225 different expressions (3 configurations of the brows $\times$ 5 configurations of the eyes $\times$ 3 different cheek styles $\times$ 5 configurations of the mouth), however, not all of them make sense. Still, our virtual face can express the six basic emotions (anger, happiness, fear, surprise, disgust and sadness) that according to Ekman and Friesen (1971) are recognizable across different cultures. The display can also support what the robot is currently doing, for instance, it can show an open mouth while talking or it can look to the left when it is currently performing a manipulation on its left side. The different slices of the facial display are shown in Figure 4.9.

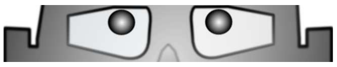
4.4.3.2 Instruction Panel

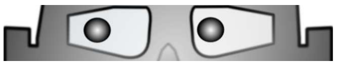
Sometimes the user needs to be instructed on what to do with certain tasks. For example, as part of a calibration procedure the human user may be requested to take a certain pose for the robot to analyse. Figure 4.10 shows a selection of instructions displayed on the screen; some of them are used in the demo application presented in the next section.

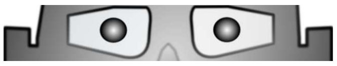

(a) brows frowning

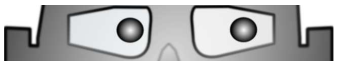

(b) brows default

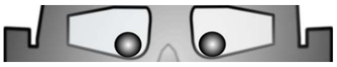

(c) brows lifted

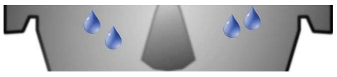

(d) eyes up

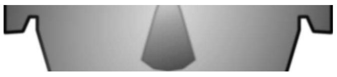

(e) eyes left

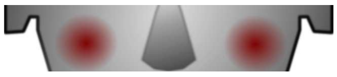

(f) eyes default

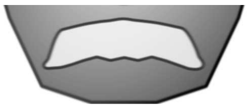

(g) eyes right

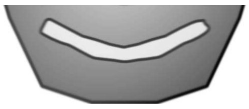

(h) eyes down

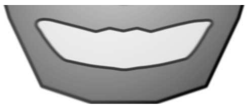

(i) jowl tears

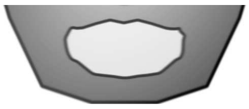

(j) jowl default


(k) jowl blush


(l) mouth scowling


(m) mouth closed


(n) mouth default


(o) mouth open

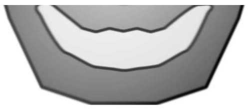
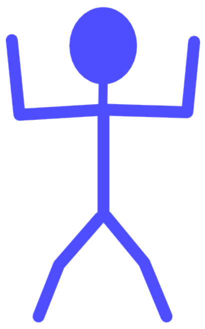
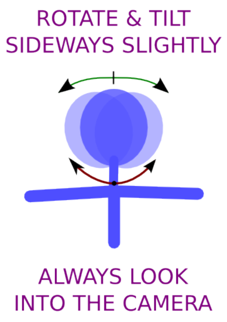
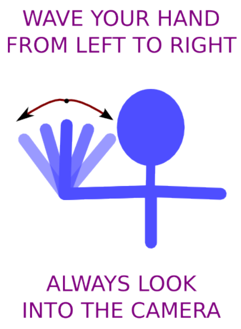

(p) mouth smiling

Figure 4.9: Virtual facial display used in interaction


(a) body calibration



(b) head movement



(c) gesture activation

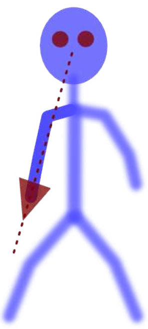

(d) pointing explained

Figure 4.10: Some of the figures displayed on the robot’s instruction panel

4.4.4 Summary

Besides the three modes of speech recognition, face detection, recognition and learning, and gesture recognition, we presented additional modules that extend our robot with means to generate spoken output, to display information and to show a virtual face that can express basic emotions. All these components extend our mobile platform towards being a social robot that can successfully perform human-robot interaction.

4.5 The Interactive Robot in Action

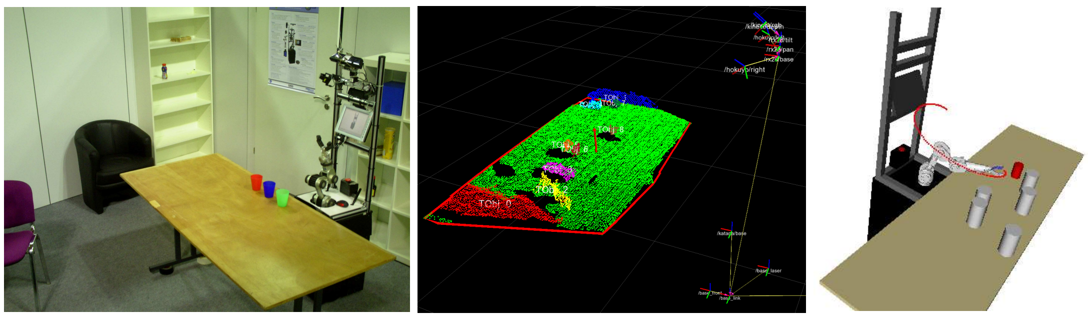
To illustrate the newly added capabilities we describe an interactive demonstration performed by our domestic service robot CAESAR (Schiffer et al., 2012a). In a home-like environment CAESAR's task is to help setting the table. Our robot CAESAR, presented in detail in Section 2.3, meets all the basic requirements put on an autonomous mobile robot, that is, it can navigate in its environment safely (cf. Section 2.4) and it can localize itself reliably with high accuracy in known environments. With the new capabilities described in the earlier sections of this chapter, it is able to interact with people around it, it can further detect and recognize and objects and it can also manipulate objects with its robotic arm. Of particular importance for the demo discussed in this section are the three modules we presented in this chapter that extend the robot with capabilities for human-robot interaction. Namely, these are its robust speech recognition, the face detection and the gesture recognition Those components are orchestrated in the Fawkes robot framework (Niemueller et al., 2010a). Thereby, we illustrate the interplay of several modules in carrying out complex tasks. The overall system allows to perform robust reliable service robotics in domestic settings like in the ROBOCUP@HOME league. Also, we show how our high-level programming language provides a powerful framework for agent behaviour specification that can be beneficially deployed for service robotic applications.

The challenge in domestic service robotics scenarios which are replicated in benchmarks like the ROBOCUP@HOME league is to build such a robust robotic system that can safely operate in a human environment and that can interact with humans. As the complexity of the tasks to solve in domestic settings rises, so increases the benefit a robot has from using sophisticated means for decision-making and deliberation. The high-level control of CAESAR is based on the language READYLOG (Ferrein and Lakemeyer, 2008) which we introduced in Section 3.4. A particular feature that we use in the following is decision-theoretic planning which allows to generate optimal plans for complex tasks.

We showcase a special helping task that CAESAR is able to perform: the *Robotic Order Cups Demo (robOCD)*.[17] The robot's task in this demo is to help decorating a table. In the scenario there are three differently colored cups (red, green, and blue) on a table. To complete the re-arranging they have to be put in a specific order. A human user is instructing the robot on the desired order by pointing to positions on the table and by simultaneously specifying which cup should be placed at that very position using speech. Figure 4.11 shows the robot in front of the table with three cups on it. A user stands on the other side of the table and specifies the desired order of the cups by pointing.

Algorithm 2 shows the READYLOG procedures used in the demo. The procedure *main* is

[17]A video of the demonstration is available at `http://goo.gl/7rEEY`.



(a) CAESAR in front of the table (b) 3D scene perception (c) grasp planning

Figure 4.11: The robotic order cups demo

Algorithm 2: READYLOG program for the Order Cups Demonstration The terms p_1 to p_4 denote four positions on the table, while I_i and P_i are variables that hold the color of a cup at position i in the initial and the goal situation, respectively. $pos(C)$ returns the position of the cup with color C.

```
proc $main$,
  $get\_Initial\_Order(I_1, I_2, I_3, Init)$;      %% perceive initial order
  $get\_Goal\_Order(P_1, P_2, P_3, Goal)$;         %% inquire about goal order
  $sort\_cups(P_1, P_2, P_3, 4)$;                  %% start dt-planning
endproc
proc $sort\_cups(P_1, P_2, P_3, H)$,
  solve$(H, reward\_cup(P_1, P_2, P_3)$,
    while$(\neg(p1 = pos(P_1) \wedge p_2 = pos(P_2) \wedge p_3 = pos(P_3)))$ do
      pickBest$(cup, \{red, green, blue\}$,
        pickBest$(to, \{p_1, p_2, p_3, p_4\}, move\_cup(cup, pos(cup), to)))$
    endwhile
  endsolve
endproc
```

a sequential program calling sub-procedures for specific tasks. The first step is a call to *get_Initial_Order* to perceive the initial order of the cups on the desk. That is, the robot uses its vision system to detect and recognize three cups, namely one cup for each of the colors red, green, and blue. It stores the initial order of the cups in variables I_i. The call to *get_Goal_Order* then initiates an interactive procedure where the robot asks the user to specify the desired goal positions for the three cups. To do so, the user is requested to point at a position on the table and to say which cup should be placed at that position. For this to work CAESAR's modules for speech and gesture recognition are constantly running. They post the results of their recognition along with a time-stamp to a central blackboard. All other modules and the high-level control can access the information there. In the robOCD scenario, the system uses the simultaneous occurrence of position keywords like *there* in the speech recognition data and a pointing gesture in the gesture recognition output to determine the desired goal position of a specific cup.

Apart from constructs known from imperative programming languages, e.g., if-then-else, loops, and procedures, READYLOG also offers less common constructs like **solve** and **pickBest**. The former initiates decision-theoretic planning, the latter is the non-deterministic choice of argument. During planning, the logical specification of the

Algorithm 3: The simplified READYLOG policy for an example run. The initial order was "green, blue, red", the desired order is "red, green, blue".

```
1 $exogf\_Update$, **if** $\neg done$ **then**
2    $move\_cup(\text{blue}, cup\_position(\text{blue}), \text{p}_4)$,
3    $exogf\_Update$, **if** $\neg done$ **then**
4       $move\_cup(\text{green}, cup\_position(\text{green}), \text{p}_2)$,
5       $exogf\_Update$, **if** $\neg done$ **then**
6          $move\_cup(\text{red}, cup\_position(\text{red}), \text{p}_1)$,
7          $exogf\_Update$, **if** $\neg done$ **then**
8             $move\_cup(\text{blue}, cup\_position(\text{blue}), \text{p}_3)$
9 **done**
```

dynamics in the world can be used to reason about the state of the world after executing a program. The non-determinism is resolved by opting for those choices that maximize a reward function. For details we refer the interested reader to (Ferrein and Lakemeyer, 2008) or the description in Section 3.4.3. Once CAESAR has collected all the necessary information as described above, it determines an execution strategy for the non-deterministic procedure *sort_cups* such that the desired arrangement of the cups is achieved eventually. This is done by means of the decision-theoretic planning just mentioned. In our scenario, the reward function only considers the number of actions needed to reach the desired goal situation. Therefore, CAESAR computes the re-ordering with a minimum number of movements.

The outcome of the decision-theoretic planning is a so-called policy, which is a conditional READYLOG program that contains the optimal course of action to achieve the goal. Algorithm 3 shows the simplified policy for an example run of robOCD. In the policy, positions p_1–p_3 are the initial positions of the three cups while position p_4 is used as a temporary spot in re-ordering the cups. The method *exogf_Update* is used to update the robot's world model to account for changes which were not due to the robot's actions. The current position of the cup with a specific color is retrieved again every time with the helper function *cup_position*. The initial order of the cups in the example was "green, blue, red". The desired final order is "red, green, blue". The optimal re-ordering, that is the re-ordering with the minimum number of actions consists of four *move_cup* actions. To achieve the goal, the robot needs to first move the blue cup to the spare position, then move the green cup to the second spot. After this, the red cup is moved to the first spot and finally the blue cup can be put at the third position. This yields the final order.

The application described above shows the potential benefit of deliberation and that it can be integrated in the behaviour specification of a robot very easily. The high-level control with its decision-theoretic optimization capabilities could be used for path-planning or for more complicated tasks such as planning the course of actions in daily activities of an elderly person. In an extended version of the demonstration the robot drives around looking for different tables with cups on them, picks up cups from those tables and moves them from one table to another. Then, the robot also uses its localization and navigation capabilities to get around and remember table positions. The different versions of the system were successfully showcased repeatedly, most notably at a national ROBOCUP competition and later at an international conference.

4.6 Discussion

In this chapter we extended our domestic service robot with capabilities that allow for natural human-robot interaction. First, we proposed a robust speech recognition system that uses two decoders in parallel, one decoder based on finite state grammars and a second decoder using TriGrams. By comparing their outputs and only accepting when both decoders yield similar results, false positive recognitions can be reduced while keeping the recognition performance high. Second, we introduced a one-step approach for face detection, recognition and learning. By using random forests we achieve very low training times which allows us to retrain the system whenever it has to incorporate a new identity. Third, our gesture recognition enables requires no training and allows to use static as well as dynamic gestures to instruct the robot or to refer to places or objects in the environment. Additional modules further enhance the human-robot interaction by speech synthesis and a facial display that can also show visual instructions. Lastly, we presented an interactive demonstration where the newly added features and the high-level control of the robot add up to a successful application of the robot system in a domestic service robot scenario.

Although the extended robot system is able to successfully perform complex tasks along with the required human-robot interaction the system is in its early stages yet. The performance of the speech recognition for far field speech has to be investigated and filter methods such as beam forming for on-board microphones with sound-source localization might be need to yield sufficiently accurate results. Still, being restricted to finite state grammars might limit the application domains. We discuss a possible alternative in Chapter 6. The detection accuracy in our face detection approach needs to be improved. This could be done by adopting a merging scheme from the literature. The gesture recognition system suffers from shortcomings in the hand verification and posture recognition. However, we identified integrating more complex features in our feature-based approach as a possible solution. Due to our modular architecture exchanging single components in the system can be realized relatively easy.

While the contributions described in this chapter represent building blocks for human-robot interaction in domestic domains, a more thorough integration with the high-level control seems necessary. As a particular instance of this integration, in the next chapter we discuss qualitative spatial representations and reasoning for applications of our domestic service robot.

5 Qualitative Spatial Representations and Reasoning

One of the issues in developing a robotic system that interacts with humans is the difference in representations with humans and machines. More specifically, humans use different means to represent their surroundings when talking to each other than a robot would do. In a technical system, in general, and in a robot, in particular, numbers are used to represent things like speed, distance, and orientation. In contrast, humans do not use numbers – at least not as their preferred means. Instead, they use imprecise linguistic notions. This is even more true for representing spatial entities such as positions. Now, if we are to build a robotic system that assists humans in their daily life, we must equip such a robot with means to understand and to communicate with humans in terms and with notions that are natural to humans.

In this chapter, proceeding from earlier work on qualitative world models for soccer robots (Schiffer, 2005; Schiffer et al., 2006b), we investigate human-oriented representations based on a fuzzy set semantics. Further, we present human-oriented control mechanism based on fuzzy controllers. At first, both these extensions are introduced in a general fashion. Then, we show possible applications of the two approaches in the domestic service robotics domain. Lastly, we give considerable attention to employing the qualitative representations to the spatial realm. That is, we use the methods presented in the earlier sections to formulate spatial relations and develop an extension of the underlying concept specific to the spatial domain.

The work on human-oriented representations has previously been published in (Ferrein et al., 2008), the findings on fuzzy-based control have been presented in (Ferrein et al., 2009) already. The application in domestic service robotics scenarios has been reported on in (Schiffer et al., 2010a) and in (Schiffer et al., 2011b). The approach on qualitative spatial reasoning in domestic domains was presented in (Schiffer et al., 2012c).

5.1 Related Work

We now review related work on both, qualitative spatial representations and reasoning as well as on fuzzy logic. Further, we briefly discuss combinations of the above and their application to robotics.

5.1.1 Qualitative Spatial Representation and Reasoning

Humans have a natural ability to perceive spatial objects and reason about their relations. However, this is different for computers. A common problem from the computational point of view is the difficulty to identify and to manipulate qualitative spatial representations. For example, although the pixels in a digital image implicitly define the locations of spatial objects in a scene, the task at hand might require a more qualitative characterization of the configuration. For instance, it may be of interest whether a particular object is *far away* from another object or not.
The processing of spatial data is a key task in many different applications, such as in geographic information systems (GIS), meteorological and physical analysis, in computer-aided design (CAD) systems, and in protein structure databases. Another important application is robotics, for example in robot navigation. Qualitative spatial reasoning (QSR) provides representational primitives and inference mechanisms for different tasks in the above areas. In the previous decades, a number of qualitative constraint calculi have been developed which are used to represent and reason about spatial configurations. Cohn and Hazarika give an overview of major qualitative spatial representation and reasoning techniques in (Cohn and Hazarika, 2001). They survey the main aspects of qualitative representations and they also consider methods for qualitative reasoning. Their survey covers ontological aspects and topological approaches as well as methods on distance, orientation, and shape.

5.1.1.1 Constraint-based Approaches to QSR

One of the most well-known works on qualitative spatial reasoning is the region connection calculus (RCC) introduced by Randell et al. in 1992. The fundamental approach bases on extended spatial entities, that is regions, and the relations, namely connections, between them. Although the RCC provides a powerful method to describe and reason about spatial structures, e.g. in geographic information systems (Cohn et al., 1997), especially for topological structures, we opt against using this calculus or any of its derivatives for several reasons. Firstly, we do not intend to use regions as primitives for representing spatial objects in our target domain, at least not yet. Secondly, we think that we do not have to rely on the generality provided by the RCC. From our point of view, the description of spatial settings needed for our applications in domestic environments can be achieved more easily. Lastly, we do not want to incur the computational cost that come with the RCC. We are looking for a mechanism that allows for qualitative representation of positional information rather than topological relations that is computationally as inexpensive as possible. An overview of constraint-based techniques for qualitative spatial reasoning along with a discussion of their computational properties is given in (Renz and Nebel, 2007).

5.1.1.2 Other Approaches to QSR

A well-known approach to qualitative representations of positional information was proposed by Freksa and Zimmermann in 1992. It bases on directional orientation information. The approach is motivated by considerations on how spatial information is available to humans and to animals: directly through their perception. Thus, cognitive considerations about the knowledge acquisition process build the basis here. Qualitative

orientation information in two-dimensional space is given by the relation between a vector from start point A to an end point B and a point C. The vector represents the orientation of a possible movement. Different positions of the point C can be described in relation to a line through A and B, and further in relation to additional lines through A and B orthogonal to the line from A to B. Reasoning with these relations is possible through four operations. The approach presented in (Freksa, 1992; Freksa and Zimmermann, 1992) is quite intuitive, not only because it is based on human cognition. Any relation is based on a vector between two points which, however, cannot be taken for granted in the context of our work, since spatial settings in a domestic environment do not always involve a movement, but they may also describe static situations. Even if we consider the intrinsic orientation of an object to construct such a vector, not all the relations are meaningful if the vector does not have a length. Further, we aim for unified representation for distance and orientation. This is also why we do not consider $\mathcal{OPRA}_m$ (Moratz et al., 2005) or variants thereof despite the favourable property of adjustable granularity.
We already discussed a group of approaches that we will be using in more detail in Chapter 3. Namely, the works presented in (Hernandez, 1991; Hernandez et al., 1995; Clementini et al., 1997) serve as a basis for our approach to qualitative spatial representations.

5.1.1.3 Context in Positional Information / Frames

The concept of *frames* that we introduce in Section 5.5.3 to capture the context of a positional reference is not new. It has been found to develop in children already (Piaget, 1955). In schema theory (Arbib, 1998) the similar concept of a schema is used to structure modelling of functional units such as perception or motion. It has even been attempted to replicate the construction of meaning by means of learning structure, e.g. from grammatical constructions (Dominey and Boucher, 2005). We rely on frames to capture the contextual information needed to be able to relate positional information given with different contexts to one another. Therefore we restrict ourselves to frames in a positional sense and we try to keep it as simple as possible.

5.1.1.4 Applications of QSR in Robotics

Müller et al. (2000) present an application of qualitative spatial representations to robot navigation. They consider the following scenario: In a hospital, a patient should visit a certain room for a medical examination. Since the patient is handicapped, a wheelchair is used to get to the examination room. Normally, the patient would be guided by a nurse, but the hospital is equipped with intelligent power wheelchairs due to time constraints. The nurse is able to instruct the wheelchair where to go so that the patient can automatically be transported to the examination room. To extract the qualitative notions, Müller et al. use methods presented in (Musto et al., 2000) which mainly base on the approach by Clementini et al. (1997) that was already mentioned. They generate qualitative motion vectors by using qualitative distance and orientation relations. Then, these qualitative vectors are generalized to simplify the motion track. By this method they stress on the coarse form and only regard the major directional changes and the overall shape of the course of motion. For a detailed account on the algorithms applied

we refer to (Musto et al., 2000).
A combination of an approach to qualitative spatial reasoning with reasoning about actions and change, namely the situation calculus, is approached in (Dylla and Moratz, 2005). While the general idea of integrating spatial reasoning with reasoning about actions and change is very appealing and we support the use of the situation calculus as an adequate mechanism for reasoning in agent control, we argue that the underlying concept of spatial neighbourhood based on the dipole calculus does not quite match our needs. There have been further approaches to integrate qualitative spatial reasoning techniques with agent control in the situation calculus such as (Pommerening et al., 2009) and (Dylla and Kreutzmann, 2010). Instead of using a particular qualitative spatial reasoning calculus, we rely on fuzzy sets for the representation of qualitative spatial notions and exploit the connection between an approach to qualitative positional information (Clementini et al., 1997) and the Cartesian space to reason about positional information. Bhatt and Loke (2008) attempt a domain-independent formalization of dynamic spatial systems. Using the formal underpinning of the situation calculus they are able to reason about the dynamics of spatial systems. Their presentation supports our argumentation that integrating reasoning about actions and change with reasoning about spatial information is helpful in building intelligent systems such as a cognitive robot.

5.1.2 Fuzzy Logic in Robotics

Our approach to human-oriented representations bases on the combination of two things. Namely, we combine qualitative (spatial) representations with fuzzy set theory. After we discussed related work on qualitative spatial representations we now review related work on fuzzy representations and control. We discussed the general adequacy of fuzzy logic for qualitative modelling in Section 3.2.1 already. This is why we now focus on applications of fuzzy logic in robotics and for qualitative spatial representations.

5.1.2.1 Fuzzy Control and Robotics

Fuzzy logic was deployed successfully for several robotics tasks. As an example for the large body of work dealing with fuzzy control and robots, we want to mention (Saffiotti, 1997) where Saffiotti shows how fuzzy controllers can be used to design robust behaviour-producing modules, and even how high-level reasoning and low-level execution can be integrated on a mobile robot. He attributes the success of fuzzy logic in control to “its ability to represent both the symbolical and the numerical aspects of reasoning. Fuzzy logic can be embedded in a full logical formalism, endowed with a symbolic reasoning mechanism; but it is also capable of representing and processing numerical data.” Liu et al. (2008) describe a robot kinematics in a qualitative fashion making use of fuzzy descriptions. They use fuzzy qualitative trigonometric functions to describe the movements of a PUMA robot manipulator. According to the authors this fuzzy qualitative description is very helpful for calibration procedures in terms of measuring accuracy or repeatability. They further stress that the fuzzy qualitative predicates provide the connection between the numerical data and interval symbols, which are then used for building up the behaviour vocabulary from which in turn the motion control of the robot can be derived. Liu (2009) uses the aforementioned fuzzy kinematics to close the representational gap between the quantitative methods applied to drive the PUMA arm, and a logical representation based on fuzzy logic predicates. Many other successful examples for fuzzy control applications are given in (Isermann, 1998). Good overviews of the fields are also given in (Dubois and Prade, 1998; Zadeh, 1989; Mendel, 1995; Passino and Yurkovich, 1998).

5.1.2.2 Fuzzy Approaches to Spatial Representations

Schockaert et al. (2006) approach a generalization of the RCC to allow for representations and reasoning in terms of fuzzy relations between vague regions using fuzzy set theory. Similarly, there have been approaches such as (Bhatt et al., 2006) to integrate the RCC into the situation calculus. Although the same reasons as given earlier hold for why we do not use the RCC, both these ideas are appealing and might be examined in future work.

Bloch and Saffiotti (2003) make use of fuzzy set theory-based representations for robot maps. They state that an application of their approach to self-localization and reasoning seems possible. Still, they mainly cover directional information only and the accuracy of the localization is pretty coarse yet. Bloch (2006) investigates the use of a fuzzy set-based framework for spatial reasoning with a focus on the use in image understanding, structure recognition, and computer vision. The author claims to be able to derive useful representations and a reasoning mechanism by making use of connections to mathematical morphology and formal logic. Interestingly, the approach presented allows for quantitative, semi-qualitative, fuzzy, and symbolic representations. We aim to retain the above flexibility in a slightly different way, though, keeping the connection of our qualitative spatial representations to Euclidean space and casting reasoning into classic geometric operations.

An early approach that tried to combine metric and topological representations using fuzzy notions was presented in (McDermott and Davis, 1984). Similarly to our basic idea of retaining the connection between qualitative concepts and metric representations, they use associated frames of reference and ranges for positions. These are first used to build a fuzzy map on which later route planning can be done. The fuzzy notions used, according to the authors, may or may not be related to fuzzy logic. We, instead, try to achieve a formal integration of the fuzzy set semantics as the basis of our integration of qualitative spatial information. Furthermore, we are more interested in representations of general positional information than on route planning.

5.2 Qualitative Representations

Specifying the behaviour of an intelligent autonomous robot or agent is a complex and non-trivial task. Based on data available from the robot's sensors, usually some kind of world model is constructed. With world model variables often a rule base is established which encodes the behaviour of the agent. This specification is often done with the help of a domain expert. A question then is how to encode the knowledge of this domain expert in the agent program. One way to ease the specification problem for the domain axiomatizer is to make use of qualitative descriptions of the world. Humans are, in general, good in describing behaviours in qualitative terms. To this end, Zadeh for example introduced so-called linguistic variables in his pioneering work (Zadeh, 1975).

These variables allow an expert to state certain rules about an application domain in a demotic way, without having to care for tricky mathematical details and complex mathematical formalizations. Another facilitation is to use a behaviour representation language which allows for flexible and expressive behaviour specifications.

In Section 3.4 we presented the situation calculus (McCarthy, 1963), the formalism that builds the basis for the high-level specification and control of our robotic system. We also presented READYLOG (Ferrein and Lakemeyer, 2008), a dialect of the robot programming and planning language GOLOG (Levesque et al., 1997) that is based on the situation calculus. READYLOG is used within the high-level control of our domestic service robot. We now aim at combining this expressive representation language and the possibility to use a qualitative world model in order to facilitate the behaviour specification for a robot.

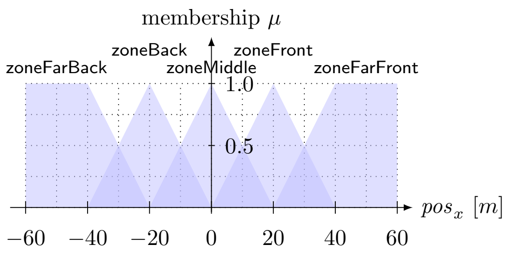
In (Schiffer et al., 2006b) we proposed a qualitative world model for the robotic soccer application. However, the integration of the qualitative world model into our action formalism at the time was ad-hoc. In this section, we put the original idea to a proper formal basis and propose a fuzzy set semantics for qualitative fluents. Our work is inspired by Dubois and Prade (1998) who argue that "the main application [...] [of fuzzy logic was] fuzzy clustering and classification, [and] a smooth interface between numerical and symbolic knowledge [...]". We introduce a new type of qualitative fluents which have a fuzzy membership function associated to them. These functions describe to what extend a fluent value belongs to a particular qualitative category. Thus, we are able to pose qualitative queries of the form: *is obstacle close?* Especially intriguing about a fuzzy set semantics for qualitative fluents is that the ranges of the qualitative predicates may overlap, and a value can, at the same time, fall into several categories.

We introduced the basic concepts of fuzzy set theory and membership functions in Section 3.2 and detailed the situation calculus in Section 3.4.1. In the following, we present our extension of the situation calculus with qualitative fluents using a membership semantics from fuzzy logic. We start with restating an example from the soccer domain from (Schiffer et al., 2006b), now with the new membership semantics as a motivating example. Then, we formally introduce fuzzy sets, fuzzy fluents, and the membership relation. Finally, we give an extended example of the new semantics of qualitative fluents.

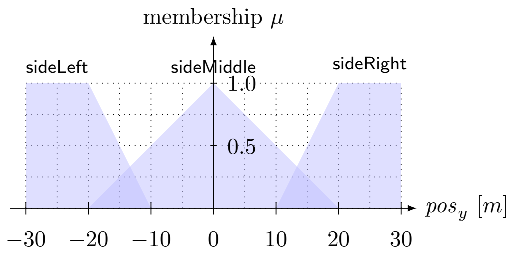
5.2.1 Membership Values For Qualitative Fluents

Consider the following soccer example, taken from (Schiffer et al., 2006b). Our qualitative world model consists of different predicates describing the position on the field as well as the orientation. The position is defined by several equivalence classes for (x, y) positions. The classes we defined here are zoneFarBack, zoneBack, zoneMiddle, zoneFront, and zoneFarFront for the x coordinate, and sideLeft, sideMiddle, and sideRight for the y coordinate. The origin is in the centre of the field with the positive x axis pointing northwards, and the positive y axis pointing to the west. The orientation of the robot is given by the qualitative categories ..., left, frontLeft, front, frontRight, right, ... Figure 5.1 shows possible membership functions suitable for the soccer domain.

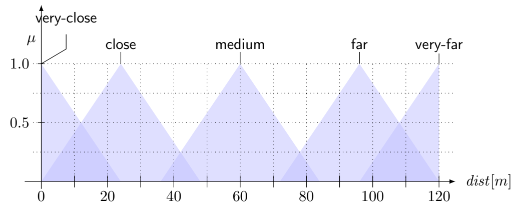
Suppose that, in situation s_n the robot is located at the qualitative pose $pose(s_n) = (\mathsf{zoneBack}, \mathsf{sideLeft}, \mathsf{front})$. Suppose further, that in situation s_n the high-level control program of the agent contains the action $\mathrm{goto\_relative}(\mathsf{far}, \mathsf{frontRight})$, mentioning the



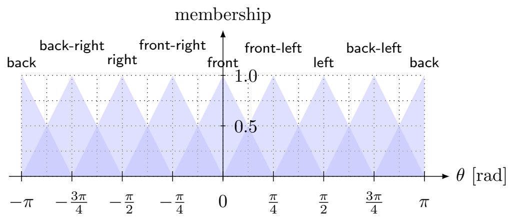
(a) membership for zones



(b) membership for sides

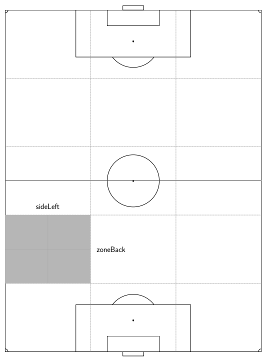


(c) membership for distance

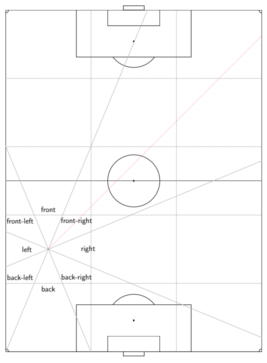


(d) membership for orientation

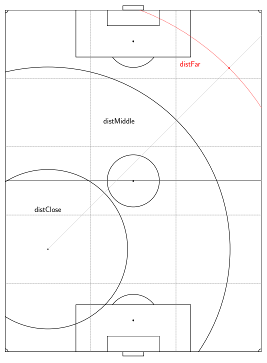
Figure 5.1: Membership functions used in the soccer domain



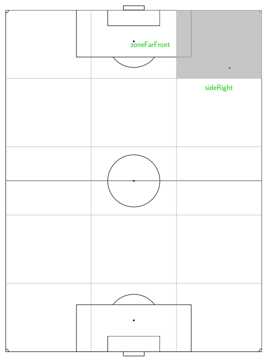
(a) quantification



(b) composition of orientation



(c) composition of distance



(d) qualification

Figure 5.2: An example for reasoning with qualitative representations in soccer

qualitative categories far and frontRight. The question is, at which position will the robot end up in the situation $s_{n+1} = do(\text{goto\_relative}(\mathsf{far}, \mathsf{frontRight}), s_n)$?
As one can see in Figure 5.2, the result should be that the agent ends up at pose $pose(s_{n+1}) = (\mathsf{zoneFront}, \mathsf{sideRight}, \mathsf{frontRight})$. Assume the following successor state axiom for the pose fluent:

$$\begin{aligned} &pose(do(a,s)) = (x,y,\theta) \equiv \\ &\quad \ldots a = \text{goto\_relative}(d,\psi) \wedge pose(s) = (x',y',\theta') \wedge \\ &\qquad \theta = \theta' + \psi \wedge x = x' + d \cdot \sin\theta \wedge y = y' + \cos\theta \ldots \end{aligned}$$

In order to calculate the new pose of the robot after performing the action, we have to defuzzify the qualitative predicate values according to the membership of the qualitative category mentioned. The literature on fuzzy logic mentions several defuzzifiers. Here, we choose the centre of gravity, which we give in Definition 5.2.7. As for now, it is sufficient to know that it returns the abscissa of the centroid of the respective category.

In the case of the triangular-shaped membership functions we used here, this is simply the apex of the membership function. For our example, this means that the pose in situation s_n is mapped to the numerical values $(x, y, \theta)^T = (-24, -21.6, 0)^T$. Now, for applying the action $\text{goto\_relative}(\mathsf{far}, \mathsf{frontRight})$ we need to defuzzify the qualitative distance and orientation far and $\mathsf{frontRight}$ to 90 and $\pi/4$. Applying the successor state axiom yields the new quantitative pose $pose(s_{n+1}) = (52.58, 25.67, \pi/4)$. Applying the membership function for the qualitative position and orientation, yields a new qualitative pose $pose(s_{n+1}) = (\mathsf{zoneFarFront}, \mathsf{sideRight}, \mathsf{frontRight})$. This is exactly as we desired.

5.2.2 Fuzzy Sets in the Situation Calculus

As we have seen in the previous section, using fuzzy membership functions facilitates the representation of qualitative world model predicates. The reason is that with fuzzy representations using membership functions, one has a means to access and alter the abstraction from quantitative to qualitative predicates, and using defuzzifiers, one can easily associate a qualitative category with a single distinct quantitative value.
In the following, we elaborate on this idea and extend the situation calculus in such a way that qualitative world model predicates can be described in terms of membership values. To ease the presentation, we assume a discrete representation of our membership functions.

Definition 5.2.1 (Reals and Linguistic Terms) *We introduce two new sorts*[1] *of the definition of to the situation calculus:* real *and* linguistic. *We do not axiomatize reals here, and assume their standard interpretation together with the usual operations and ordering relations.*[2] Linguistic terms *are a finite set of constant symbols $c_1, \dots, c_k$ in the language. They refer to qualitative classes; examples are* close *or* far. *We further require a unique names assumption for these linguistic categories, i.e. $\forall i, j, c_i, c_j . i \neq j \supset c_i \neq c_j$.*

Having introduced reals and linguistic terms into the language of the situation calculus, we can now define the degree of membership of a particular value to a given category. For ease of notation we assume that the domain of a particular category is from the domain of real numbers. In general, the domain can be defined arbitrarily.

Definition 5.2.2 (Fuzzy Sets) *Let $c_1, \dots, c_k$ be of sort linguistic. We introduce a relation $\mathfrak{F} \subseteq linguistic \times real \times [0,1]$ relating each linguistic term c of the domain, a real number, and a degree of membership in the category c as*

$$\begin{aligned}
&\forall c, u, \mu_u . \mathfrak{F}(c, u, \mu_u) \equiv \\
&\quad (c = c_1 \supset u = u_{c_1,0} \wedge \mu_u = \mu_{c_1,0} \vee \dots \vee u = u_{c_1,m_1} \wedge \mu_u = \mu_{c_1,m_1}) \ \wedge \ldots \wedge \\
&\quad (c = c_k \supset u = u_{c_k,0} \wedge \mu_u = \mu_{c_k,0} \vee \dots \vee u = u_{c_k,m_k} \wedge \mu_u = \mu_{c_k,m_k}),
\end{aligned}$$

where all $u_{c_i,j}$ and $\mu_{c_i,j}$ are constants of sort real and $\mu_{c_i,j} \in [0,1]$ respectively, i.e. $\forall c, u, \mu_u . \mathfrak{F}(c, u, \mu_u) \supset 0 \leq \mu_u \leq 1$. To ensure that, for each category, each pair (u, μ_u) is unique, we require unique names for linguistic terms. So we need Σ (the set of foundational axioms in our basic action theory) to contain the constraint

[1]Using sorts makes it easier to treat qualitative and non-qualitative fluents differently.
[2]We refer to (Harrison, 1996) for an in-depth treatment of theorem proving with reals.

$\forall c \exists u, \mu_u \forall \mu_{u'}.\mathfrak{F}(c, u, \mu_u) \wedge \mathfrak{F}(c, u, \mu_{u'}) \supset \mu_u = \mu_{u'}$. *We further require one of the* $u_{c_i,j}$ *to equal the centre-of-gravity of the respective category, i.e.* $u_{c_i,j} = cog(c_i)$ *(cf. Equation 5.1 in Definition 5.2.7).*

Note that the above definition yields a formalization of fuzzy sets $F = \{(x, \mu(x)) | x \in U\}$ as described in Section 3.2.1.1. A common notation in the fuzzy control literature is $F = \sum_U \mu_F(x)/x$, $x \in U$. That means that all value-membership pairs belonging to a particular linguistic category are enumerated. While we regard a discrete formalization here, our examples (Section 5.2.3) and the implementation make use of a continuous formulation of fuzzy sets.[3] Further note that variables occurring free in the logical sentences are implicitly universally quantified in the following definitions. As we restricted $\mathfrak{F}$ to assign membership values ranging from 0 to 1 to real numbers, we need to ensure that the fluent to which we apply qualitative categories, takes only real values. Therefore, we need to introduce fuzzy fluents as a specialization of functional fluents.

Definition 5.2.3 (Fuzzy Fluent) *A* fuzzy fluent $\mathfrak{f}$ *is a functional fluent restricted to take only values from sort* real*. We write* $\mathfrak{f}(\vec{x}, s)$ *to refer to a fuzzy fluent, and* $f(\vec{x}, s)$ *to refer to a non-fuzzy fluent.*

To query whether or not a fluent value belongs to a certain category, similar to fuzzy control theory, we introduce predicates $is_{\ldots}$. These predicates are true if a fuzzy fluent value belongs to the category in question to a non-zero degree.

Definition 5.2.4 (Membership) 1. *To query if a fuzzy fluent* $\mathfrak{f}(\vec{t}, \sigma)$ *belongs to a given category* γ*, we define the predicate* is $\subseteq$ *real* $\times$ *linguistic as*

$$\mathrm{is}(\mathfrak{f}(\vec{t}, \sigma), \gamma) \doteq \exists u, \mu_u.\mathfrak{f}(\vec{t}, \sigma) = u \wedge \mathfrak{F}(\gamma, u, \mu_u) \wedge \mu_u > 0$$

2. *Similarly, we define* $is_{\complement} \subseteq$ *real* $\times$ *linguistic, to know if a fuzzy fluent does* not *belong to a certain category*

$$\begin{aligned}\mathrm{is}_{\complement}(\mathfrak{f}(\vec{t}, \sigma), \gamma) \quad \doteq \quad & \neg \exists u, \mu_u.\mathfrak{f}(\vec{t}, \sigma) = u \wedge \mathfrak{F}(\gamma, u, \mu_u) \vee \\ & \exists u, \mu_u.\mathfrak{f}(\vec{t}, \sigma) = u \wedge \mathfrak{F}(\gamma, u, \mu_u) \wedge \mu_u < 1.\end{aligned}$$

It holds that a fluent value does not belong to a certain category, if either the value in question is not defined in terms of a fuzzy set, or the value exists and its degree of membership is less than 1 (cf. also Equation 3.3 for the definition of complement).

3. *For complex queries, for example if a fuzzy fluent value belongs to several overlapping categories at the same time, we define a predicate* $\mathrm{is}_\star \subseteq$ *real* $\times$ $(linguistic)^n$ *with* $n \in \mathbb{N}$ *as*

$$\begin{aligned}\mathrm{is}_\star(\mathfrak{f}(\vec{t}, \sigma), \gamma_0, \ldots, \gamma_n) \quad \doteq \quad & \exists u, \mu_{u,0}, \ldots, \mu_{u,n}.\mathfrak{f}(\vec{t}, \sigma) = u \wedge \mathfrak{F}(\gamma_0, u, \mu_{u,0}) \\ & \wedge \cdots \wedge \mathfrak{F}(\gamma_n, u, \mu_{u,n}) \wedge (\mu_{u,0} \star \cdots \star \mu_{u,n} > 0).\end{aligned}$$

Here, the t-norm $\star$ *refers to the min operation as given in Equation 3.2.*

[3]For the exposition we consider a finite set of value-membership pairs only. However, this could be generalized, and in the implementation we use a continuous membership function.

4. *Similarly, for asking whether or not a fuzzy fluent value belongs to one category or the other, we introduce the predicate* $is_{\oplus} : real \times (linguistic)^n$ *with* $n \in \mathbb{N}$

$$\begin{aligned}\mathrm{is}_{\oplus}(\mathfrak{f}(\vec{t},\sigma),\gamma_0,\ldots,\gamma_n) \;\doteq\;& \exists u,\mu_{u,0},\ldots,\mu_{u,n}.\mathfrak{f}(\vec{t},\sigma) = u \wedge \mathfrak{F}(\gamma_0,u,\mu_{u,0})\\ & \wedge\cdots\wedge \mathfrak{F}(\gamma_n,u,\mu_{u,n}) \wedge (\mu_{u,0}\oplus\cdots\oplus\mu_{u,n} > 0).\end{aligned}$$

In our case, the s-norm $\oplus$ *refers to the max-operator as given in Equation 3.1.*

The σ in the above definition denotes ground terms of sort situation. How the predicate "is" is used to query whether or not a fuzzy fluent belongs to a qualitative category will be shown in our one-dimensional robot domain in Section 5.2.3.
By now, we defined fuzzy fluents as a specialization of functional fluents operating on reals, introduced qualitative categories as constants of sort *linguistic*, and defined a fuzzy set in our domain axiomatization which allows for defining which values make up a qualitative category to which degree. We can further query whether or not a fuzzy fluent belongs to a qualitative category. Moreover, we can ask if a fuzzy fluent belongs to several categories at the same time, or if it belongs to the complementary category. What is still missing, though, is the ability to assign a qualitative value to a fuzzy fluent. To do this, we first need to define a defuzzifier, a function that computes a single numerical value for a given linguistic category. Note that, while we choose the *centre-of-gravity* defuzzifier here, our approach is not restricted to this. Instead, any defuzzifier could be used just as well. Then, we define a defuzzifying function that selectively applies the defuzzifier to any linguistic term.
Let us first define the function cog, which is the *centre-of-gravity* defuzzifier.
The centre-of-gravity is defined as follows.

Definition 5.2.5 (Centre of Gravity)

$$\begin{aligned}& cog(c) = \hat{u} \equiv\\ &\quad \exists u_0,\ldots,u_k,\mu_{u_0},\ldots,\mu_{u_k}.\mathfrak{F}(c,u_0,\mu_{u_0}) \wedge\cdots\wedge \mathfrak{F}(c,u_k,\mu_{u_k}) \wedge\\ &\qquad u_0 \neq \cdots \neq u_k \wedge \forall u^*,\mu^*.(u^* \neq u_0 \wedge\cdots\wedge u^* \neq u_k \wedge\\ &\qquad\quad \mu^* \neq \mu_{u_0} \wedge\cdots\wedge \mu^* \neq \mu_{u_k} \supset \neg\mathfrak{F}(c,u^*,\mu^*)) \wedge\\ &\qquad \hat{u} = \sum_{i=0}^{k} u_i \cdot \mu_{u_i} \Big/ \sum_{i=0}^{k} \mu_{u_i}\end{aligned}$$

■

We use the centre-of-gravity in a function $defuzz$ as follows.

Definition 5.2.6 (Defuzzifying Qualitative Category Values) *Let* τ *be a term of* $\mathcal{L}_{sitcalc}$. *We define a function* $defuzz$ *inductively as:*

1. *if* τ *is an atomic term*
   (a) *and* τ *is of sort linguistic, then* $defuzz(\tau) = cog(\tau)$
   (b) *otherwise* $defuzz(\tau) = \tau$
2. *if* τ *is a non-atomic term of the form* $f(\vec{t})$ *with* $\vec{t} = t_1,\ldots,t_n$, *then* $defuzz(\tau) = f(defuzz(t_1),\ldots,defuzz(t_n))$

■

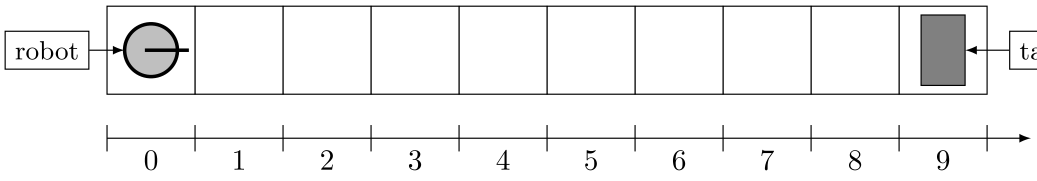


Figure 5.3: The one-dimensional domestic robot world.

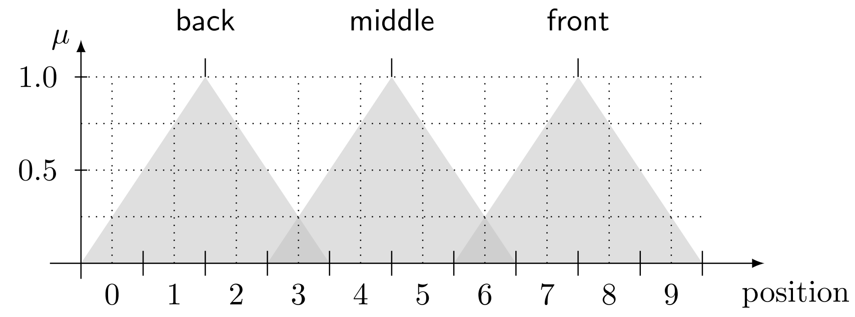
Applying the defuzzifying function defuzz, a fuzzy fluent's function value is eventually substituted by its defuzzified value. This can, for example, be used in a fluent's successor state axiom as we show in the example in Section 5.2.3. Note that the number k in the definition of the centre-of-gravity defuzzifier above refers to the number of all value-membership pairs defined in the fuzzy set for the linguistic categories plus one value for each category itself (Definition 5.2.2). Further note that the number of value-membership pairs is required to be finite. As the above definition of a defuzzifier is not closed under division in general, note that the definition is however well-defined. This is because we postulate that the centre-of-gravity for a qualitative category will be added to the set explicitly. In our implementation, where we make use of continuous fuzzy sets, this requirement can be dropped, as the set then is closed under division.
With the function defuzz we can now turn to assigning qualitative values to a fuzzy fluent.

Definition 5.2.7 (Assignment of Fuzzy Category Values) *For a fuzzy fluent $\mathfrak{f}(\vec{x}, s)$ we define a special assignment operator $:=$ which assigns the value of a category c to the fluent $\mathfrak{f}$ in situation s. The intention is to assign the category's mean value $\hat{u}$ with $cog(c) = \hat{u}$ as given in Definition 5.2.5. The assignment action can be formalized by adding the following case to the successor state axiom of fluent $\mathfrak{f}$: $\mathfrak{f}(\vec{x}, do(a, s)) = \hat{u} \equiv \ldots a = {:=}(c) \wedge cog(c) = \hat{u} \ldots$.*

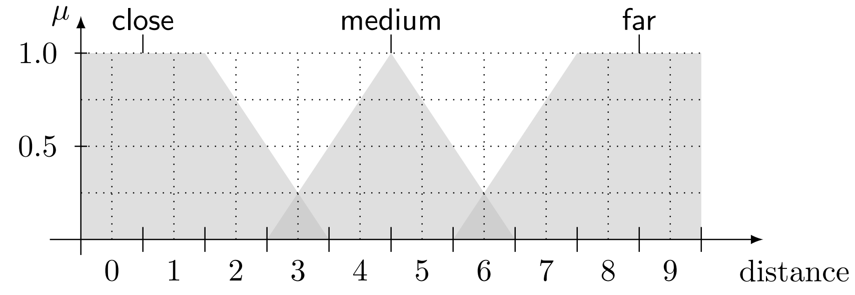
We have now defined everything we need to reason with qualitative predicates based on fuzzy membership functions. We illustrate this with an example now.

5.2.3 A One-dimensional Example

To illustrate reasoning with qualitative positional information using linguistic terms and the representations introduced above, consider the following simple example. A robot is situated in a one dimensional room with a length of ten metric units as depicted in Figure 5.3.
To keep things simple, we restrict ourselves to integer values for positions in the following. We have one single action called $\mathrm{gorel}(d)$ denoting the relative movement of d units of the robot in its world. For sake of simplifying the notation in this example, we assume that this action is always possible, i.e. $Poss(\mathrm{gorel}(d), s) \equiv \top$. The action has impact on the fluent pos which denotes the absolute position of the robot in the world. The position of the table is defined by the macro $pos_{table} = p \doteq p = 9$. In the initial situation, the robot is located at position 0, i.e. $pos(S_0) = 0$. We partition the distance in categories close, medium, and far, and introduce qualitative categories for the position of the robot as back, middle, and front. We give the (fuzzy) definition of those categories below, where we use (u_i, μ_{u_i}) as an abbreviation for $u = u_i \wedge \mu = \mu_{u_i}$.



(a) Qualitative positions of a robot in a one-dimensional world.



(b) Qualitative distance in the one-dimensional world.

Figure 5.4: Membership functions for position and distance in our one-dimensional robot domain

The fuzzy categories for the position of the robot in the world can be defined as

$$\begin{aligned}&\mathfrak{F}(position, u, \mu_u) \equiv\\ &\quad(position = \textsf{back} \supset (0, 0.25) \lor (1, 0.75) \lor (2, 0.75) \lor (3, 0.25) \lor (3/2, 0.5)) \land\\ &\quad(position = \textsf{middle} \supset (3, 0.25) \lor (4, 0.75) \lor (5, 0.75) \lor (6, 0.25) \lor (9/2, 0.5)) \land\\ &\quad(position = \textsf{front} \supset (6, 0.25) \lor (7, 0.75) \lor (8, 0.75) \lor (9, 0.25)),\end{aligned}$$

while the distances can take the values

$$\begin{aligned}&\mathfrak{F}(distance, u, \mu_u) \equiv\\ &\quad(distance = \textsf{close} \supset (0, 1.0) \lor (1, 1.0) \lor (2, 0.75) \lor (3, 0.25) \lor (13/12, 0.5)) \land\\ &\quad(distance = \textsf{medium} \supset (3, 0.25) \lor (4, 0.75) \lor (5, 0.75) \lor (6, 0.25) \lor (9/2, 0.5)) \land\\ &\quad(distance = \textsf{far} \supset (6, 0.25) \lor (7, 0.75) \lor (8, 1.0) \lor (9, 1.0) \lor (95/12, 0.5)).\end{aligned}$$

For readability reasons, we assume in this example that the robot can only move around in integer steps. Restricting to integers requires to use an altered version $cog'(c)$ of the centre-of-gravity defuzzifier formula: $cog'(c) \doteq \lfloor cog(c) \rfloor$. Of course our function $defuzz$ has to mention cog' instead of cog then. A graphical illustration of the membership functions for position and distance is given in Figure 5.4.
In the definition of the successor state axiom of the fluent pos, we have to handle its qualitative categories. We need to apply the function $defuzz(c)$ to the qualitative term which yields always a quantitative representative:

$$\begin{aligned}&pos(do(a, s)) = y' \equiv y' = defuzz(y) \land\\ &\quad a = \mathrm{gorel}(d) \land y = pos(s) + d' \lor a \neq \mathrm{gorel}(d) \land y = pos(s).\end{aligned}$$

Suppose we want to evaluate the robot's position and its distance to the table. Therefore, we define a functional fluent $dist$ which returns the distance between the robot and the table:

$$\begin{aligned} & dist(do(a,s)) = d \equiv \\ & \quad \exists p_1 . pos_{table} = p_1 \land \exists p_2 . pos(do(a,s)) = p_2 \land d = p_1 - p_2 . \end{aligned}$$

Now that we have linguistic terms for position and distance, we want to showcase that qualitative statements can in fact be used easily and that their integration in our reasoning framework yields correct results for the same. For that, we consider three main uses of qualitative notions: (1) in specifications of the initial situation, (2) linguistic terms in action arguments, and (3) fluents taking a qualitative category as the result of an action.

Linguistic Terms in the Initial Situation Suppose the robot's position in situation S_0 is characterized by the linguistic term back and the table is located at position 9, i.e. $\mathcal{D}_{S_0} = \{pos(S_0) = \mathsf{back}, dist(S_0) = 9\}$. Suppose now that the robot travels 4 units to the right. Then we can show that

$$\mathcal{D} \models \mathrm{is}(pos(do(\mathrm{gorel}(4), S_0)), \mathsf{middle}) \land \mathrm{is}(dist(do(\mathrm{gorel}(4), S_0)), \mathsf{medium}).$$

Proof Sketch Using regression and the successor state axiom for the fluent $dist$ we apply the centre-of-gravity $cog'(\mathsf{back}) = 1$ if the value of a fuzzy fluent is a linguistic term in the initial situation. It thus holds in S_0 that $\mathcal{D} \models \mathrm{is}(dist(S_0), \mathsf{far})$. By performing the action $\mathrm{gorel}(4)$ the robot moves four positions to the right. The proposition holds because $pos(do(\mathrm{gorel}(4), S_0)) = 1 + 4 = 5$ and $\mathfrak{F}(\mathsf{middle}, 5, 0.75)$ has a non-zero membership value. The quantitative distance from 5 to 9 equals 4 units or medium distance, as is given by $\mathfrak{F}(\mathsf{medium}, 4, 0.75)$. □

Qualitative Statements in Action Arguments Suppose now that the robot's control program contains the action $\mathrm{gorel}(\mathsf{far})$ mentioning the qualitative term far. At which position will the robot end up in situation $s = do(\mathrm{gorel}(\mathsf{far}), S_0)$? It follows that

$$\mathcal{D} \models \mathrm{is}(pos(do(\mathrm{gorel}(\mathsf{far}), S_0)), \mathsf{front})$$

i.e. the robot ends up in the front part of its world after executing $\mathrm{gorel}(\mathsf{far})$.

Proof Sketch Determining the robot's position in situation $do(\mathrm{gorel}(\mathsf{far}), S_0)$ we again use regression. It is sufficient to show that $\mathcal{D}_{S_0} \models \mathrm{is}(\mathcal{R}[pos(do(gorel(\mathsf{far}), S_0))], \mathsf{front})$ which is –according to the successor state axiom above– regressed to $pos(S_0) = cog'(\mathsf{back}) \land \mathfrak{F}(\mathsf{far}, 7, 0.75) \land d' = cog'(\mathsf{far}) \land \mathrm{is}(y = cog'(\mathsf{back}) + cog'(\mathsf{far}), \mathsf{front}) \equiv \mathrm{is}(y = 1 + 7, \mathsf{front}) \equiv y = 8 \land \mathfrak{F}(\mathsf{front}, 8, 0.75) \land 0.75 > 0$. Hence, we can infer that the robot ends up at position front. □

Using Qualitative Categories for Fluents as a Result of an Action Assume that apart from $\mathrm{gorel}(x)$ there is another action $\mathrm{go}(x)$ which makes the robot move directly to position x. The successor state axiom of $\mathrm{go}(x)$ is given as $pos(do(a,s)) = y \equiv a = \mathrm{go}(x) \land y = x \lor a \neq \mathrm{go}(x) \land y = pos(s)$. What happens if we put in a qualitative category there, i.e. at which position will the robot end up in situation $s = do(\text{go}(\mathsf{front}), S_0)$? It turns out that we have

$$\mathcal{D} \models \text{is}(pos(do(\text{go}(\mathsf{front}), S_0)), \mathsf{front})$$

i.e. the robot ends up in the front part of its world after executing $\text{go}(\mathsf{front})$.

Proof Sketch When regressing a formula that contains a linguistic term, the defuzzification function (e.g. centre-of-gravity $cog'(c)$) is applied if the result of a previous successor state axiom assigned a qualitative term to the fuzzy fluent. Then, $\mathcal{D} \models \text{is}(pos(do(\text{go}(\mathsf{front}), S_0)), \mathsf{front})$ iff $\mathcal{D}_{S_0} \models \text{is}(\mathcal{R}[pos(do(\text{go}(\mathsf{front}), S_0))], \mathsf{front})$ which is regressed to $pos(S_0) = cog'(\mathsf{back}) \wedge x = \mathsf{front} \wedge u = cog'(\mathsf{front}) \wedge \text{is}(u, \mathsf{front}) \equiv \text{is}(u = 7, \mathsf{front}) \equiv u = 7 \wedge \mathfrak{F}(\mathsf{front}, 7, 0.75) \wedge 0.75 > 0$. Thus, it can be inferred that the robot will end up at position front. $\square$

5.2.4 Summary

We formalized a novel semantics for qualitative fluents in the situation calculus based on fuzzy sets. Each quantitative fluent value is associated with a membership value, describing to which degree the respective fluent value belongs to the qualitative category. For each qualitative category, one specifies pairs (u, μ_u) defining to which degree μ_u the fluent value u belongs to a particular category.
To be able to query, if a fluent value belongs to a qualitative category, we introduced the predicate is which is true iff a fluent $\mathfrak{f}$ belongs to the category c. Also, we introduced an assignment operator which allows to assign a qualitative category to a fluent. As is done in fuzzy control theory, the semantics of this operation is *defuzzification*, i.e., we assign a single quantitative value as a representative for the whole qualitative category. With this, we have a defined semantics for qualitative values in the situation calculus. It allows for integrating human oriented notions in the world model and the high-level control of a domestic service robot easily. With the contribution above, we can also interpret fuzzy rules of the kind "$\text{if } A \text{ is } R_1 \text{ and } \cdots \text{ and } A \text{ is } R_n \text{ then } Y{:=}B$". We detail using such rules to allow for qualitative control in the next section.

5.3 Qualitative Control

In the last decades there have been several approaches on how to specify the high-level behaviour of an agent or robot. Especially in domains where a fast reaction time is required approaches such as planning tend to fail due to their increased computational complexity. That is why one often seeks to combine reactive control with high-level deliberation trying to get the best of both worlds. In the high-level control of our domestic service robot we make use of the Golog dialect Readylog (Ferrein and Lakemeyer, 2008). The paradigm in Golog was to combine deliberation with explicit programming (Levesque et al., 1997). Readylog comes with many useful features like on-line fluents or continuous change. We now combine Golog with fuzzy control as a means for reactive decision making while retaining the advantages of deliberation and a logic-based robot controller framework. Note that we use Golog as a synonym for Readylog in the following. Our implementation is based on Readylog, however, the formal semantics we give can be transferred to any other Golog dialect easily.

Before we start with defining the predicates, which map the common fuzzy inference rules into logic, we introduce new program statements for embedding fuzzy controllers in our control language and define their transition semantics formally.

5.3.1 Fuzzy Controller in Golog

In this section we show the embedding of a fuzzy controller in the Golog language. We presented the general architecture of a fuzzy controller in Section 3.2.2.
Following Figure 3.1, we need some means to do the fuzzyfication, apply some inference mechanism to our rule base and defuzzify the output again. The fuzzyfication and defuzzyfication we have from the previous section already, so what is missing is the inference mechanism and the rule base. Therefore, we start with introducing a new statement

$$\textbf{fuzzy\_controller}(\text{rule-base})$$

to our language. The rule base basically consists of conditionals in form of if-then statements which encode the fuzzy rules. These conditionals can be grouped in sequence. Moreover, we additionally define actions which will be selected as output in case no other rule matches. Thus, a rule base can be seen as a restricted Golog program consisting of a sequence of if-then statements and a number of fallback rules in case none of the conditional statements matches. For example, a rule base might look like

$$\begin{aligned}&\textbf{if } \phi \textbf{ then } \mathit{assign}(\mathfrak{f}, c_k); \cdots ;\\ &\textbf{if } \psi \textbf{ then } \mathit{assign}(\mathfrak{g}, c_l);\\ &\textbf{otherwise}(\mathit{assign}(\mathfrak{f}, c_n); \mathit{assign}(\mathfrak{g}, c_m))\end{aligned}$$

Formally, we define

$$\begin{aligned}&\mathit{Trans}(\textbf{fuzzy\_controller}(p), s, \delta, s') \equiv\\ &\quad \exists p', d.\mathit{infer}(p, p', d, s) \land s' = s \land\\ &\quad (Final(p') \land \delta = d \lor \neg Final(p') \land \delta = p')\end{aligned} \tag{5.1}$$

With providing a predicate for the transition semantics we yield a seamless integration with the semantics of Golog (cf. Section 3.4.2.1). Note that free variables in formulae are implicitly universally quantified. The predicate $\mathit{infer}(p, p', d, s)$ is the interface to our fuzzy inference engine. The idea is that a certain fuzzy rule base p and some particular world situation s are handed over to the inference engine, and the result will be a sequence of output assignments for the control variables. This way, we transform the rule base to one or more matching fuzzy outputs. The variable d defines some fallback output assignment, as we will describe below. We will defer the explanation of the $Final$ predicate in the definition above until we introduced the definition of conditionals and the fallback statement. For now it is sufficient to know that the remaining program δ will consist of one or more assignment actions which will assign a value to our control output variables.
To interpret a fuzzy conditional, we use

$$\begin{aligned}&\mathit{infer}(\textbf{if } \phi \textbf{ then } \mathit{assign}(\mathfrak{f}, c), p', d, s) \doteq\\ &\quad \phi[s] \land p' = \mathit{assign}(\mathfrak{f}, c) \lor \neg\phi[s] \land p' = \mathit{nil}\end{aligned}$$

where $\phi[s]$ stands for the formula ϕ with the situation argument restored. If the antecedent ϕ of a fuzzy rule holds, then we will add the consequence $assign(\mathfrak{f}, c)$ to our control output p', otherwise the rule will contribute nil. Before we discuss how ϕ is evaluated, we need to define how sequences are defined in terms of the predicate $infer$. As stated above, our rule base consists of sequences of conditionals and fallbacks. Hence, we need to interpret sequences:

$$\begin{aligned} &infer((p_1; p_2), p', d, s) \doteq \\ &\quad \exists p'_1, p'_2 . infer(p_1, p'_1, d, s) \land \\ &\quad infer(p_2, p'_2, d, s) \land p' = p'_1; p'_2. \end{aligned}$$

Each matching fuzzy rule will be replaced by its consequence, i.e. the assignment statement, while non-matching ones contribute nil. Sometimes, it may happen that no given rule in a controller block matches at all, nevertheless some output would be required. We therefore define an additional statement $\mathbf{otherwise}(assign(\mathfrak{f}, c); \ldots)$, which is interpreted in case the control output was the nil action after evaluating the rule base. We allow one or more assignment actions as parameter of the fallback, which is denoted by the ellipsis:

$$\begin{aligned} &infer(\mathbf{otherwise}(assign(\mathfrak{f}, c); \ldots), p', d, s) \doteq \\ &\quad d = assign(\mathfrak{f}, c); \ldots \end{aligned}$$

With the predicate $infer$ we are able to interpret the statements which are allowed inside the rule base. The idea is pretty much the same as using Golog's evaluation semantics to find a legal action sequence (cf. (Reiter, 2001) for example). Now we can also see why we used the $Final$ predicate in the definition of our fuzzy controller statement (Equation 5.1). It holds $Final(nil; \ldots; nil, s) \equiv true$, i.e., if all rules in the rule base contributed nil, the final predicate becomes true and we choose a fallback output. If one rule matched, then the program p' in Equation 5.1 contains an assignment action for which $Final(nil; \ldots; assign(\cdot, \cdot); nil; \ldots; nil, s) \equiv false$.

5.3.2 Fuzzy Inference in Golog

In order to interpret fuzzy conditionals we need to define the truth value of these conditionals. For that, we define the truth values for the following inference rules as given by Zadeh (1989): (1) Entailment rules, (2) Negation rules, (3) Conjunction and Disjunction rules, and (4) Categorical rules.

Entailment rules The entailment rule refers to a simple entailment of the form

$$\texttt{IF X} = \texttt{C THEN Y} = \texttt{B} \ldots .$$

If some antecedent of a fuzzy rule holds, i.e. the control variable `X`'s value belongs to a certain category `C`, then the consequence of the rule is applied by means of rule 4 (see below). If $\phi = is(\mathfrak{f}, c)$, then

$$\phi \equiv is(\mathfrak{f}[s], c).$$

As we already defined the value-membership relation in the situation calculus (cf. Definition 5.2.4), the above definition is straight-forward.

Negation rules In fuzzy control there are rules which refer to an object not being part of a certain category. Usually, they are written in the form

$$\texttt{IF X} \neq \texttt{C THEN Y} = \texttt{B}\ldots$$

i.e. if $\phi = is_{\overline{\texttt{C}}}(\mathfrak{f}, c)$ then

$$\phi \equiv is_{\overline{\texttt{C}}}(\mathfrak{f}[s], c).$$

Conjunction and Disjunction rules Fuzzy variables are connected disjunctively and conjunctively, resp., as in

$$\texttt{IF X} = \texttt{C}_1 \circ \cdots \circ \texttt{X} = \texttt{C}_\texttt{n} \texttt{ THEN Y} = \texttt{B}\ldots$$

where $\circ \in \{\star, \oplus\}$ refers to the t-norm and s-norm, respectively.
Hence, if $\phi = is_\circ(\mathfrak{f}, c_1, \ldots, c_n)$ then

$$\phi \equiv is_\circ(\mathfrak{f}[s], c_1, \ldots, , c_n)$$

referring to $is_\star$ and $is_\oplus$ as given in Definition 5.2.4. As the fuzzy conditions ϕ and ψ are logical formulae defined by the predicate is (Definition 5.2.4), also $\phi \wedge \psi$, $\phi \vee \psi$, and $\neg\phi$ are formulae and we can compose complex fuzzy conditions.

Categorical rules Categorical rules are rules stating properties of fuzzy variables, like X *is small*. In our Golog dialect, this simply breaks down to assigning a certain category to a fuzzy fluent. We do this by making use of the situation calculus action $assign$ as given in Definition 5.2.7.

Hedges
As another useful feature for dealing with linguistic terms, we introduce so-called *hedges*. Hedges are linguistic modifiers that are acting on a fuzzy set membership function in order to modify its meaning. An common example is the hedge very. If weak pressure is a fuzzy set, then very weak pressure, or extremely weak pressure are examples for hedges. One can distinguish between several kinds of hedges: (1) Concentration, (2) Dilatation, and (3) Artificial Hedges.
The idea of concentration is to express the cumulativeness of a membership function. For example, we would like to express that something is very small, or very very large. This is done by means of concentration, and we introduce some new function into the situation calculus for this purpose.

Definition 5.3.1 *Let $\mathfrak{f}(\vec{x}, s)$ be a fuzzy fluent with normalized support $\mathfrak{F}(c, u, \mu)$.*[4] *(1) Concentration. We define a function $very : \mathbb{R} \to \mathbb{R}$ with $very(\mu_c) = \mu_c^2$; (2) Dilatation. we define a function $less : \mathbb{R} \to \mathbb{R}$ with $less(\mu_c) = \mu_c^{\frac{1}{2}}$; and (3) Artificial Hedges. We define a function $plus : \mathbb{R} \to \mathbb{R}$ and a function $minus : \mathbb{R} \to \mathbb{R}$ with $plus_c(u) = (\mu(u))^{1.25}$ and $minus_c(u) = (\mu(u))^{0.75}$.*

These hedges can help us keep the number of categories in the definition of the membership functions smaller and we can still make use of finer grained qualitative descriptions in our rules.

[4]With normalized support we mean that the values of $\mathfrak{F}$ range between 0 and 1.

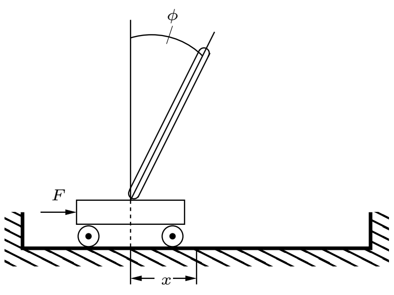


Figure 5.5: The Pole Balance Task

Table 5.1: The Control Parameters

Symbol	**Name**	**Description**
ϕ	Pole angle $[\mathrm{rad}]$	current angle of the pole
$\dot{\phi}$	Pole velocity $[\mathrm{rad/sec}]$	angular velocity of the pole
$\ddot{\phi}$	Pole acceleration $[\mathrm{rad/sec^2}]$	ang. acceleration of the pole
x	Cart position $[\mathrm{m}]$	measured as relative offset from middle of track
$\dot{x}$	Cart velocity $[\mathrm{m/sec}]$	current velocity of the cart
$\ddot{x}$	Cart acceleration $[\mathrm{m/sec^2}]$	current acceleration of the cart
m_c	Mass of the cart, $m_c = 1$ kg	$\div$
m_p	Mass of the pole, $m_p = 0.1$ kg	$\div$
l	Length of the pole, $l = 1$ m	$\div$
t	Time $[\mathrm{sec}]$	$\div$
F	force $[\mathrm{N}]$	force applied to the cart in steps of $(\pm 10$ N)
h	track limit, ± 2.5 m	border of the track
r	pole failure angle $[\mathrm{rad}]$	$r \in [-0.209, 0.209]$, ($\pm 12°$ to $0°$)
τ	time step	discrete time step

5.3.3 A Fuzzy Control Example in Golog

As an example of how and where to use the integration of fuzzy controllers as defined above we take a look at a classical control problem.

The Pole Balancing Domain

The pole-balancing problem requires the proposal of a close-loop feedback control system with the desired behaviour of balancing a pole (an inverted pendulum) that is connected to a motor-driven cart by a ball-baring pivot. The movement of the cart is restricted to the horizontal axis by a track, and the pole is free to move about the horizontal axis of the pivot. The state of the system is defined by four real values: the angle of the pole ϕ, the angular velocity of the pole $\dot{\phi}$, the position of the cart relative to the centre of the track x and the velocity of the cart $\dot{x}$. The output of the control system is a forward or backward movement of the cart in form of a force applied to it (see Figure 5.5 and Table 5.1).

We connected a simple pole balance simulator to our Golog run-time system for providing the closed control loop and for applying the force to the cart appropriately. In our simulator, the cart behaves according to the following motion equations:

$$
\begin{aligned}
\ddot{\phi}_t &= \frac{g \sin\phi_t + \cos\phi_t \left(\frac{-F_t - m_p l \ddot{\phi}_t^2 \sin\phi_t}{m_c + m_p} \right)}{l \left(\frac{4}{3} - \frac{m_p \cos^2\phi_t}{m_c + m_p} \right)} \\
\ddot{x}_t &= \frac{F_t + m_p l \left(\phi_t^2 \sin\phi_t - \ddot{\phi}_t \cos\phi_t \right)}{m_c + m_p}
\end{aligned}
$$

The inputs ϕ and $\dot{\phi}$ are connected to our system as exogenous fluents, the force is applied to the cart via primitive actions.

Controlling the Cart with Golog

Axiomatizing fuzzy controllers in Golog requires a basic action theory. For our axiomatization of fuzzy controllers we need to add sentences defining linguistic categories, exogenous fluents, fuzzy fluents, and membership functions. We add these sentences to the set $\mathcal{D}_{S_0}$, defining what is true in the initial situation.

$$
\begin{aligned}
\mathcal{D}_{S_0} = \{ & \ldots, \mathit{ffluent}(\phi), \mathit{ffluent}(\dot{\phi}), \mathit{ffluent}(F), \\
& \mathit{category}(\mathit{med}_{\phi}^{-}), \mathit{category}(\mathit{small}_{\phi}^{-}), \mathit{category}(\mathit{zero}_{\phi}), \\
& \mathit{category}(\mathit{med}_{\dot{\phi}}^{-}), \ldots\ \mathit{category}(\mathit{med}_{F}^{-}), \ldots \\
& \mathfrak{F}(\phi, \mathit{med}_{\phi}^{-}, \ldots, (-\pi/2, 1.0), \ldots, (\pi/2, 0.04)), \\
& \mathfrak{F}(\phi, \mathit{small}_{\phi}^{-}, \ldots), \ldots, \\
& \mathfrak{F}(\dot{\phi}, \ldots), \mathfrak{F}(F, \ldots) \\
& \mathit{connect}(\phi, \mathit{cart\_ang}), \mathit{connect}(\dot{\phi}, \mathit{cart\_ang\_vel}) \\
& \ldots \}
\end{aligned}
$$

Above, we defined the fuzzy fluents ϕ, $\dot{\phi}$ and F together with their fuzzy sets $\mathfrak{F}$. For each fluent, denoted by the predicate *category*, we define the value/membership pairs for each fluent. We defined the categories small, medium, zero for each fuzzy fluent (indicated by the subscript), for positive and negative values (indicated by the superscript). Finally, we need to pair the fuzzy fluent with an exogenous fluent, which will be needed for updating the quantitative fluent value. Here, *cart_ang*, *cart_ang_vel*, and *cart_force* are such exogenous fluents which relate to the fuzzy fluents ϕ and $\dot{\phi}$. Note that while in our formalization we refer only to discrete fuzzy sets which explicitly enumerate the value-membership relationship for every fluent value, in our Prolog implementation we make use of triangular-shaped membership functions for the cart controller, though.
Now, we can define the controller and the rule base in Golog. The fuzzy controller for the inverted pendulum is simply the set of rules based on the qualitative categories of the sensor input ϕ, the angle of the pendulum, and its angular velocity $\dot{\phi}$. The control output is the force F applied to the cart. The Golog procedure for balancing the pole and the corresponding rule base is given below. As long as the reference input $r(t)$ is not exceeded, i.e., the pendulum's angle is not beyond ± 0.209 radians, we update our sensor values, consult the rule base, and finally apply the force to the cart.
For the whole set of rules used in the pole balancing domain, we refer to the literature, e.g. (Passino and Yurkovich, 1998).

Algorithm 4: The control loop to balance the pole.

```
proc pole_balance
  while $| cart\_ang | \leq 0.209$ do
    $update$; $rulebase$; $apply\_force$
  endwhile
endproc
```

Algorithm 5: A rulebase for the pole-balance controller.

```
proc rulebase
  fuzzy_controller( ... ;
  if $\text{is}_\star(\phi, zero_\phi, \dot{\phi}, med^{+}_{\dot{\phi}})$ then $assign(F, med^{-}_{F})$;
  if $\text{is}_\star(\phi, small^{+}_{\phi}, \dot{\phi}, small^{-}_{\dot{\phi}})$ then $assign(F, zero_F)$;
  ... ; otherwise($assign(F, zero_F)$)
  ) /* end fuzzy_controller */
endproc
```

As already mentioned, the rule base consists of a block of rules enclosed by the program statement **fuzzy_controller**(). The rules themselves are arranged as sequences of conditionals, with a fuzzy condition as antecedent and an assignment action as consequence. In our controller implementation we moreover make use of the fallback assignment, which we introduced in the previous section. When the control flow returns from the rule base, we have at least one assignment to the output force. Finally, we need to apply the force to the cart.

Algorithm 6: Procedure to apply the force as returned by the controller.

```
proc apply_force
  if $F = med^{-}_{F}$ then push_left_strong
    else if $F = small^{-}_{F}$ then push_left
    ...
    else $F = med^{+}_{F}$ then push_right
  endif
endproc
```

5.3.4 Results

We implemented a pole balancing agent in Golog using fuzzy controller as it was just presented. With a simple set of 25 rules the agent was able to keep the pole upright not taking longer than 4 ms for any of its decisions. Thus, as a proof of concept, we were able to combine the reactive decision making process provided by a fuzzy controller with the expressive power of the Golog language. The experimental results are clearly preliminary and a example problem like in the pole balancing domain – a prototypical problem for applying fuzzy rules – is not very challenging. However, we think that the approach to combine purely reactive control patterns like fuzzy rules with a deliberative reasoning engine, which comes with the situation calculus and Golog, is intriguing and can be beneficially deployed in the robotics context.

5.3.5 Summary

We have presented an approach to integrate fuzzy controllers into Golog. We introduced a new program statement which allows for formulating fuzzy rules in Golog programs using the qualitative fluents introduced in Section 5.2 and we defined the Golog transition semantics of this program statement. For evaluating the antecedent of fuzzy rules we defined the truth values of these formulae in the situation calculus. The consequence of a fuzzy rule is simply mapped to a special primitive Golog action which defuzzifies the output value and assigns it to a fuzzy fluent. Also, we defined hedges like *very* or *less* which are very useful when formulating qualitative action theories.

The possibility to use such qualitative control laws in Golog extends the set of available expressions and increases the possibilities to tackle robotic high-level problems. We turn to applications in the domestic service robotics domain in the next section.

5.4 Applications in Domestic Service Robotics

Tasks where a robot should deliver a letter or fetch a cup of coffee (in a reasonably sophisticated environment) are classical applications for approaches to cognitive robotics and reasoning about actions. These domains show that solving such tasks with deploying reasoning and knowledge representation is superior to, say, pure reactive approaches in terms of flexibility and expressiveness. An even more advanced application domain is ROBOCUP@HOME as we presented in Section 2.1.2. Robots have to fulfil complex tasks such as “*Lost&Found*”, “*Fetch&Carry*”, or “*WhoIsWho*” in a domestic environment. Recall that in the first tasks the robot has to remember and to detect objects, which are hidden in an apartment, or has to fetch a cup of coffee from, say, the kitchen and bring it to the sitting room, while in the latter the robot needs to find persons and recognize their faces. The outstanding feature of these applications is that they require integrated solutions for a number of sub-tasks such as safe navigation, localization, object recognition, and high-level control (e.g. reasoning). A particular complication is that the robot may only be instructed by means of natural interaction, e.g. speech or gestures. Human-robot interaction is hence largely based on natural language. For example, in the *Fetch&Carry* task it is allowed to help the robot with hints like “The teddy is near the TV set”.

In the previous sections we extended our high-level control language READYLOG with means of qualitative representations based on fuzzy sets in Section 5.2 and we integrated fuzzy control techniques into the language as shown in Section 5.3. This enables us (1) to map qualitative predicates to quantitative values based on a well-defined semantics, and (2) to combine fuzzy control and logic-based high-level control. We have shown preliminary example applications for both these extensions so far. We now show how these concepts can be used beneficially to formulate compact solutions for tasks such as *Fetch&Carry* or *FollowMe!* that can be deployed in the ROBOCUP@HOME competitions.

5.4.1 Tasks for Domestic Service Robots

As already mentioned in Section 2.1.2 the RoboCup@Home competition should foster development in domestic service and assistive robot technology relevant for future applications (Wisspeintner et al., 2009). The competition features tasks for the robots to solve such as:

- *FollowMe!*: the robot has to follow a human through the apartment;
- *Fetch&Carry*: a human names known objects and the robot needs to fetch them. The human may give hints such as: "*The teddy is near the TV.*";
- *Walk'n'Talk*: in a guidance phase, a human instructor leads the robot around in an apartment and tells it certain landmarks such as "kitchen table", "TV set", or "fridge". In a second phase the robot is instructed to navigate to some of these just learnt places.

The rules of the RoboCup@Home competition state that a robot – to be successful in the competition – is to be endowed with a certain set of basic abilities, like navigation, person and object recognition, and manipulation. Furthermore, fast and easy calibration and setup is essential, as the ultimate goal is to have a robot up and running out of the box. Also, human-robot interaction has to be achieved in a natural way, i.e. interacting with the robot is allowed only using natural language (that is by speech) and gesture commands.

As already mentioned, humans tend to make use of qualitative concepts such as near or far. With introducing suitable qualitative concepts, we can bridge the gap between human and robot representations of domestic environments. With the extension for qualitative representations from Section 5.2 we can implement a high-level controller for deliberation that uses such concepts seamlessly. Not all parts of the solution of a domestic task require deliberation, though. For some decisions simple reactive controllers are sufficient. However, these reactive mechanisms also need to understand qualitative concepts. Here, we can make use of our embedding of fuzzy controllers in Readylog from Section 5.3.

5.4.2 Qualitative Representations for Domestic Environments

A key element of interaction between a human and a robot in the RoboCup@Home domain is to refer to objects in the environment, e.g. to give the robot some hints on where objects might be located and where places are. This information is very often spatial. That is why we now take a first attempt at developing and deploying qualitative representations for positional and directional information that can be used to instruct the robot. This attempt is largely based on work by Clementini et al. (1997) presented in more detail in Section 3.3. In this section we will keep the description very brief and we ask the reader to be patient until we discuss an extension in the next section (Section 5.5).

As described in Section 3.3 the position of a *primary object* is represented by a pair of *distance* and *orientation* relations with respect to a *reference object*. Both relations depend on a so-called *frame of reference* which accounts for several factors like the size of objects and different points of view. For the example applications in the domestic scenario we have to choose and model appropriate instances of the two relations now.

For the orientation relation one can subdivide the $360°$ around the reference object into equally spaced angular intervals (like it is common to do with a compass). It is possible to choose different levels of granularity as already discussed in Section 3.3.2. For the example in this section assume we use a level of granularity of 3 resulting in eight ($= 2^3$) different orientations. This should provide a sufficiently fine distinction to capture common human notions.

In a domestic setting we can define different distance relations according to: (1) external references such as the maximal size of the apartment: *"The plant is at the far end of the corridor"*; (2) intrinsic references used in relating objects to each other such as room or table: *"The cup is on the table close to the plate"* vs. *"The teddy is close to the TV"*; and (3) an appropriate distance system. In our domestic environment we suggest to make finer distinctions in the neighbourhood of the reference object than in the periphery. Hence, we model the distance systems accordingly, by having smaller intervals for lesser distances and larger intervals for farther distances. To account for possibly different scales derived from internal or external references we can distinguish the scales $dist\text{-}scale \in \{\text{apartment}, \text{room}, \text{object}(o)\}$, where object o refers to objects such as *table*, or *bookshelf*. This is sufficient to instantiate a set of predefined distance relation to model qualitative descriptions accordingly.

Now, to be able to make use of the positional information given to the robot, we provide a procedure $analyseHint$. It takes a hint given by the human instructor and distills the position of the object and the frame of reference accounting for the scale and the point of view from that hint. For instance:

(a) *"The plant is far on the left side of the corridor"*;
the primary object is the plant, the point of view is the view point of the robot, the distance scale is set to the size of the corridor.

(b) *"The cup is on the table close to the plate"*;
the primary object is the cup, the reference object is the plate, the distance scale is set to the size of the table. No orientation relation is given.

(c) *"The teddy is close to the TV"*;
the primary object is the teddy, the reference object is the TV, the distance scale should be set to the size of the room where the TV is located. Again, no orientation relation is given.

With this procedure at hand, we can adopt our fuzzy fluents for the qualitative distance and orientation. A possible membership function for the orientation fluent is later given in Figure 5.10. We can define membership functions for distance in a similar way. Note that in this first modelling we have individual membership functions for the distance at every scale. Later we unify this. In the next section, we give an impression of how these fluents can be used for programming the domestic service robot.

5.4.3 Qualitative Notions in High-level Programs

Now that we have an initial modelling of qualitative representations for positional information in a domestic setting we show how we can make use of these representations within our existing high-level control mechanism. Algorithm 7 shows a slightly abstracted version of a Readylog control program for the *Fetch&Carry* task.

The procedure *fetch_and_carry* takes the object that should be fetched and a user hint

Algorithm 7: A Readylog program for the "*Fetch&Carry*" test.

```
proc fetch_and_carry($object, hint$)
  analyseHint($hint$);
  $\pi(pos, for_\theta, for_{dist}).[ori\_type(for_\theta) \wedge dist\_system(for_{dist}) \wedge$
     $dist\_scale(for_{dist}) \wedge dist\_type(for_{dist}) \wedge object\_pos(pos)]?$;
  search($object$, $pos$, $for_\theta$, $for_{dist}$ )
endproc

proc search($object$, $pos$, $for_\theta$, $for_{dist}$)
  solve(while $\neg$objectFound do
            pickBest($search\_pos$ = defuzzify($pos$, $for_\theta$, $for_{dist}$));
            lookForObjectAt($object$, $search\_pos$);
         endwhile, H) /* end solve with horizon H */
  pickup_and_return($object$);
endproc
```

as input. At first, the action *analyseHint* is executed. This is a complex action which involves natural language processing. From the user phrase, the frame of reference for orientation and distance including the distance scale is extracted (as pointed out in the previous section). The action's effect axioms are changing fluent values for the fluents describing the orientation's frame of reference, the distance system, the distance scale, the distance's frame of reference as well as the qualitative position of the reference object. The next statement in the program is a "*pick*" statement (π) which is used to instantiate the free variables in the logical formula in the next test action (denoted by the "?"). The whole construct can be seen as an existential quantifier, and the effect is that the variables pos, for_θ, for_{dist} are bound. The next step is to call the search routine with these parameters. The search involves the activation of decision-theoretic planning (*solve*) at a position where the object is meant to be according to the user's hint. The position is defuzzified, taking the frame of reference information into account. That is, the position based on the distance scales and the quantitative orientations given the points of view etc. can now be calculated. The action *lookForObject* again is a complex action which actually tries to seek the object.

5.4.4 Domestic Golog Fuzzy Controllers

As detailed in Section 5.3 we integrated fuzzy controllers in Golog. If (a part of) a task does not require high-level decision making (like decision-theoretic planning as used in the previous section), but can instead be solved with a reactive mechanism it may still be convenient to make use of the qualitative representations. One example in the domestic setting for such a task is the "*FollowMe!*" test. The control of the follow behaviour can be modelled quite straight-forwardly. Already a simple rule base can be used to solve the *FollowMe!* task. The rule base could look like given in Algorithm 8.
As stated in Section 5.3.1, a rule base consists of a number of if-then rules where both, the antecedent and the consequence, mention fuzzy fluents. So, the first rule reads as follows: "*if the distance to the user is close and its speed is slow, then set the robot speed to slow*", the second rule reads "*if the distance to the user is far and its speed is medium, then set the robot speed to fast*", where *user* is the person whom to follow. The $is_\star$ predicate

Algorithm 8: A fuzzy controller for the "*FollowMe!*" test

```
proc follow_me_rulebase
  fuzzy_controller( ...;
    if $is_\star(\mathrm{dist_{user}}, \mathrm{close}, \mathrm{speed_{user}}, \mathrm{slow})$ then $assign(\mathrm{speed_{robot}}, \mathrm{slow})$;
    if $is_\star(\mathrm{dist_{user}}, \mathrm{far}, \mathrm{speed_{user}}, \mathrm{medium})$ then $assign(\mathrm{speed_{robot}}, \mathrm{fast})$;
    ...;
    otherwise$(\mathrm{speed_{robot}}, \mathrm{medium})$) ) ;/* end fuzzy_controller */
  applySpeed()
endproc
```

is defined in Definition 5.2.4 and denotes the conjunction of the fuzzy fluents $dist_{user}$ and $speed_{user}$. If neither condition applies, the fallback speed selection is set to medium. Finally, the $speed_{robot}$ fuzzy fluent has to be defuzzified, that is, a quantitative value is calculated for the qualitative class. Then, we can apply the quantitative speed to the robot motors.
By using a fuzzy controller with its simple concept of a set of rules we alleviate the specification of the control. We can use linguistic terms to describe the intended behaviour and leave the details on what values to send to the mid- and low-level modules to our automatic machinery.

5.4.5 Summary

We applied the two approaches to qualitative representations and qualitative control to the domestic service robotics domain. With the RoboCup@Home tasks *Fetch&Carry* and *FollowMe!* we showed example implementations for how qualitative representations and fuzzy controllers can be beneficially deployed. Enabling our high-level robot controller to deal with more human-oriented concepts proves useful for two reasons. For one, because in domestic applications such as RoboCup@Home the robot needs to be instructed by a human operator by natural language. For another, the intended behaviour of the robot can often be formulated in few simple rules with linguistic terms. Having qualitative representations in place allows for more human-like instructions as humans tend to use qualitative (spatial) representations such as far, left-of or slow.
The programs give an impression of how fuzzy fluents and fuzzy controllers can be deployed in domestic robot applications. In both application the relevance of spatial notions became apparent. This is why we go into detail on this in the next section.

5.5 Reasoning with Qualitative Spatial Representations

The domain of domestic service robotics features two important aspects that we need to account for: space and humans. To see why these two aspect are critical, let us look at a real world example. Suppose you want to instruct your domestic robot with the command "*Get me my cup from the kitchen table*". Besides natural language processing and sophisticated robot control for navigation, localization, object recognition etc., the robot needs to be equipped with a flexible high-level control entity that can figure out from the instruction that the robot needs to go to the kitchen, pick up "*my cup*" (if there are more cups on the table, which is the right one?), grab it, and bring it to the

human instructor. One of the challenges for the robot is to handle the spatial references often made in such commands. What is more, humans tend to use qualitative concepts and notions like near or far to refer to positions in Euclidean space. The robot needs to be able to interpret these concepts to cope with them. When reasoning techniques are deployed to come up with a problem solution for these domestic tasks, also such mechanisms need to be able to cope with the qualitative concepts.
We already mentioned that we use the logic-based plan and programming language READYLOG for the high-level control of our robot. In the preceding sections we presented two extensions that enable the use of qualitative representations and qualitative control within READYLOG. Applying these approaches in the domestic service robotics domain of ROBOCUP@HOME revealed that much of the information is spatial, for example positions of objects that is being referred to or instructions of how and where to move. In this section we revisit the use of fuzzy sets for building qualitative representations, now with a special emphasis on qualitative spatial information, in particular in domestic environments.
An important property when dealing with such qualitative spatial notions is that they depend on the context in which they were expressed. For a human, a distance "far" w.r.t. the living room has a smaller scale than "far" with respect to the garden. This means that a robot dealing with these concepts also needs to know the context in which a qualitative instruction is given. Therefore, we formalize the concept of positional contexts, so-called *frames*. Each positional information needs to be related to a particular frame. We present a unified way to transform qualitative positional information into different frames. Then, after axiomatizing the domestic robot domain in the situation calculus we present a high-level controller for a fetch-and-carry task known from RoboCup@Home in READYLOG using these techniques. With this we show how the theoretical concepts of qualitative positional information and frames can be seamlessly integrated into and deployed for a robot controller for domestic service robots in a real-world application.

5.5.1 Qualitative Positional Information

We first briefly reiterate the method to represent qualitative positional information that is the foundation of our approach. A detailed description was already given in Section 3.3. Based on a representation mechanism for qualitative orientation presented by Hernandez (1991) and a basic method for qualitative distances discussed in 1995 by Hernandez et al., Clementini et al. (1997) present a unified framework which allows for qualitative representation of positional information. This is done by combining the orientation and the distance relation. The position of a *primary object* po is represented by a pair of distance and orientation relations with respect to ro, a *reference object*. Both relations depend on a so-called *frame of reference* which accounts for several factors such as the size of objects and different *points of view*. Thus, to represent positional information, we need to define an orientation and a distance relation.

Orientation Relation
The orientation relation describes where objects are placed relatively to each other. Based on the fundamental observation of how three points in the plane relate to each other, an orientation relation can be defined in terms of three basic concepts: the

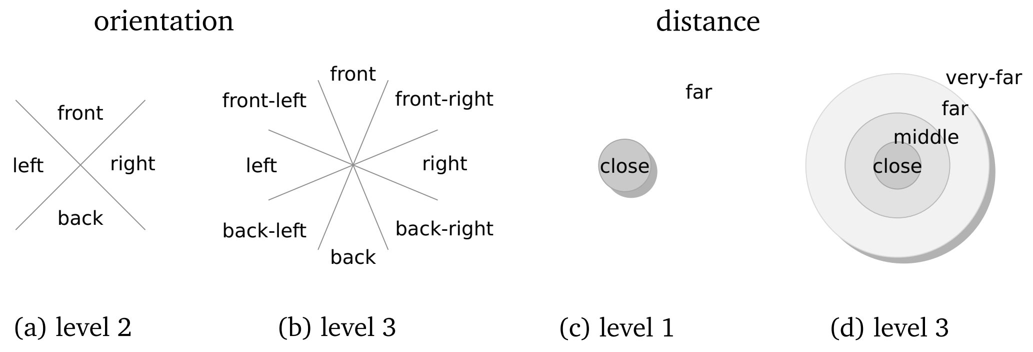


Figure 5.6: Different levels of granularity for orientation and distance according to Clementini et al. (1997)

primary object po, the *reference object* ro, and the *frame of reference* which contains the *point of view*. The point of view and the reference object are connected by a straight line. The view direction is then determined by a vector from the point of view to the reference object. The location of a primary object is expressed with regard to the view direction as one of a set of relations. The number of distinctions made is determined by the *level of granularity*. There are different levels of granularity for orientation relations. On the first level, the point of view and the reference object are connected by a straight line such that the primary object can be to the left, to the right, or on that line. Thus, the first level partitions the plane into two half-planes. On the second level, there would be four partitions, the third level would have eight, and so on (cf. Figure 5.6a and Figure 5.6b). Based on the frame of reference there is a '*front*' side of the reference object. Independent from the level of granularity there is a uniform circular neighbouring structure. In general, at a level of granularity k the set $\{\alpha_0, \alpha_1, \ldots, \alpha_n\}$ denotes the $n + 1$ orientation relations where $n = 2^k - 1$.

Distance Relation

The distance relation requires the three elements ro, po, and the frame of reference. Moreover, a distance relation requires a distance system. A commonly used distance system is the Euclidean space, which is reflexive ($dist(P_1, P_1) = 0$), symmetric ($dist(P_1, P_2) = dist(P_2, P_1)$), and follows the triangle inequality ($dist(P_1, P_2) + dist(P_2, P_3) \leq dist(P_1, P_3)$). The distance of two points expressed in a qualitative way often depends not only on their absolute positions but also on cultural and experimental factors and on the frame of reference. Similar to the orientation relation we can distinguish distances at various levels of granularity (cf. Figure 5.6c and Figure 5.6d). An arbitrary level n of granularity with $n + 1$ distinctions yields the set $Q = \{q_0, q_1, \ldots, q_n\}$ of qualitative distances. Given a reference object ro, these distances partition the space around ro such that q_0 is the distance closest to ro and q_n the one farthest away.

Combining Distance and Orientation

As already discussed in Section 3.3.4, from a quantitative point of view, the combined description of a position with the above model using distance and orientation can be seen as the representation of a point in polar coordinates. A point p in polar coordinates is defined by the distance r from the origin to this point and the angle φ measured from

the horizontal x-axis to the line from the origin to p in the counter-clockwise direction. Thus, the position of a point p is described as (r, φ). This description directly corresponds to the combination of the distance and the orientation relation. We illustrated this in Figure 3.5.
We make use of this correspondence by relating qualitative positional information given as distance and orientation in the above sense to quantitative positions in Euclidean space. This way we can simply use methods from Euclidean geometry to conduct spatial reasoning.

5.5.2 The Domestic Robot Domain

Before we present an extension of fuzzy fluents for positional information we now introduce our target domain: the Domestic Robot Domain. In this domain, a service robot is instructed by a human operator via natural language to fulfil tasks such as *Fetch&Carry* in an apartment environment. Figure 5.7 shows an exemplary instance of this domain. In this instance, there are several rooms and several pieces of furniture and objects. The human-machine interaction in this domain should be as natural as possible. Therefore, our goal is to support instructions such as "*get me the left cup on table2*" or "*bring me a coke to the living room*". A closer look at such instructions reveals that mainly qualitative positional information is used by the human instructor, i.e. qualitative distance and orientation. Further, to be able to put the information given by the human instructor into the right context, the concept of a frame of reference possibly including a point of view as mentioned already is required. For example, "far" in the context of the living room refers to a larger distance than "far" in the bath room. To cope with these types of contexts and different points of view, we introduce the concept of *frames* in the next section and formalize them in the situation calculus.
For positional information in our domestic domain, we need qualitative distance and orientation which we already introduced in Section 5.5.1. It makes sense to fix things like the distance system and the number of granularity levels for the indoor environment. We do so by specifying membership functions for distance and orientation, respectively. That means we fix a level of granularity of 5 for the distance relation with the categories very-close, close, medium-far, far, and very-far. Of course, in different frames, these categories have different scales. As the parlour is larger than the bath room, we have to scale these categories according to their respective frame. Our concept of *frame* discussed in the next sections is doing exactly that. This, however, requires, that the categories for distance are defined on a unit scale. We show our definitions of qualitative unit distance in Section 5.5.4. For the qualitative orientation, we select a level of granularity of 3, meaning that we distinguish $2^3 = 8$ different qualitative orientations. In order to relate the orientation to its context, each frame needs to have a distinguished front side. Our concept of *frame* covers this as well.

5.5.3 Positional Fuzzy Fluents and Positional Frames

We introduced a generic method for qualitative fluents in the situation calculus in Section 5.2. Now, we extend these qualitative notations for positional information. To account for the frame of reference mentioned above which carries important information about the context of positional spatial representations we introduce *positional frames*.

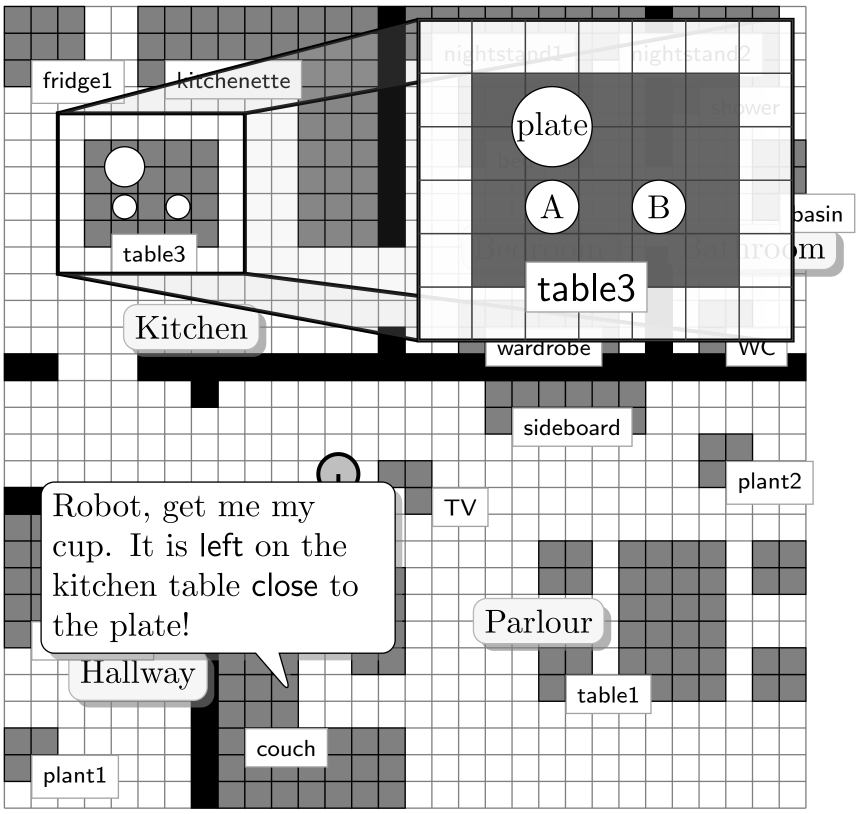
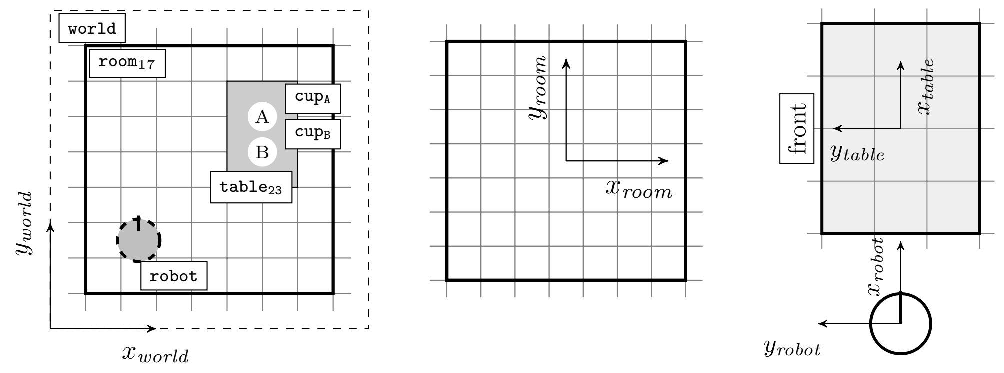


Figure 5.7: The domestic robot domain

In short, a positional frame is like the specification of a coordinate system in which a particular positional information is given. Thus, it can be used to relate positional information given in different frames to one another if we know the relation between the frames. As we are always keeping the connection between qualitative representations and their underlying quantitative information we can model the transformation between different frames of reference as a coordinate transform in Euclidean geometry.
For each domain object which should be reasoned about in a qualitative fashion, we need to know the object's coordinate and its reference frame. For example, $\texttt{table}_{23}$ could be either defined to be at a certain position in room $\texttt{room}_{17}$, or its position could be instantiated with a global world coordinate. In the former case, we would attach frame $room_{17}$ to the positional information of the table, in the latter we would attach frame $world$. Figure 5.8 is showing different example frames. Attaching its frame to an object allows for transforming the object's coordinates to any other given frame.

Positional Fuzzy Fluents and Frames

As the frame is an inherent property of positional fluents, we define a new type of fuzzy fluent, called *positional fuzzy fluent*, that always carries its frame with it.
Associated with a frame is a local Cartesian coordinate system $\mathbb{R}^2$ of an object in the world. The origin of a frame's coordinate system is defined with respect to a superordinated frame, with the $world$ being the most general frame. For each frame f_s denoting the source frame, we need to specify, how the frame's coordinate system can be transformed into the target frame f_t. The parameters we need are the origin (x_o, y_o)



(a) World or global frame (b) Room frame (c) Table and robot frame

Figure 5.8: Examples of different frames in our domain.

of the target system expressed in the coordinates of the source frame, the angle θ_o between the source and the target frame and m_o a scaling factor between the units of the respective distance systems. The scaling is due to the different frames of reference that influence the distance system. Further, it is connected to us introducing a so-called *unit-distance* which we discuss in more detail in Section 5.5.4.

As objects can be moved in the world, these parameters are not rigid and have to be defined in terms of fluents. We therefore require without loss of generality that for each object in the world, there is a sentence of the following form in $\mathcal{D}_{ssa}$:

$$\mathit{frmparam}(f_s, f_t, s) = (x_o, y_o, \theta_o, m_o) \doteq \mathit{pos}_{f_s}(f_t, s) = (x_o, y_o, \theta_o, m_o)$$

with pos_{f_s} defining the position, angle, and scaling factor of an object f_s expressed in the frame f_t. Note that *frmparam* is defined as a macro. The domain axiomatizer has to provide sentences about the initial position of an object and a successor state axiom describing how the position of the respective object changes. Further note that the basic action theory is just used, not extended. Therefore, all results for BATs shown in Section 5.2.3 still apply.

To give an example, consider Figure 5.8a. To express the position of the table in the world, we need to add the sentence $\mathit{pos}_{\mathit{table}_{23}}(\mathit{room}_{17}, S_0) = (5, 4, 0, 1)$ to $\mathcal{D}_{S_0}$ and $\mathit{frmparam}(\mathit{table}_{23}, \mathit{room}_{17}, s) = (x_o, y_o, \theta_o, m_o) \doteq \mathit{pos}_{\mathit{table}_{23}}(\mathit{room}_{17}, s) = (x_o, y_o, \theta_o, m_o)$ to $\mathcal{D}_{ssa}$. Now, we can keep track of the position of the table w.r.t. its coordinate in the kitchen. Defining the position of objects w.r.t. a superordinate frame also allows to derive, say, the coordinate of a cup on the kitchen table, even if the table was moved around in the kitchen.

Changing the Frame

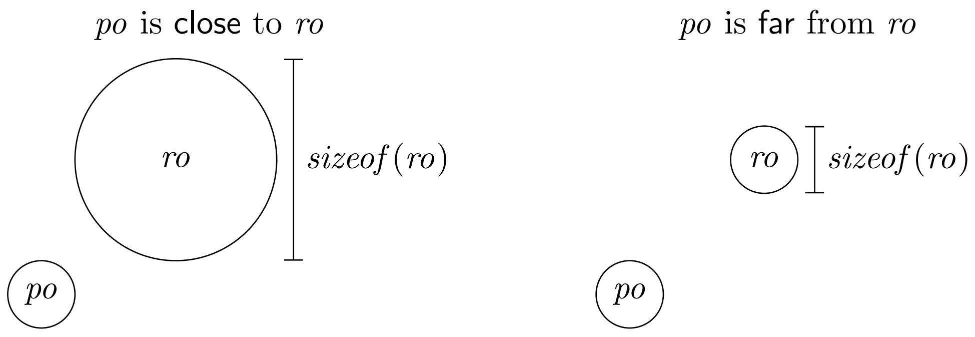
To convert between a source and a target frame, we define a function $chfrm$ as

$$\begin{aligned}
&chfrm(x_s, y_s, f_s, f_t, s) = (x, y) \doteq \\
&\quad \exists x_o, y_o, \theta_o, m_o . frmparam(f_s, f_t, s) = (x_o, y_o, \theta_o, m_o) \wedge \\
&\quad [f_s \neq f_t \wedge (x, y)^T = \begin{pmatrix} \cos\theta_o & -\sin\theta_o \\ \sin\theta_o & \cos\theta_o \end{pmatrix} \cdot \begin{pmatrix} x_o + x_s \\ y_o + y_s \end{pmatrix} \vee \\
&\quad f_s = f_t \wedge (x, y) = (x_s, y_s)]
\end{aligned}$$

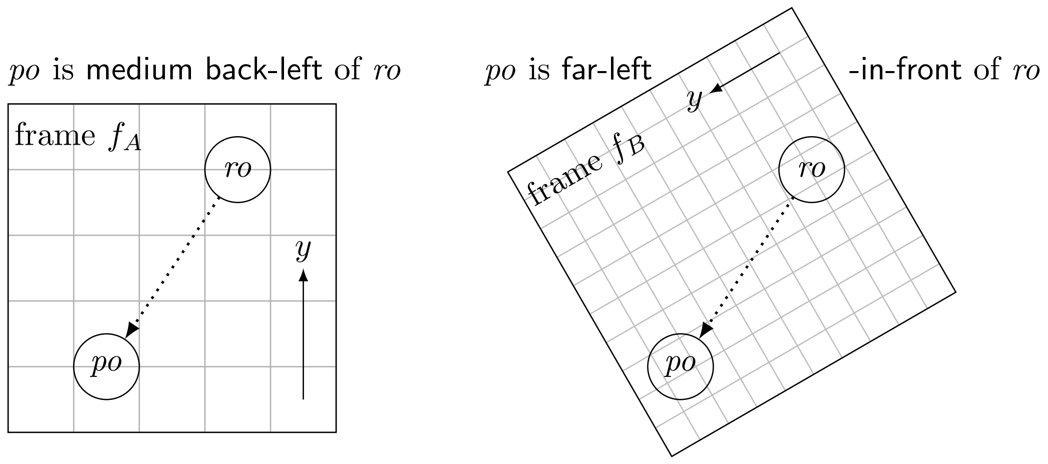
The scaling factor m_o is used to determine the unit lengths of each interval of the distance system. Each frame requires a distinct front side in order to provide a standard point of view. We assume the standard view point along the y-axis of the frame's coordinate system.
Note, that the above definition of $chfrm$ directly corresponds to a coordinate transformation in Euclidean space in terms of translation and rotation as well as scaling. This is exactly the correspondence that we exploit to retain a simple way to reason with positional information. Enabling the direct relation between the qualitative description and a coordinate based numerical representation does exactly this. Whenever reasoning for qualitative fluents is needed, we can transform them to their quantitative representatives, perform computations in Euclidean geometry and re-qualify the result to yield a qualitative value again.

Positional Frames as Frame of Reference

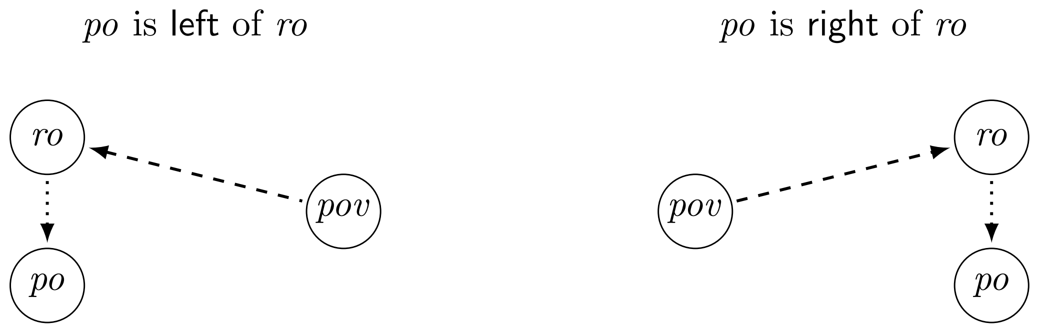
Before we show an extended example of the coordinate transformations making use of $frmparam$ and $chfrm$ in Section 5.5.5, we show that our concept of *frame* satisfies the properties of the frame of reference $FofR$ as given in (Clementini et al., 1997). According to Clementini et al. (1997), the frame of reference for the distance relation is the tuple $FofR = (D, S, T)$ with D a distance system, S a distance scaling factor, and T the type of relation. The type can be either (1) *intrinsic*, i.e. the distance is determined by an inherent characteristics of the reference object such as its size, shape, or topology; (2) *extrinsic*, i.e. the distance is determined by some external factor, for instance, the arrangement of objects or a measure for the costs involved in travelling (e.g. time); and (3) *deictic*, i.e., the distance is determined by an external point of view, e.g. the viewpoint of an observer perceiving an object. The frame of reference for orientation given in (Clementini et al., 1997) is similar to the one for distance given above but we omit the formal definition here.
From that we can derive the positional frame of reference, defining the distance and orientation of the primary object po, the reference object ro, and a point of view pov. If the relation between po and ro is *intrinsic*, then the scale is depending only on properties of ro such as the size or the weight of the reference object. Figure 5.9a is showing an example. In the left-hand figure po is close to ro because ro's size is huge, hence the distance between both objects related to ro is small. In the right-hand figure on the other hand, ro is small, and although the quantitative distance between the centres of both objects is the same, po is now far away from po. In the *extrinsic* case shown in



(a) An *intrinsic* distance relation $dist(po, ro)$. Here the relation depends on the size of ro.



(b) *Extrinsic* relations $dist(po, ro)$ for distance and $\theta(po, ro)$ for orientation.



(c) A *deictic* orientation relation $\theta(po, ro)$. Depending on the position of pov, po is either left or right of ro.

Figure 5.9: Different frames of reference

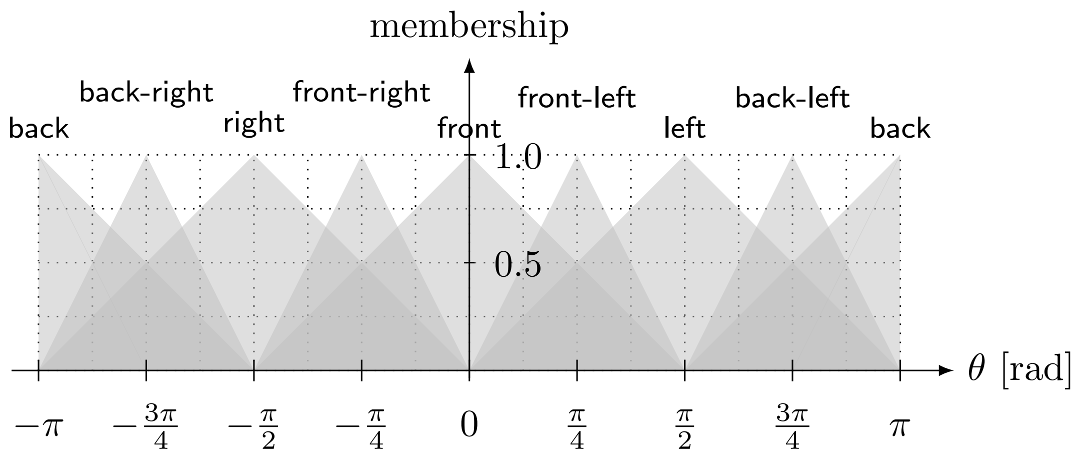


Figure 5.10: Membership function for qualitative orientation at level of granularity 3 in our domestic robot domain

Figure 5.9b, the relation between objects is only dictated by the reference frame and its size. In the left-hand case of Figure 5.9b, po is at a medium distance on the back-left side of ro. Note that, in the right-hand figure, the scale is different and the θ between f_A and f_B is about 30 degrees. Hence, po is far-left-in-front of ro. The third case in Figure 5.9c shows the *deictic* case including an external point of view pov. Therefore, in the left figure, po is left of po, while in the right figure, po is right of ro.
These cases are all captured with our concept of *frame* as given above together with fluents defining the respective coordinate transformation between two different reference systems by translation, rotation, and scaling. For example, in the first case depicted in Figure 5.9, we only need to adapt the scaling, while in Figure 5.9b we need to adopt translation, rotation, and scaling. Finally, for the last case shown in Figure 5.9c, we need to express both, the po and the ro in the coordinate system of pov. Then, we can establish the relation between po and ro as seen from pov.

5.5.4 Axiomatizing the Domestic Robot Domain

We now axiomatize the domestic robot domain based on the introduced qualitative distance and orientation relations together with their respective frames. That is, we construct a basic action theory $\mathcal{D}$ as described in Section 3.4.1.2 to perform reasoning in our domestic robot domain. We start with the required fuzzy sets. As already mentioned, we fix our distance and orientation system for the domestic robot domain. A level of granularity of 5 for the distance and a level of granularity of 3 for the orientation seems to be sufficient. With that, we can define the fuzzy sets on which our qualitative positional fluents are based. We omit the formal definition of the fuzzy sets $\mathfrak{F}(distance, u, \mu_u)$ and $\mathfrak{F}(orientation, u, \mu_u)$ that is added to $\mathcal{D}$. Instead, we show their continuous membership functions in Figure 5.10 and Figure 5.11. The formal definition of both fuzzy sets is similar to the ones for position and distance in Section 5.2.3. The difference to Figure 5.1 for the orientation relation is that we enlarged the categories for front, left, back, and right. That is because this allows for referring to the intermediate relations front-right, front-left, back-left, and back-right either by their explicit category name or by combining the categories available at a level of granularity of 2.

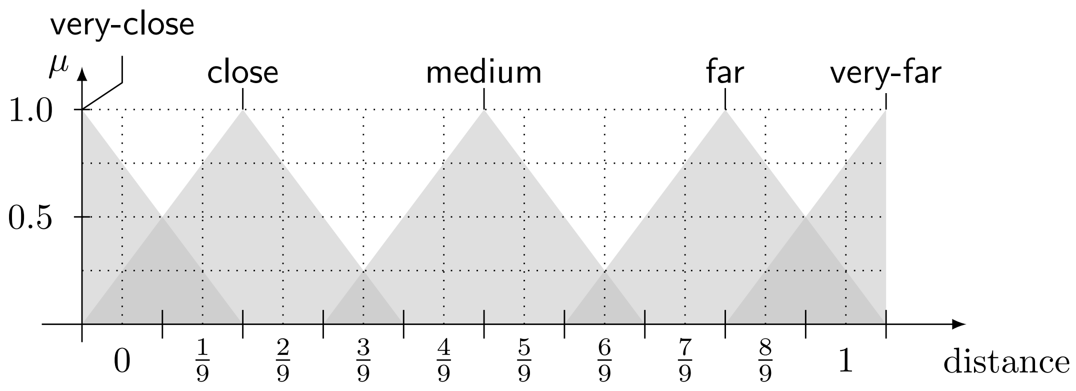


Figure 5.11: Unit membership for qualitative distance in our domestic robot domain

Unit Distance

The differences for the distance relation are less subtle. We already mentioned that the qualitative categories depend on properties captured in the frame of reference. In our formalization of positional fluents we introduced positional frames which account for these properties. To facilitate a better integration of the properties with our definition of the distance relation we now introduce a *unit distance* system. Any concrete distance system then provides an appropriate scale to instantiate the particular distance relations. In order to do this, we point to the fact that the scale of any frame of reference for the distance relation in our domestic setting can be determined. For instance, rooms feature a maximal distance of some kind that can be used to scale the farthest distance relation accordingly, with objects in the intrinsic case their size can be used to determine the length of the smallest distance interval and then to compute the maximal distance accordingly. The membership function for the unit distance now is defined for values from 0 to 1 (cf. Figure 5.11). A concrete membership function then can be achieved by dividing any distance value by the maximal distance and then referring to the unit membership function for distance. In our domestic robot domain we deal with confined spaces only. Hence, we determine the maximal distance of any such space, e.g. for a rectangular room we compute the diagonal. A distance value exceeding 1.0 after scaling to the unit distance can be mapped to the highest distance category. Inserting an additional category like "out-of-reach" would be possible also, but is not done here.

Frames

Our domain, as depicted in Figure 5.7, consists of the rooms *hallway*, *kitchen*, *parlour*, *bedroom*, and *bathroom*. Further, we have different tables, the kitchenette, the night-stands, or the coat rack, on which objects can be placed or from which objects can be taken. Each of these objects has its own local frame, which is defined w.r.t. the room where it is located in. We assume a room's origin to be at its centre. The origin of the world is also at the centre of Figure 5.7, i.e. at position $(15, 15)$. Therefore, the *parlour* has its origin at world coordinate $(4, -7)$. In the following, we give some examples for the $chfrm$ predicate, omitting a tedious and not very exciting complete axiomatization of all conversions between all room-, object- and world-frames that is added to $\mathcal{D}$.
Our world is 30×30 square units large. Therefore, $\delta^{world}_{\text{very far}} = \sqrt{2} \cdot 30 \approx 42.42$. The

parlour has a size of 22×16 square units and $\delta^{parlour}_{\text{very far}} \approx 27.20$ That means that our distance relation has to be multiplied with $m = 1.56$ to convert very-far distances from the *parlour* into the *world* frame. Therefore assuming that the parlour does not move in the world,

$$\mathit{frmparam}(\mathit{parlour}, \mathit{world}, s) = (4, -7, 0, 1.56).$$

To convert a coordinate from the parlour-frame to the world-frame we simply apply the chfrm-predicate. Then, the coordinate $(1,2)_{parlour} = (5,-5)_{world}$ as

$$(5,-5)^T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4+1 \\ -7+2 \end{pmatrix}.$$

The function chfrm can also be used for deictic relations between objects. Recall, that for a deictic relation, a separate point of view is required (see Figure 5.9). For now, we assume that the view direction is always along the positive y-axis of the frame coordinate system. With this assumption, categories such as left or right are uniquely determined. Note, however, that this assumption can be easily overcome by adding a standard view point to the definition of the frame, e.g. by a predicate $\mathit{pov}(\mathit{frame}, x, y)$, denoting that the standard point of view of frame frame is along the line between the origin of the frame coordinate system and the point (x, y).

Fluents and Actions

To keep track of the position of objects in our world, we need a number of fuzzy positional fluents, one for each object. Each fluent describes the object's position w.r.t. a frame. As we want our robot to fulfil fetch-and-carry tasks in this domain, we need actions like $\mathit{goto}(x, y)$, $\mathit{grab}(\mathit{object})$, or $\mathit{drop}(\mathit{object})$, and an action $\mathit{move}(\mathit{object}, x, y)$ to be able to model that objects in the world have been moved or carried around. The positional fluent for the kitchen table could thus be:

$$\begin{aligned}
&\mathit{pos}_{\mathit{kitchentable}}(\mathit{kitchen}, \mathit{do}(a,s)) = (x, y, \theta, m) \equiv \\
&\quad \exists x', y', \theta'.a = \mathit{move}(\mathit{kitchentable}, x', y', \theta') \land x = x' \land y = y' \land \theta = \theta' \land \\
&\qquad \exists x'', y'', \theta''.\mathit{pos}_{\mathit{kitchentable}}(\mathit{kitchen}, s) = (x'', y'', \theta'', m) \lor \\
&\quad \neg\exists x', y', \theta'.a \neq \mathit{move}(\mathit{kitchentable}, x', y', \theta') \land \\
&\qquad \mathit{pos}_{\mathit{kitchentable}}(\mathit{kitchen}, s) = (x, y, \theta, m).
\end{aligned}$$

In the initial situation, the position of the kitchen table is:

$$\mathit{pos}_{\mathit{kitchentable}}(\mathit{kitchen}, S_0) = (-1, -0.5, 1.56, 1).$$

Similarly, we can define and keep track of the position of the robot:

$$\begin{aligned}
&\mathit{pos}_{\mathit{robot}}(\mathit{world}, \mathit{do}(a,s)) = (x, y, \theta, m) \equiv \\
&\quad \exists x', y'.a = \mathit{goto}(x, y) \land x = x' \land y = y' \land \\
&\qquad \exists x'', y''.\mathit{pos}_{\mathit{robot}}(\mathit{world}, s) = (x'', y'', \theta, m) \lor \\
&\quad \neg\exists x'.y'.a = \mathit{goto}(x, y) \land \mathit{pos}_{\mathit{robot}}(\mathit{world}, s) = (x, y, \theta, m)
\end{aligned}$$

with its initial position given by $\mathit{pos}_{\mathit{robot}}(\mathit{world}, S_0) = (-2.5, -2.5, 0, 1)$. Note that in the definitions of the position of the kitchen table and the robot, we carry over the value of the scaling factor m from the previous situation. Of course, we need further fluents to describe our domain. For example, to decide whether or not we can pickup an object: picking it up is only possible if the robot does not already have something in its gripper:

$$\begin{aligned} Poss(pickup(obj), s) &\equiv \forall x.\neg holding(x, s) \text{ with} \\ holding(x, do(a, s)) &\equiv a = pickup(x) \vee a \neq drop(x) \wedge holding(x, s). \end{aligned}$$

All these fluents and all actions along with their precondition and effect axioms are part of our domain axiomatization $\mathcal{D}$.
To be able to quantify over the positions of objects in our world, we define a macro $object\_at\_pos$ as

$$\begin{aligned} & object\_at\_pos(s) = (x, y) \doteq \\ & \quad \exists x', y', \theta', m' \\ & \quad \exists f.pos_{kitchen}(f, s) = (x', y', \theta', m') \wedge chfrm(x', y', f, world, s) = (x, y) \vee \\ & \quad \exists f.pos_{kitchentable}(f, s) = (x', y', \theta', m') \wedge chfrm(x', y', f, world, s) = (x, y) \vee \cdots \end{aligned}$$

which is a disjunction over all object positions (transformed into world coordinates) in a particular situation s. This macro is true, if any object is located at the position (x, y). As our definition of $chfrm$ captures also the case where source and target frame are the same, the above definition is sound and works also in the case where $f = world$. With that we are, for instance, able to query if there is any object at position $(5, 5)$:

$$\mathcal{D} \models \exists x, y.object\_at\_pos(s) = (x, y) \wedge x = 5 \wedge y = 5.$$

Next, we define a function $dist$, which yields the distance between two coordinates and returns the value in the scale of a given frame

$$\begin{aligned} & dist(x_1, y_1, f_1, x_2, y_2, f_2, f_t, s) = d \equiv \\ & \quad \exists x'_1, y'_1.chfrm(x_1, y_1, f_1, world, s) = (x'_1, y'_1) \wedge \\ & \quad \exists x'_2, y'_2.chfrm(x_2, y_2, f_2, world, s) = (x'_2, y'_2) \wedge \\ & \quad \exists x_o, y_o, \theta_o, m_o.frmparam(world, f_t, s) = (x_o, y_o, \theta_o, m_o) \wedge \\ & \quad \exists d'.d' = \sqrt{(x'_1 - x'_2)^2 + (y'_1 - y'_2)^2} \wedge d = d' \cdot m_o, \end{aligned}$$

where f_1 is the frame of the first coordinate, f_2 is the frame of the second coordinate, and f_t is the target frame; and a function ori that yields the angle between two objects:

$$\begin{aligned} & ori(x_1, y_1, f_1, x_2, y_2, f_2, f_t, s) = \theta \equiv \\ & \quad \exists x'_1, y'_1.chfrm(x_1, y_1, f_1, f_t, s) = (x'_1, y'_1) \wedge \\ & \quad \exists x'_2, y'_2.chfrm(x_2, y_2, f_2, f_t, s) = (x'_2, y'_2) \wedge \\ & \quad \exists \theta_1. \arctan(y'_1/x'_1) = \theta_1 \wedge \exists \theta_2. \arctan(y'_2/x'_2) = \theta_2 \wedge \theta = \theta_1 - \theta_2. \end{aligned}$$

Now, we have everything in place in our domain theory $\mathcal{D}$ to fetch an object in our domestic world.

5.5.5 Fetching a Cup: A Domestic Robot Example

Next, we want to instruct our robot for a fetch-and-carry task. An example scenario is shown in Figure 5.7. Suppose the following situation. The user is sitting on the couch in her parlour, watching TV. The domestic service robotic servant is quietly staying aside waiting for instructions. At some point, the instructor is commanding: "*Robot, get me my cup. It is left on the kitchentable, close to the plate.*" Having a closer look at this instruction, the following objects and frames are referred to with it:

$$\text{Robot, get me } \underbrace{\textit{my cup.}}_{po} \text{ It is } \underbrace{\textit{left}}_{ori} \text{ on the } \underbrace{\textit{kitchentable}}_{frame}\text{, } \underbrace{\textit{close}}_{dist} \text{ to the } \underbrace{\textit{plate}}_{ro}.$$

The first part of the instruction is a deictic hint. As shown in Figure 5.9a, the position of the primary object *cup* is dependent on a reference object ro and a point of view pov. As we mentioned earlier, both the ro and the pov are given by a frame, the kitchentable, in this case. That means that the ro is the origin of the kitchen table's coordinate system, the pov is along the y-axis. The second part is an example for the intrinsic case. Here the distance relation depends on the reference object *plate* and close depends on the size of the reference object ro. The correct scaling is defined in the frame parameter of "plate". Assume, that our natural language processing (NLP) software is capable to extract the above marked information.[5] The hints of the human instructor tell us something about the object we seek and with the domain axiomatization $\mathcal{D}$ sketched in the previous section can be formalized as follows:

$$\begin{aligned}\mathcal{D} \models\ & \exists x_1, y_1 . object\_at\_pos(s) = (x_1, y_1) \wedge \\ & \exists x_2, y_2, \theta_2, m_2 . pos_{kitchentable}(kitchen, s) = (x_2, y_2, \theta_2, m_2) \wedge \\ & \text{is}(ori(x_1, y_1, world, x_2, y_2, kitchen, kitchentable, s), \mathsf{left}) \wedge \\ & \exists x_3, y_3, \theta_3, m_3 . pos_{plate}(kitchentable, s) = (x_3, y_3, \theta_3, m_3) \wedge \\ & \text{is}(dist(x_1, y_1, world, x_3, y_3, plate, kitchentable, s), \mathsf{close})\end{aligned} \tag{5.2}$$

That means that we are looking for an object with the coordinates (x_1, y_1) that is on the left side of the table and that is, measured in the scale of the plate, close to the plate. The only object that meets these conditions is cup_A. A plan that our robot should therefore make up for this case should be something similar to this:

$$\begin{gathered} do([goto(kitchen), approach(kitchentable), pickup(cup_A), \\ goto(parlour), approach(human), drop(cup_A)], S_0). \end{gathered}$$

Planning to Fetch the Cup

The required information is: the kitchentable is in the kitchen, the robot has to conclude that the object left close to the plate is cup A, the position of the human needs to be known, and cup A needs to be dropped without spilling etc. Of course, a sophisticated robot controller is required to execute this simple-looking plan on a real robot in the

[5]In fact, it should not be too hard for the NLP component installed on our @Home robot, to extract these information. In Section 6.2 we present an approach to language processing that we could also use to parse and interpret statements with qualitative positional information.

Algorithm 9: A Readylog program making use of the qualitative positional notions for the "*Fetch&Carry*" task. We omit some specification details to retain reasonable clarity.

```
proc fetch_and_carry($f, x_t, y_t, f_t$)
  $(\pi\ x, y, \theta, m, x_1, y_1, x_2, y_2)[pos_f = (x, y, \theta, m)?$; approach$(x, y)$];
  if $\exists x, y.\Phi(x, y)$ then
    $(\pi\ x', y', \theta', m', obj)[\Phi(x', y') \wedge pos_{obj} = (x', y', \theta', m')?$;
      pickup($obj$) ;
      approach($x_t, y_t, f_t$) ;
      drop($obj$)]
  else
    fail
  endif
endproc

proc approach($x, y, frame$)
  if $\exists x_1, y_1$, is($dist(x, y, frame, x_1, y_1, frame, frame)$, close) then
    $(\pi\ x_1, y_1)$[is($dist(x, y, frame, x_1, y_1, frame, frame)$, close; goto($x_1, y_1$)];
  else
    fail
  endif
endproc
```

real world. In the following, we abstract from many of the complications that arise during the execution. That is because we want to concentrate on planning the above action sequence in an abstract yet flexible way, thereby making use of the integrated qualitative spatial representations and reasoning facilities. Having all the ingredients such as the position of all mentioned objects and their frames, the control program shown in Algorithm 9 is doing the job.
From the NLP system we expect to get a set $h_{ori} = \{(ro_1, f_1, c_1), \ldots, (ro_k, f_k, c_k)\}$ with statements qualifying the orientation between the primary object po and a reference object ro_i together with a reference frame f_i and the qualitative orientation category c_i describing the relation between po and ro_i. Similarly for distance information, we get a set $h_{dist} = \{(ro_{k+1}, f_{k+1}, c_{k+1}), \ldots, (ro_n, f_n, c_n)\}$. From these sets, we construct a macro $\Phi(x, y)$ as:

$$\begin{aligned}
\Phi(x,y) \doteq\ & object\_at\_pos(s) = (x,y) \wedge \\
& \exists x_1, y_1, \theta_1, m_1.pos_{ro_1}(f_1, s) = (x_1, y_1, \theta_1, m_1) \wedge \\
& \quad \text{is}(ori(x, y, world, x_1, y_1, f_1, f_t, s), c_1) \wedge \cdots \wedge \\
& \exists x_k, y_k, \theta_k, m_k.pos_{ro_k}(f_k, s) = (x_k, y_k, \theta_k, m_k) \wedge \\
& \quad \text{is}(ori(x, y, world, x_k, y_k, f_k, f_t, s), c_k) \wedge \\
& \exists x_{k+1}, y_{k+1}, \theta_{k+1}, m_{k+1}.pos_{ro_{k+1}}(f_{k+1}, s) = (x_{k+1}, y_{k+1}, \theta_{k+1}, m_{k+1}) \wedge \\
& \quad \text{is}(dist(x, y, world, x_{k+1}, y_{k+1}, f_{k+1}, f_t), c_{k+1}) \wedge \cdots \wedge \\
& \exists x_n, y_n, \theta_n, m_n.pos_{ro_n}(f_n, s) = (x_n, y_n, \theta_n, m_n) \wedge \\
& \quad \text{is}(dist(x, y, world, x_n, y_n, f_n, f_t, s), c_n)
\end{aligned}$$

For our example these sets would be $h_{ori} = \{kitchentable, kitchen, \mathsf{left}\}$ and $h_{dist} = \{plate, plate, \mathsf{close}\}$ with the target frame $f_t = kitchentable$ (abusing notation slightly). It should be obvious that $\mathcal{D} \models \exists x, y.\Phi(x, y)$ yields Equation 5.2.

Executing the Program

Now let us showcase how the execution of the program given in Algorithm 9 leads to a successful retrieval of the cup as desired. We can assume that the robot has initial knowledge about the positions of rooms and furniture therein. In terms of our specification of *frames* it would comprise the following sentences:

- $frmparam(kitchen, world, s) = (-8.0, 8.5, 0, 2.23)$
- $frmparam(kitchentable, kitchen, s) = (-1.5, -0.5, 0, 3.17)$
- $frmparam(kitchentable, world, s) = (-9.5, 8.0, 0, 7.07)$

Firstly, the robot retrieves the position of the frame mentioned in the user's request, namely the *kitchentable*. It then moves to that very frame using the method approach. Note that we are using READYLOG's features here to pick a position close to the frame already deploying a first use of the qualitative notions established in this paper.
Then, the robot picks positions[6] that meet the hints given in the initial request ensuring to satisfy $\Phi(x, y)$. As laid out before, Φ contains the structured information that our NLP could retrieve from the user utterance as given in Equation 5.2. Evaluating different objects, cup_A is the only one that satisfies the given specifications as follows.
The predicate $\mathrm{is}(ori(x_1, y_1, world, x_2, y_2, kitchen, kitchentable, s), \mathsf{left})$ can only be satisfied for cup_A with $pos_{cup_A}(world) = (-10.5, 7.5)$ and $pos_{kitchentable}(kitchen) = (-9.5, 8.0)$[7] since $ori(-10.5, 7.5, world, -9.5, 8.0, kitchen, kitchentable, s)$ is then computed as $atan2(-0.5, -1.0) = -2.68$ and $\mathfrak{F}(\mathsf{left}, -2.68, 0.65) \wedge 0.65 > 0$ indicates that this angle does in fact belong to the category left.[8] Analogously, to satisfy the predicate

$$\mathrm{is}(dist(x_1, y_1, world, x_3, y_3, plate, kitchentable, s), \mathsf{close})$$

extracted from the distance hint the robot computes the distance for cup_A with $pos_{cup_A}(world) = (-10.5, 7.5)$ and $pos_{plate}(plate) = (0.0, 0.0)$ as $dist(-1, 1, kitchentable, -1, -0.5, kitchentable, plate, s)$ resulting in 1.5 which, after being normalized with the scaling factor for the plate results in $\mathfrak{F}(\mathsf{close}, 0.23, 0.75) \wedge 0.75 > 0$ which in turn indicates that the distance between the cup and the plate does belong to the category close (w.r.t. the plate's size).
We give an illustration of applying the qualitative categories while respecting the corresponding frames both for the orientation and the distance hint in Figure 5.12.
Since cup_A is the only object satisfying both the conditions extracted as hints from the user utterance the robot continues executing the program with cup_A as the object to grab and to deliver to the target position. Hence, by following the program from Algorithm 9 the robot eventually comes up with the action sequence

$$do([goto(-9.5, 5.0), pickup(cup_A), goto(-2.0, -8.0), drop(cup_A)], S_0).$$

[6]Recall that π stands for a non-deterministic choice of arguments.
[7]For sake of readability, we leave out the angle and the scaling factor in the positional fluents.
[8]Note that we have formally defined the entries of the membership function only for integer values. However, we assume here that real values are possible also. This is true for our implementation anyways.

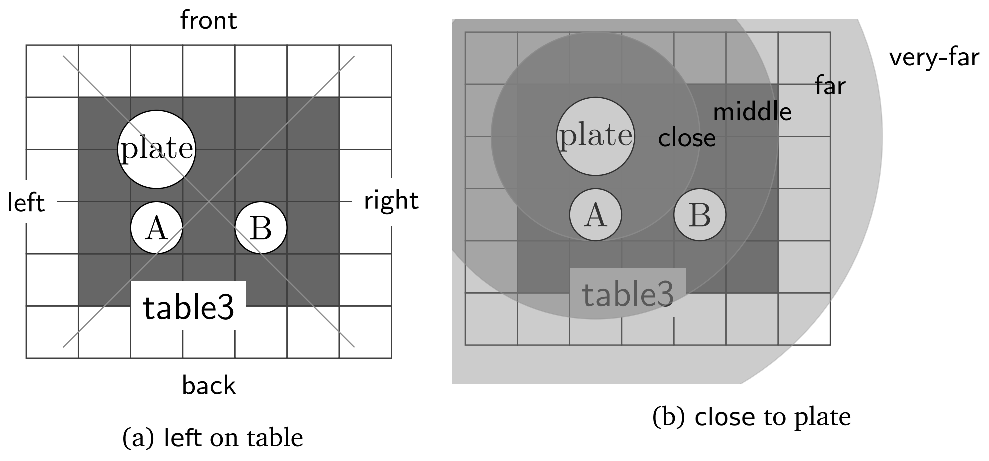


Figure 5.12: Details on the coordinate transformation to answer the request in the domestic robot domain

as claimed initially. It is able to do so satisfying the qualitative spatial description given as hints by the user. This is due to the use of the underlying mechanics that we defined and formally introduced to the situation calculus.

5.5.6 Summary

In this section, we showed how qualitative spatial reasoning about positional information in a domestic environment can be conducted. For the domestic environment, we extended our qualitative notions from Section 5.2 for spatial relations to account for different contexts in indoor environments. Our extended notion allows for coping, for example, with the fact that different objects have different sizes and that “far” w.r.t. a large room has a different quantitative scale than “far” on the kitchen table. To do so, we introduced the concept of frames and defined it formally. The *frame* concept is needed to capture all relevant positional information of domain objects. Otherwise it would be impossible to evaluate spatial relations in the right context. This lead to the notions of fuzzy positional fluents that always carry their contextual frame with them. We presented a high-level controller in the robot programming and plan language READYLOG for a domestic fetch-and-carry task. The controller shows how easy it is to integrate the formal notions for positional information in our high-level controllers programmed in READYLOG. It also indicates that the extension for reasoning with qualitative positional information constitutes a proper theoretical foundation for controllers dealing with qualitative positional information that frequently appear in real world domestic settings and that it can be deployed beneficially on a real robot in such scenarios.

5.6 Summary and Discussion

In this chapter we presented a set of approaches to deal with qualitative information frequently used by humans. Our motivation was to bridge the gap in the difference between the linguistic notions used by humans and the numerical information required by robots, both, to facilitate human-robot interaction, but also to ease the specification of robot controllers. The former makes robot accessible for the laymen they need to operate the robot in our intended target scenario of domestic service robotics. The latter accounts for the fact that the knowledge of domain experts can often be formulated in human terms very easily but is sometimes hard to translate to robot terms.

5.6.1 Summary

The basic idea is to combine the situation calculus and qualitative representations based on fuzzy sets. We represent the different qualitative categories we want to reason about as fuzzy sets. This is appealing as it is possible that an object falls into several categories at the same time. Fuzzy set theory gives us a formal account for that. For our reasoning engine, we use the situation calculus as a powerful calculus to reason about action and change. We embed fuzzy sets into the situation calculus. To this end, we introduce linguistic categories and fuzzy fluents, which are special functional fluents that can take qualitative, linguistic terms as function values. These linguistic terms are evaluated based on fuzzy sets and related to quantitative representations with a special defuzzifier function, the centre-of-gravity defuzzifier in our case. The defuzzifier is also used to allow for assigning fuzzy values to qualitative fluents.

After enabling the use of qualitative notions in the world model of a robot we also integrated fuzzy controllers into Golog. We introduced program statements to specify a rule base and to perform inference on such a rule base. With a fuzzy controller it is often possible to capture the intended behaviour for a task that requires reactive behaviour with some simple if-then rules. The possibility to use such qualitative control laws in GOLOG increases its expressiveness and increases the possibilities to tackle robotic high-level problems.

We applied the two extensions in a real-world setting taken from the ROBOCUP@HOME domain. For fetch-an-carry tasks, we facilitated the human-robot interaction because in domestic applications such as ROBOCUP@HOME the robot needs to be instructed by a human operator by natural language. Our integration of qualitative fluents allows for seamlessly integrating such notions in the world model and in the high-level control of our domestic service robot. For tasks like follow-me the intended behaviour of the robot can be formulated in few simple rules with linguistic terms. This is supported by our integration of fuzzy controllers in Golog.

Applying the two extensions to real-world scenarios revealed that spatial information play a crucial role in the domestic domain. That is why presented an extension of qualitative notions particularly fitted for representing and reasoning with positional information. To this end, we reiterated an approach to qualitative positional information and extended fuzzy fluents to fuzzy positional fluents. Those positional fluents carry with them a so-called frame that accounts for contextual information that is needed to be able to interpret the meaning of qualitative positional references correctly. With a unified approach to transform positional information from one frame to another we yield a powerful framework for reasoning about positional references within our high-level control. This was shown in an extended real-world scenario taken from the domestic service robotics domain.

5.6.2 Discussion

Although the extensions of our high-level control presented in this chapter contribute very much towards making our robot more interactive and more fitted for domestic service robotics tasks, some issues remain open or at least leave room for improvement. The assumption for the domestic environments we make in Section 5.5 is that it is sufficient to have one fixed unit distance and orientation relation for all the different contexts. Of course, we adopt our distance relation, for instance, according to the size of a room, but still the number of categories remains fixed. It would be interesting, though, to drop this assumption and learn which categories are required in what context by interacting with a human instructor, such as in (Robinson, 2000), where fuzzy distances between cities are learnt in a GIS context. Looking at the table scenario, the instruction that is actually meant by the human instructor is more like "*cup A is closer to the plate than cup B*" or "*cup A is closer to the plate than any other object*". Fuzzy logic also offers this kind of reasoning facilities. For example, as we pointed out in Section 3.2.1.1, the compositional rule is able to reason that *X is much larger* ∘ *large* if *X is much larger than Y* and *Y is large*. Together with so-called hedges (a kind of intensifier of a fuzzy category), we could draw the required conclusions. We proposed a situation calculus semantics for hedges in Section 5.3. This needs to be integrated into READYLOG further to be able to reason about such kind of human instruction, as well.

Our example also reveals another interesting problem that we want to address in the future. When approaching the table, we were assuming that we could pick up the target object from the front side of the table. This is, in general, not true. In some situations the robot might need to approach the table from a different angle. Then, the qualitative positions and orientations of primary and reference objects to each other are changing. Our framework, however, is able to keep track of the positions. Together with READYLOG's capabilities to perform decision-theoretic planning making use of an optimization theory, we could plan the optimal approach angle to grab the object in question.

It would also be interesting to integrate further human-machine interaction into our robot controller. If, for example, the robot is not able to detect the object in question based on the hints that the human instructor is giving, it could start a dialogue with the human to get more evidence about the object. In our controller representation in Algorithm 9, such a dialogue system could be used in the failure cases in lines 9 and 16 of the robot controller. For a realization of a similar idea in the context of interpreting human commands with taking into account human fallibility we refer to Section 6.2.

6 Robustness and Flexibility

If we deploy robots in human environments, where they operate with laymen, usually over extended periods of time, those robots should ideally feature some kind of resilience against failure. More precisely, we would like our robots to be **robust** against and take care of internal failures, something we refer to as *self-maintenance* in the following. They should further be **flexible** by taking into account and by handling human fallibility. In this chapter we discuss two extension of our high-level robot controller in this regard. First, we attend to internal faults of the robot with the goal of increasing robustness against such faults. To this end, we equip our robot with a form of self-maintenance that enables it to handle a certain class of errors by itself. Second, we increase the robot's flexibility in dealing with external failures, specifically with errors that occur in human-robot interaction by taking into account and handling human fallibility.

In Section 6.1 we present an approach to self-maintenance. In order to make a robot execute a given task plan more robustly we want to enable it to take care of its self-maintenance requirements during online execution of this plan. This requires the robot to know about the (internal) states of its components, constraints that restrict execution of actions and how to recover from faulty situations. The general idea is to implement a transformation process on the plans, which are specified in our agent programming language READYLOG, to be performed based on explicit qualitative temporal constraints. Afterwards, a 'guarded' execution of the transformed program results in more robust behaviour.

In Section 6.2 we propose a system for robust and flexible command interpretation for our domestic service robot. Existing language processing for instructing a mobile robot often make use of a simple, restricted grammar where precisely pre-defined utterances are directly mapped to system calls. In fact, the first attempt for the speech recognition of our domestic service robot as described in Section 4.1 is using such a restricted grammar. This does not take into account fallibility of human users and only allows for binary processing; either a command is part of the grammar and hence understood correctly, or it is not part of the grammar and gets rejected. We now model the language processing as an interpretation process where the utterance needs to be mapped to one of the capabilities available on the robot. We do so by casting the processing as a (decision-theoretic) planning problem on interpretation actions. This allows for a flexible system that can resolve ambiguities and which is also capable of initiating steps to achieve clarification.

6.1 Self-Maintenance

To arrive at truly intelligent robotic agents, many issues need to be addressed such as perception, locomotion, manipulation, human-robot interaction, planning and reasoning. Two research question in the field of *cognitive robotics* are “what does the robot need to know about its environment” and “when should the inner workings of an action be available to the robot for reasoning”. In this section we approach a specialization of the intersection of these two questions, namely “what does the robot need to know about itself and its requirements”. Here, requirements refer to dependencies between actions that the robot performs in the real world and its internal configuration such as the software setup. This is especially interesting as present agents are often unable to explicate their requirements (e.g. calibration of manipulators before usage) relative to a plan. They usually need these requirements to be considered in an external process and in advance, otherwise they fail during plan execution.
We propose a *constraint-based self-maintenance* framework, which enables an agent to monitor its self-maintenance requirements during program execution. Whenever the framework determines unsatisfied requirements, appropriate recovery measures are performed online. This behaviour increases agent autonomy and robustness. We achieve this by adding a program transformation step in our high-level control language READYLOG. This program transformation uses explicitly formulated constraints that express dependencies between actions on the task level and the robot itself. These dependencies are only important at run-time and we cannot and do not want to consider them at planning time already. Thus, by decoupling the run-time requirements, we also alleviate the costs for planning.

6.1.1 Temporal Constraints

The dependencies we need to model can be captured with temporal constraints. To formulate temporal constraints between actions we obviously require a notion of temporal relations between actions (or more generally, between states). Since we are interested in constraints that should be easy to formulate for the designer we prefer a qualitative description of these relations. We consider this sufficient for most cases we intend to handle and spare computing explicit timing values. We therefore chose Allen’s Interval Algebra (Allen, 1983) as our basis. For an overview on important relations in this algebra see Figure 6.1. An example of a constraint that we want to formulate could be

$$calibrate\_arm \text{ BEFORE } manipulate$$

to indicate that the manipulator has to be calibrated before we can actually use it. We are not the first to consider an interval formulation in GOLOG (Finzi and Pirri, 2004). However, our approach is not targeted at actually performing interval planning. Instead, we use intervals only to formulate the constraints and augment a given program accordingly.

6.1.2 Durative Actions in the Situation Calculus

Usually, actions are durative, i.e., they consume time. The original situation calculus only knows *instantaneous* actions. There are, however, some extensions that we are

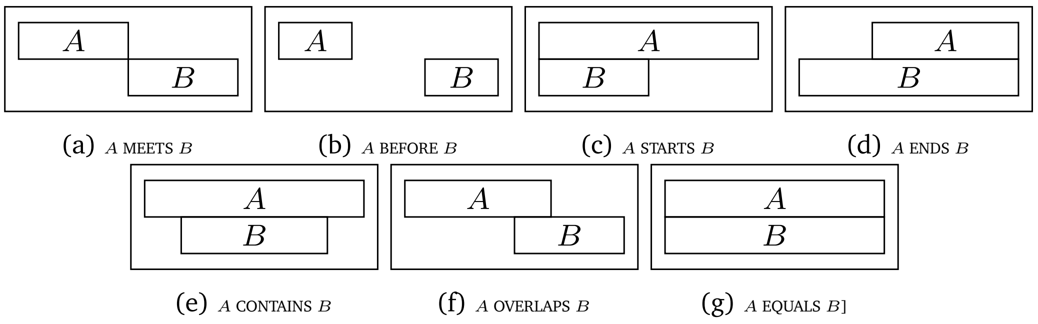


Figure 6.1: Seven of the temporal interval relations mentioned in Allen (1983). We omit the inverse relations here for simplicity.

going to adopt to represent durative actions (De Giacomo et al., 2000; Claßen et al., 2007). In these approaches, actions with a duration are considered *activities* that are bounded by *instantaneous* start/stop-actions. The fact that such an activity is currently being performed is indicated by a fluent for each activity. See Figure 6.2 for an example.

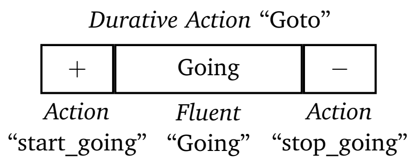


Figure 6.2: Exemplary decomposition of a durative action

6.1.3 Constraint-based Program Transformation

The general idea for our self-maintenance is to implement a program transformation process based on temporal constraints and the program to be performed. We assume here that READYLOG is used to specify our agents, hence our approach is an extension to READYLOG. As programs in READYLOG represent task plans, we will use the term *program* from now on instead of *plan*. Figure 6.3 depicts how we propose to integrate the components of our self-maintenance framework into the existing agent controller.

We propose to intervene between decision-theoretic planning, whose output is an executable program, and the online execution of this program. Before any action of the program is performed in the real world, a self-maintenance interpreter checks whether there are unsatisfied constraints for this action. If such constraints are found, the program execution is delayed and the program is augmented with maintenance and recovery measures. The augmented program is only then passed on for execution. Depending on the constraint(s) the transformation also includes monitoring markers, e.g., making sure the *locomotion_module is off* throughout the execution of *manipulating* something. It possibly also requires a list of commitments to actions in the future. we detail this in Section 6.1.3.2.

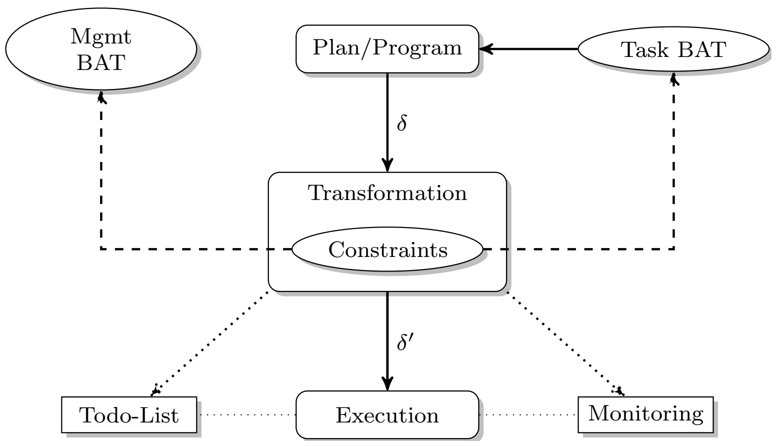


Figure 6.3: Architectural overview of our approach. The program δ, which only uses the *Basic Action Theory* for the task, is passed to our transformation process. This process uses constraints, which link elements from the Task BAT to elements from the self-maintenance domain. The latter are specified in the Mgmt BAT. The transformation yields a modified program δ' which is passed on for execution together with instructions for monitoring and commitments to future action. These commitments are maintained in a todo-list to ensure that they will eventually be satisfied.

6.1.3.1 Constraints

For the proposed transformation to work, we need to make one important restriction, though. Since the self-management may not invalidate the program, we separate the task from the maintenance domain and restrict the constraints to only map elements from the former to the latter. The invalidation could be caused because the transformation might insert actions and inserting task level actions could render effects intended in the program void.
Our approach is similar to a work from Lemai and Ingrand (2004) who propose a framework for online plan repair and execution control based on temporal constraints. Their motivation are problems similar to ours, namely that “taking into account run-time failures and timeouts” requires online plan recovery. However, they rely on partial order planning and assume temporal flexible plans such that “postponing and inserting actions” at run-time is possible. Their objective is to execute a plan under resource and timing constraints by grounding time points at execution time. We, on the other hand, are interested in interleaving self-maintenance and task actions at execution time on a qualitative level. Time points in the situation calculus are only characterized by actions. Still, we borrow their notion of (non-)preemptive actions and the idea that the execution system sends some form of report about action completion and the components’ states in the system.
Our constraint syntax is $\mathcal{A} \otimes \mathcal{B}$ where

$\mathcal{A}$ is from the task domain. It can be (a) an *instantaneous action* which corresponds to an interval end-point, (b) a *durative action* that needs to be decomposed to its end-points, or (c) a *fluent formula* that needs to be checked with respect to the interval.

$\otimes$ is one of Allen’s relations.

Table 6.1: Translation of a constraint to an order on situations

$\mathcal{A}$ BEFORE $\mathcal{B}$		Mgmt ($\mathcal{B}$)		
		b	B	ψ
Task ($\mathcal{A}$)	a	$a < b$	$a < B^+$	$a < \Delta_\psi^+$
	A	$A^- < b$	$A^- < B^+$	$A^- < \Delta_\psi^+$
	ϕ	$\Delta_\phi^- < b$	$\Delta_\phi^- < B^+$	$\Delta_\phi^- < \Delta_\psi^+$

$\mathcal{B}$ is from the self-management domain. The same cases as described above for $\mathcal{A}$ also apply to $\mathcal{B}$.

Agent Components

In the maintenance domain we model components of the robot and the means that the robot has to influence the status of those components. Instead of directly encoding this in the basic action theory of the maintenance domain, we introduce a special model to do so based on *deterministic finite state automata* (DSFA). Roughly following ideas from (Lespérance et al., 2000), (Finzi and Orlandini, 2005), and (Tam, 1998), we use state machines to describe the status that a robot component can have, e.g. the motor can be *idle*, and the transitions between status as actions, some of which the robot can take as part of its maintenance, for instance, the *shutdown* would change the status of the laser component to *offline*. We then have a mechanism in place that translates our component model into the basic actions theory. Using the state machine based model, for one, yields a more natural formulation for describing the components of the robot, and for another, it allows to derive the maintenance action needed to reach a certain status more easily. Also, the required status of a component can be used in specifying the constraints. Since we model is of no particular importance for the remainder of this section we omit further details for the sake of clarity.

6.1.3.2 Online Program Transformation

We transform the program (i.e. a plan generated by READYLOG) using the set of constraints available for the next task action to be executed. The set of constraints is translated to a *Constraint Satisfaction Problem* (CSP) by resolving each constraint to an order on situations described by primitive or start/stop-actions. An example is given in Table 6.1. Small case letters denote instantaneous actions, capital letters stand for complex actions, and Δ_ϕ^- and Δ_ϕ^+ represent a fluent formula ϕ becoming false or true in a certain situation, respectively. The solution of the CSP then dictates the transformation. It schedules maintenance actions and augments the program with appropriate so-called *occurrence commitments* and *persistence commitments*, i.e. commitments to future situations. Before we detail the construction of the CSP, we first elaborate on the concept of commitments mentioned above.

Commitments

There are maintenance constraints that require a commitment to future points in time. Consider, for example, the constraint A BEFORE B, which denotes that B has to be performed sometime after B or the constraint A EQUALS B, which, amongst others, demands that B has to stay active until A terminates. The only way to handle such commitments so far is to introduce auxiliary fluents that record these commitments. To determine the value of these fluents we would need to use regression (Pirri and Reiter, 1999), which traces the fluent's value back to the initial situation and applies all changes made since then up to the current situation. As regression is quite expensive with an increasing number of actions performed already, we follow a different approach. We extend the READYLOG language with two sets of conceptually different commitments: (i) *occurrence commitments*, which denote that some action has to be performed in the future, and (ii) *persistence commitments*, which denote that some situation calculus expression holds for some time span, e.g. for a subsequence of a situation term.

Occurrence Commitments Occurrence commitments $\mathcal{O}$ are commitments about the future program and ought to ensure that a certain action happens in the future and under certain circumstances. We consider four forms of occurrence commitments which represent the different circumstances under which we may require an action to be performed in the future.

- *sometime* (β) denotes that action β has to be performed sometime in the future,
- *after* (α, β) denotes that β has to be performed sometime in the future, but not before the next occurrence of action α.
- *within* (β, N) denotes that β has to be performed within the next N actions.
- *then* (α, β, N) denotes that β has to be performed within the next N actions, after the next occurrence of α.

The self-maintenance interpreter updates $\mathcal{O}$ according to the constraints involved. In any future step, it also needs to be notified about the existence of actions to be integrated. We therefore introduce a new construct TODO (α) to the READYLOG language which causes both, updating $\mathcal{O}$ according to the last action α performed and managing to check and schedule an action from $\mathcal{O}$ if applicable.

Persistence Commitments Persistence commitments $\mathcal{P}$ denote that some situation calculus expression needs to hold for some time span. These commitments follow similar ideas as the occurrence commitments, i.e., a set of situation calculus expressions should hold under certain circumstances in the future. As satisfying these commitments is considered vital to program execution, the self-maintenance interpreter may terminate the program if such a persistence commitment is violated.

- *between* (α, β, φ) denotes that the expression φ has to hold between the next occurrence of α and the first subsequent occurrence of β.
- *until* (β, φ) denotes that φ has to hold until the next occurrence of β.

The corresponding construct MONITOR follows the same idea as the TODO construct above, i.e., the persistence commitments are updated relative to the last action performed and then the resulting persistence commitments are checked relative to the current situation. We introduce two auxiliary relations UPDATEMONITOR $(\alpha, \mathcal{P}, \mathcal{P}')$ and CHECKMONITOR $(s, \delta', \mathcal{P})$ which perform these tasks.

Constructing the CSP If, at any point in program execution, we have more than one constraint associated with the next action α, we have to schedule the corresponding maintenance actions appropriately. We do so by using transformation rules which we present in the next section.
The constraint satisfaction problem that we construct is a fixed length CSP. It is built over variables $\{t_{\alpha_1}, \ldots, t_{\alpha}, \ldots, t_{\alpha_k}\}$, where each t_{α_i} represents a time-step in the sequence of actions induced by the maintenance constraints. The number of variables, i.e., the length of the CSP, is inferred from the transformation rules. For every action that needs to be scheduled we need one time-step, hence one variable, and we need one time-step for the task action α itself. If there are occurrence commitments of types *within* (β, N) or *then* (α, β, N), these are also added to the CSP by adding additional variables t_{α_j} for each action required in any of these commitments. The domain of all these variables are the natural numbers $[1, 2, \ldots, n]$, where n is the length of the CSP, namely, the number of actions to be scheduled around the task action α plus one (for α itself). The restrictions on the variable assignments are taken from the transformation rules and the commitments. This completes the construction of the CSP. The solution to the CSP is an assignment of values to the variables t_{α_i}. It represents an ordering on the actions to be scheduled and it can then be translated to the sequence of actions $[\alpha_{i_1}, \ldots, \alpha_{i_n}]$.

Transformation Rules The illustration of our maintenance constraints in Table 6.1 was a slight simplification. As already mentioned, we actually use a set of transformation rules per constraint. A transformation rule is a six-tuple $(A \otimes B, \alpha, \Phi, \mathcal{C}_{CSP}, \mathcal{O}, \mathcal{P})$, where

- $A \otimes B$ is a (maintenance) constraint as described in Section 6.1.3.1
- $\alpha \in A^P$, i.e., an action from the task domain
- Φ is a situation calculus formula qualifying the scope of the transformation rule, e.g., apply different rules depending on whether a fluent F holds or not.
- $\mathcal{C}_{CSP}$ is a multi-set of restrictions of the form $T \times T \times \Psi$, where T is the set of CSP variables and Ψ is an arithmetical or logical expression on these variables. For example, $(t_i, t_j, \Psi_{i,j})$ denotes that t_i and t_j are constraint by the expression $\Psi_{i,j}$.
- $\mathcal{O}$ is a set of occurrence commitments
- $\mathcal{P}$ is a set of persistence commitments

The transformation rules have to be read as follows. If there is a maintenance constraint associated with the next task action α to be executed in situation s, and the condition Φ holds, then (1) the restrictions $\mathcal{C}_{CSP}$ have to be met in the CSP that is built, and (2) the CSP has to also meet both, the occurrence commitments $\mathcal{O}$ and the persistence commitments $\mathcal{P}$. That is to say, every set of pairwise restrictions on variable assignments $\{(t_i, t_j) \mid \Gamma_{i,j}\}$ in the final CSP is made up of the Ψ in $(t_i, t_j, \Psi_{i,j})$ from $\mathcal{C}_{CSP}$. As an example, consider the constraint A MEETS B with transformation rule $(A \text{ Meets } B, A^{-}, \text{True}, \{(t_{A^-}, x_{B^+}, t_{A^-} \approx_{\varepsilon} t_{B^+})\}, \emptyset, \emptyset)$. If the next action to be performed is A^-, and the formula Φ holds in that situation, this rule causes the variables t_{A^-}, t_{B^+} to be added to the CSP with the restriction $t_{A^-} \approx_{\varepsilon} t_{B^+}$ on their ordering. It also causes the occurrence commitments to be updated with $\mathcal{O}$, and the persistence commitments to be updated with $\mathcal{P}$. Since both $\mathcal{O}$ and $\mathcal{P}$ are empty here, no new commitments are added.

6.1.3.3 Extended Configurations

The transition semantics mentioned in Section 3.4.1 so far described transitions over configurations of the form (δ, s). This does not allow iterative modifications of our commitments $\mathcal{O}$ and $\mathcal{P}$ without introducing procedural notions of variable assignments. This seems undesirable. Therefore, we extend the transition semantics of READYLOG from configurations (δ, s) to configurations $(\delta, s, \mathcal{O}, \mathcal{P})$ which include both, occurrence commitments $\mathcal{O}$ and persistence commitments $\mathcal{P}$. Then, in each transition, i.e. in one step of the interpreter, we can update not only the program and the situation but the commitments needed for successful self-maintenance as well.

6.1.3.4 Special Features

Our approach features two peculiarities, namely (non-) preemptive actions and constraint inheritance, which we elaborate on in the following.

(Non-)Preemptive Actions By introducing a decomposition of complex actions (i.e., activities) into an action starting the activity and an action stopping it, we skipped the fact that some of the actions that an agent can take may not be terminated by the agent itself in a meaningful way. Following Lemai and Ingrand (2004), we use their notion of *preemptive* and *non-preemptive* actions, i.e., we distinguish between actions which can be terminated at any time without ruining the outcome and actions which are expected to fail if terminated early. We will also borrow the idea that the underlying execution system sends some form of report about the end of *non-preemptive* actions.
Within a coffee delivery scenario we might consider a self-maintenance action *beep*, that makes an annoying sound while the agent transports coffee, to be preemptive, while the task action *pickup* clearly is not. Since we want to treat both types of action similarly during program transformation, we introduce a construct $\mathrm{waitForEnd}(A, \tau)$. It takes a complex action A and a deadline τ and terminates the action A if it is preemptive. If the action is non-preemptive, it blocks the program execution until either the execution system signals that the action has finished or the deadline is met, at which point program execution is aborted.

Inheritance It is an often seen bad practice to duplicate constraints for related actions. To alleviate this and provide a more convenient way of formulating the constraints, it should be possible to define constraints for classes of actions, e.g., we would like to have an action inheritance for several move actions. Building on work by Gu and Soutchanski (2008), we employ a modular BAT that allows for inheritance of constraints along the hierarchy of actions. See Figure 6.4 for an example.

6.1.3.5 Concurrency

The transformation, and hence the execution, may require several actions to be running concurrently. With our notion of durative actions this is possible, but the maintenance may even require two or more instantaneous actions to be executed simultaneously. READYLOG, however, currently only supports *interleaved concurrency* (De Giacomo et al., 2000), which executes two action sequences concurrently by interleaving them. This

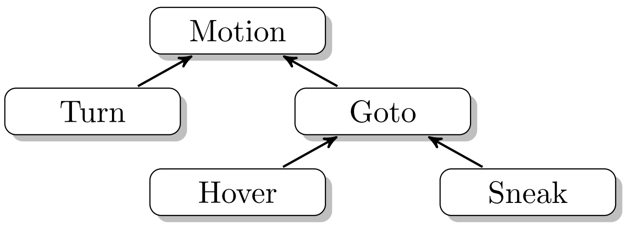


Figure 6.4: Inheritance of constraints in a hierarchy of actions

is opposed to *true concurrency* (Reiter, 1996), where sets of actions may be executed 'truly concurrently' between any two situations. To deal with this, we make use of something we call the ε-slot. Since, with a real execution system, concurrent calls to execute something will be serialized eventually, we consider two (or more) actions happening simultaneously if they happen within a time span not exceeding the ε-slot. ε is a natural number that denotes, how many subsequent instantaneous actions we think can be considered to happen simultaneously.

6.1.4 A Detailed Example

To illustrate our approach we will now step through a detailed example. Consider the following setup: Given a basic action theory that contains at least

- the fluent *Calibrated* indicating whether the robot's laser range finder has already been calibrated,
- the complex task domain action *goto* (x), which causes our robot to move to position x, and
- the two complex maintenance domain actions
  $calibrate = (start\_calibrate, stop\_calibrate, Calibrating)$,
  and $viscan = (start\_viscan, stop\_viscan, Viscaning)$.
  calibrate calibrates the laser range finder, i.e., it causes *Calibrated* to hold afterwards, and *viscan* tests for imminent collisions using a visual obstacle detector.

Each of these actions is always possible, and *calibrate* is the only way to set the fluent $Calibrated = true$. The set of constraints is

$$C = \{goto\,(x)\ \textsc{Equals}\ viscan,\ goto\,(x)\ \textsc{After}\ calibrate\},$$

which denotes that *goto* (x) is performed concurrently with *viscan* and only after the laser range finder was calibrated. These constraints imply – amongst others – three transformation rules. We abbreviate complex actions by their first letter with the usual modifiers, omitting the parameter of *goto* (x) as it is of no relevance for this example. Thus, G denotes the complex action *goto* (x), G^+ denotes *start_goto* (x), G^- denotes *stop_goto* (x) and G^F denotes the fluent *Going* (x). Furthermore, variables t_α refer to the position of an action α in a sequence of actions. The transformation rules are

(1) $(G\ \textsc{After}\ C, G^+, \neg\mathrm{Occurred}\,(C^-) \wedge \neg C^F, \{t_{C^+} < t_{C^-}, t_{C^-} < t_{G^+}\}, \emptyset, \emptyset)$

where Occurred indicates whether C^- was already performed, i.e., $\exists s', a_1, \ldots, a_N.\quad s = do([a_1, \ldots, a_N, C^-], s')$.

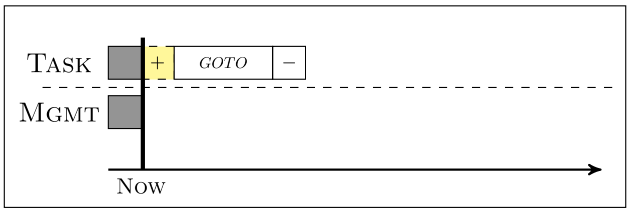
(2) $(G\ \textsc{Equals}\ V, G^+, \neg V^F, \{t_{V^+} \approx_\varepsilon t_{G^+}\}, \emptyset, \{between\,(V^+, G^-, V^F)\})$

(3) $(G \text{ EQUALS } V, G^-, V^F, \{t_{G^-} \approx_\varepsilon t_{V^-}\}, \emptyset, \emptyset)$
The agent has not performed any actions yet, i.e., the current situation is the initial situation S_0. The ε-slot length is 3 and the program to be performed is $\delta = [\textit{start\_goto}(x), \delta_1, \textit{stop\_goto}(x), \delta_2]$, where δ_1 and δ_2 are sequences of primitive actions that we do not want to detail here. Furthermore, both occurrence commitments $\mathcal{O}$ and persistence commitments $\mathcal{P}$ are empty in S_0. We now show how the execution of this program evolves in terms of single steps of the transition semantics:

(I) We start in situation $s = S_0$, with the program

$$\delta = [\textit{start\_goto}(x), \delta_1, \textit{stop\_goto}(x), \delta_2]$$

and commitment sets $\mathcal{O} = \emptyset$ and $\mathcal{P} = \emptyset$.



(i) Inspecting the set of constraint-implied transformation rules reveals that the next action to be performed, *start_goto* (x), requires self-maintenance attention, as both "$\neg\text{Occurred}(C^-) \wedge \neg C^F$" (from G AFTER C) and "$\neg V^F$" (from G EQUALS V) hold in s.
(ii) Therefore, the rules (1) and (2), together with *start_goto* are transformed into a constraint satisfaction problem over variables $t_{C^+}, t_{C^-}, t_{V^+}, t_{G^-}$. The domains are $D_{t_{C^+}} = D_{t_{C^-}} = D_{t_{V^+}} = D_{t_{G^-}} = \{1, 2, 3, 4\}$, because we know from the transformation rules that we need to perform four actions to transform the current situation into a situation that satisfies the self-maintenance requirements. The transformation rules further imply restrictions on the order of three pairs of actions, namely

- $C_{C^+,C^-} = \{(i, j) \mid i < j\}$
- $C_{C^-,G^+} = \{(i, j) \mid i < j\}$
- $C_{V^+,G^+} = \{(i, j) \mid i \approx_\varepsilon j\}$.

Furthermore, there is an additional set of global constraints for each pair of variables that states that their values are different. We illustrate the CSP in Figure 6.5. (iii) This CSP is solved using a simple backtracking CSP solver that yields (possibly amongst others) the solution $t_{C^+} = 1, t_{C^-} = 3, t_{V^+} = 2, t_{G^-} = 4$, which corresponds to the sequence of actions

$$\delta^* = [\textit{start\_calibrate}, \textit{start\_viscan}, \textit{stop\_calibrate}, R(\textit{start\_goto}(x))],$$

where $R(\textit{start\_goto}(x))$ is an indicator denoting that we already tried to perform self-maintenance for the action *start_goto* (x). This leads to the remaining program

$$\delta' = [\textit{start\_calibrate}, \textit{start\_viscan}, \textit{stop\_calibrate}, R(\textit{start\_goto}(x)), \delta_1, \textit{stop\_goto}x, \delta_2]$$

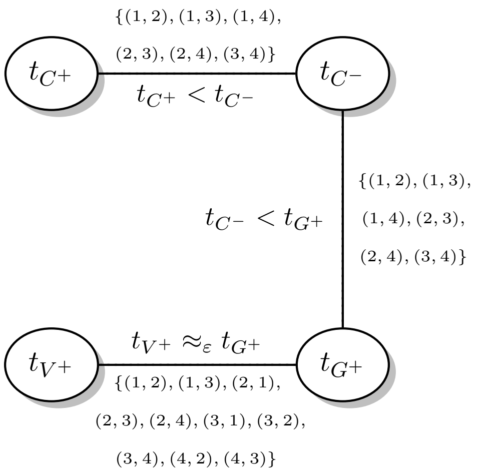


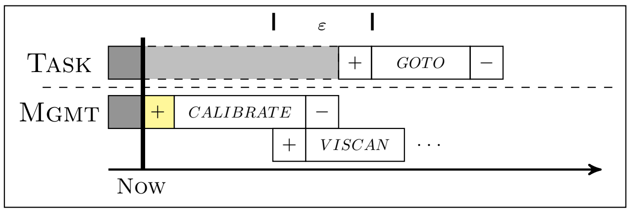
Figure 6.5: An exemplary CSP for four variables t_{C^+}, t_{C^-}, t_{G^+}, and t_{V^+} represented as nodes. Edges are labeled (left and below, respectively) with constraints between these variables and the set of possible variable assignments (above and right, respectively).

(iv) Rule (2) also yields the persistence commitment "*between* $\left(V^+, G^-, V^F\right)$", which states that we require the fluent V^F to hold from V^+ to G^-, thus we update $\mathcal{P}$ accordingly to $\mathcal{P}' = \{$*between* $(start\_viscan, stop\_goto\,(x)\,, Viscaning)\}$

(II) The second step of the transition semantics now faces the same situation $s = S_0$ as before, a modified program

$$\delta = [start\_calibrate, start\_viscan, stop\_calibrate,\\ R\,(start\_goto\,(x))\,, \delta_1, stop\_goto\,(x)\,, \delta_2],$$

modified persistence commitments
$\mathcal{P} = \{$*between* $(start\_viscan, stop\_goto\,(x)\,, Viscaning)\}$ and unmodified occurrence commitments $\mathcal{O} = \emptyset$.



(i) *start_calibrate* is performed.
(ii) The remaining program is

$$\delta' = [start\_viscan, stop\_calibrate,\\ R\,(start\_goto\,(x))\,, \delta_1, stop\_goto\,(x)\,, \delta_2]$$

(iii) Neither $\mathcal{P}$ nor $\mathcal{O}$ require updates.

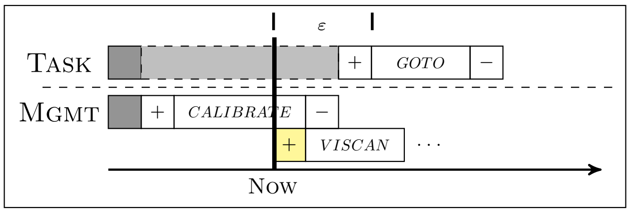
(III) The third step faces the situation

$$s = do\,(start\_calibrate, S_0)\,,$$

the program

$$\begin{aligned}\delta = [&start\_viscan, stop\_calibrate,\\ &R\left(start\_goto\left(x\right)\right), \delta_1, stop\_goto\left(x\right), \delta_2],\end{aligned}$$

and unmodified occurrence commitments $\mathcal{O} = \emptyset$ and persistence commitments $\mathcal{P} = \{\mathit{between}\left(start\_viscan, stop\_goto\left(x\right), Viscaning\right)\}$.



(i) *start_viscan* is performed.
(ii) The remaining program is

$$\begin{aligned}\delta' = [&stop\_calibrate, R\left(start\_goto\left(x\right)\right),\\ &\delta_1, stop\_goto\left(x\right), \delta_2].\end{aligned}$$

(iii) As *start_viscan* is performed, $\mathcal{P}$ updates to $\mathcal{P}' = \{\mathit{until}\left(stop\_goto\left(x\right), Viscaning\right)\}$, which denotes that the fluent *Viscaning* is required to hold from now on until $stop\_goto\left(x\right)$ is performed.

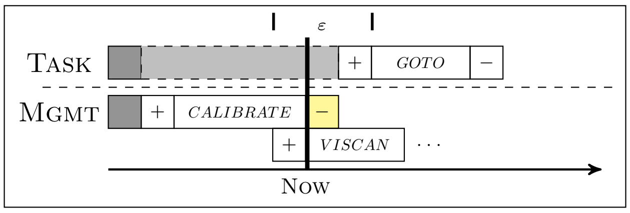
(IV) The fourth step begins with situation

$$s = do([start\_calibrate, start\_viscan], S_0)$$

and program

$$\begin{aligned}\delta = [&stop\_calibrate, R\left(start\_goto\left(x\right)\right),\\ &\delta_1, stop\_goto\left(x\right), \delta_2],\end{aligned}$$

$\mathcal{P} = \{\mathit{until}\left(stop\_goto\left(x\right), Viscaning\right)\}$, and $\mathcal{O} = \emptyset$.



(i) *stop_calibrate* is performed.
(ii) The remaining program is

$$\delta' = [R\left(start\_goto\left(x\right)\right), \delta_1, stop\_goto\left(x\right), \delta_2].$$

(iii) Neither $\mathcal{P}$ nor $\mathcal{O}$ require updates.

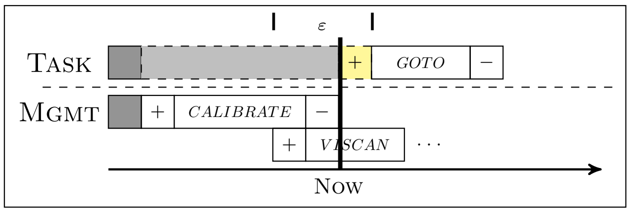
(V) The situation advanced to

$$s = do([start\_calibrate, start\_viscan, stop\_calibrate], S_0)$$

with program remainder

$$\delta = [R\,(start\_goto\,(x))\,, \delta_1, stop\_goto\,(x)\,, \delta_2]$$

and commitments $\mathcal{O} = \emptyset$ and $\mathcal{P} = \{until\,(stop\_goto\,(x)\,, Viscaning)\}$.



(i) Neither "$\neg$Occurred $(stop\_calibrate) \wedge \neg Calibrating$" (from $goto\,(x)$ AFTER $calibrate$), nor "$\neg Viscaning$" (from $goto\,(x)$ EQUALS $viscan$) hold in s, thus, recovering from the faulty situation where $start\_goto\,(x)$ was supposed to be executed, has succeeded and $start\_goto\,(x)$ may finally be performed.
(ii) Therefore, the recovery indicator is removed and $start\_goto\,(x)$ is performed. Leaving

$$\delta' = [\delta_1, stop\_goto\,(x)\,, \delta_2].$$

(iii) $\mathcal{P}$ and $\mathcal{O}$ do not change.

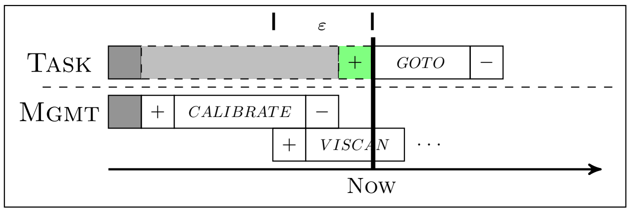
(VI) The situation is now

$$s = do([start\_calibrate, start\_viscan, \\ stop\_calibrate, start\_goto\,(x)], S_0)$$

with program remainder

$$\delta' = [\delta_1, stop\_goto\,(x)\,, \delta_2],$$

and $\mathcal{O} = \emptyset$.



(i) Now the sequence of actions δ_1 – which may require further self-maintenance recovery – is performed. Thus, $\mathcal{P}$ and $\mathcal{O}$ may have been updated to $\mathcal{P}' = \{until\,(stop\_goto\,(x)\,, Viscaning)\} \cup \mathcal{P}_{\delta_1}$ and $\mathcal{O}' = \mathcal{O}_{\delta_1}$, respectively.
(ii) $\delta' = [stop\_goto\,(x)\,, \delta_2]$.

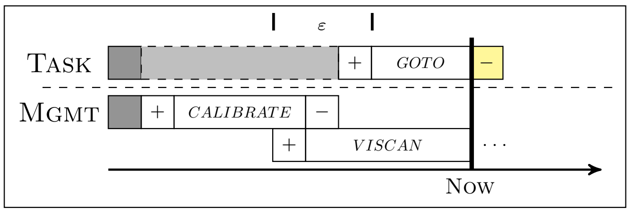
(VII) The situation has advanced to

$$s = do([start\_calibrate, start\_viscan, stop\_calibrate, \\ start\_goto\,(x)\,, \delta_1], S_0),$$

the program remainder is

$$\delta = [stop\_goto\,(x)\,, \delta_2],$$

and commitments are $\mathcal{O} = \mathcal{O}_{\delta_1}$ and $\mathcal{P} = \{until\,(stop\_goto\,(x)\,, Viscaning)\} \cup \mathcal{P}_{\delta_1}$.



(i) $stop\_goto\,(x)$ is to be performed, but there is another transformation rule left unsatisfied. Rule (3) states to assure that $stop\_viscan$ is performed immediately after $stop\_goto\,(x)$.
(ii) This is achieved by solving a CSP about rule (3), which leads to a program remainder $\delta^* = [R\,(stop\_goto\,(x))\,, stop\_viscan]$ without further modifications to $\mathcal{P}$ and $\mathcal{O}$.
(iii) Then

$$\delta' = [R\,(stop\_goto\,(x))\,, stop\_viscan, \delta_2]$$

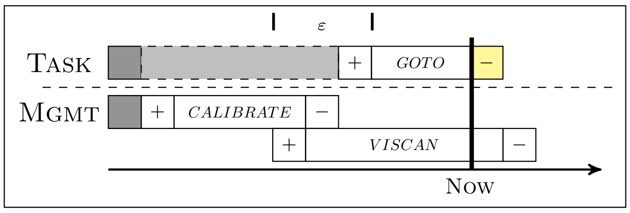
(VIII) Step eight faces situation

$$s = do([start\_calibrate, start\_viscan, stop\_calibrate,\\ start\_goto\,(x)\,, \delta_1], S_0)$$

with program remainder

$$\delta = [R\,(stop\_goto\,(x))\,, stop\_viscan, \delta_2]$$

and commitments $\mathcal{O} = \mathcal{O}_{\delta_1}$ and $\mathcal{P} = \{until\,(stop\_goto\,(x)\,, Viscaning)\} \cup \mathcal{P}_{\delta_1}$.



(i) Evaluating the recovery indicator for $stop\_goto\,(x)$ leads to re-evaluation of rule (3). We notice that $Viscaning$ still holds. Since we also know that $stop\_goto\,(x)$ is about to be executed, we can retract $until\,(stop\_goto\,(x)\,, Viscaning)$ from $\mathcal{P}$.
(ii) Finally, $stop\_goto\,(x)$ is performed and $\mathcal{P}$ is updated accordingly to $\mathcal{P}' = \mathcal{P}_{\delta_1}$.

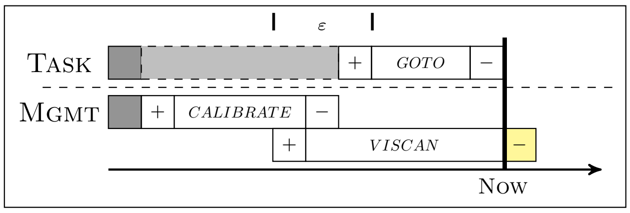
(IX) The current situation is

$$s = do([start\_calibrate, start\_viscan, stop\_calibrate,\\ start\_goto\,(x)\,, \delta_1, stop\_goto\,(x)], S_0),$$

the program remainder is

$$\delta = [stop\_viscan, \delta_2],$$

with $\mathcal{P} = \mathcal{P}_{\delta_1}$ and $\mathcal{O} = \emptyset$.



(i) *stop_viscan* is performed.
(ii) Neither $\mathcal{P}$ nor $\mathcal{O}$ require updates.
(iii) $\delta' = \delta_2$.

(X) The execution of δ_2 continues and may require further self-maintenance or not, but eventually the program finishes.

6.1.5 Evaluation

We evaluate our approach in terms of two measures. First, we compare the time needed by our method to recover from a failure situation to the time needed for the re-planning that we would have to initiate if our self-maintenance was not in effect. Second, we look at the times needed to construct and solve the CSP, and the total time to recovery, that is the time from detecting a failure until the recovery program is executed completely.

6.1.5.1 Plan Repair

When the next action to be executed has constraints attached to it, those constraints need to be satisfied before the action can actually be performed. We evaluate restoring from such unsatisfied constraints in a coffee domain scenario. We briefly present the relevant portions of this scenario as follows. The robot is located in an apartment with several rooms. It can drive from one room to another with an action *goto* and it can *pickup* and *drop* things with its arm. The robot has two components related to its navigation, namely a motor and a collision avoidance module. Without any explicitly formulated recovery programs and no model of the components of the robot, the classical READYLOG interpreter has no choice but to start a full search for a sequence of actions related to components affected by the violated constraints such that the constraints are satisfied. To make the comparison between the self-maintenance-enabled interpreter and classical READYLOG more fair, we distinguish two different sets of actions to search with: the set Act_N of actions necessary to satisfy a component's requirements relative to a situation and the complete set of component related actions Act_F, both for each fluent mentioned in the component requirement.
For the evaluation we set up the following scenario: We start in the initial situation ($s = S_0$) and every component of the robot is in its initial state. That is, motor = *offline*, colli = *offline*, and arm = *idle*. Further, there is no complex action active in s and there are no commitments. The program δ to be executed consists of a single *goto* action: δ = *start_goto* (x). The constraints relevant for this configuration are *goto* (x) DURING colli = *scanning* and *goto* (x) EQUALS motor = *slow*. We expect our self-maintenance-enabled interpreter to find a program which restores a situation such that both components mentioned comply to the constraints. Since the classical READYLOG interpreter has no means of taking into account constraints or component models,

Table 6.2: Comparing plan repair efficiency as the time in milliseconds until a situation satisfying the constraints *goto* (x) DURING colli $=$ *scanning* and *goto* (x) EQUALS motor $=$ *slow* was restored

Interpreter	Actions	Min	Average	Median	Max
Classical READYLOG	Act_F	580	630	631	780
Classical READYLOG	Act_N	20	27	29	44
Self-Maintaining	Act_F	≤ 1	11	16	41

we accept any solution which yields colliState $=$ *scanning* and motorState $=$ *slow* before performing *goto* (x). For the READYLOG interpreter we further define the set of necessary actions Act_N as {*start_colli*, *stop_colli*, *restart_colli*} and the complete set of component related actions Act_F as {*start_colli*, *stop_colli*, *restart_colli*, *start_motor*, *stop_motor*, *start_running*, *stop_running*, *increase_speed*, *decrease_speed*,*init_motor*, *restart_motor*}.
Summarizing the above, we compare

- Classical READYLOG: Non-deterministically find programs over (i) Act_N and (ii) Act_F, which yield colli $=$ *scanning* and motorState $=$ *slow* before executing *start_goto* (x)
- Self-Maintaining: Find a program such that both constraints hold by means of a transformation of the constraints into a CSP and a program synthesis over the related component automata.

To provide solid results, each setup was tested 100 times each. The times between the start and reaching a first feasible solution are given in milliseconds. The results are depicted in Table 6.2. It is of now surprise, that the classical READYLOG interpreter's performance gets worse and worse with an increasing number of actions taken into account. Our approach utilizes additional knowledge and a specialized CSP solver which provides us with a more directed approach that outperforms classical READYLOG by far.

6.1.5.2 Robustness

To evaluate the robustness of our approach we set up an extended coffee service robot scenario and we let the robot perform a coffee delivery service in different configurations of it. While the robot tries to execute its program, we introduce failures such as the laser sensor going offline and we see how long it takes for the robot to recover from such failures. We evaluate two measures, namely the time needed to construct and to solve the CSP in a given failure-situation, and the time needed to recover from detection of a failure until the program to reach full recovery is executed completely.
Slightly simplified versions of the robot's procedures are given in Algorithm 10. Please note that the actions used in the procedures are all task domain actions and that they are all complex actions. That is, they have to be decomposed to primitive *start_...* and *stop_...* actions each.
We modelled three components of the robot to be considered in self-maintenance, namely the motor, the collision avoidance, and the arm. Each component has an automaton with appropriate nodes for every possible status and corresponding transitions for the actions that the robot can take to influence the status. In our evaluation scenario, after

Algorithm 10: The procedures used for the coffee service scenario

```
proc coffee_service
  while( true,
    [ while( requests = [], [ wait ] )endwhile,
      ?( requests=[ H | _R ] ),
      fetch_order(H), fetch_coffee, deliver_coffee, goto(kitchen) ]
  ) endwhile
endproc

proc fetch_order(Room)
  [ goto(Room), locate(Person,Room), take_order(Person) ]
endproc

proc fetch_coffee
  [ while( not( pos=kitchen), goto(kitchen) ) endwhile,
    locate(Person,kitchen), ?( Person = chef ),
    ?( task = T ), place_order( T ), pickup(coffee) ]
endproc

proc deliver_coffee
  [ ?( holding = coffee ), transport(coffee), drop(coffee) ]
endproc
```

every execution of a primitive action, we check whether we need to introduce a failure. The seeding of failures depends on the setup. In every setup, we repeat the execution of the coffee service procedure until at least 100 failures have occurred and at least 100 CSPs have been solved.

Setup 1 In a first setup we start the coffee service with an initial number of four requests. We seed one *defect* for every component for every procedure. We have a total of seven constraints which are attached to different actions used in the procedures. The CSPs that result from introducing failures during the execution have a size of one to three constraints. We solved 240 CSPs in total with mostly two constraints involved. The results are given in Table 6.3a. As we can see, the size of the CSP does not contribute much to the overall time. Surprisingly, the time to full recovery was larger for the motor than for the colli, although the automaton for the motor has more states and recovery in the motor component usually requires two actions while the colli only needs one.

Setup 2 In a second setup we reduced the number of requests to one and introduced a failure after every primitive action instead. As in *Setup 1* the number of constraints in the CSPs does not contribute to the CSP times. Results for CSP construction and solving as well as the time taken to yield full recovery are similar to *Setup 1* which is why we omit the individual numbers for *Setup 2* here.

Table 6.3: Results for Setup 1
(Min = minimum, Mdn = median, Avg = average, Max = maximum)

(a) Time to solve CSPs of sizes between 1 and 3 constraints in milliseconds

Size	Min	Mdn	Avg	Max
Any	19	23	25.18	71
1	21	24	26.48	46
2	19	23	$25,20$	71
3	19	23	24.73	41

(b) Time (in milliseconds) to fully recover from a *defect* component

Com-ponent	Min	Mdn	Avg	Max
Any	20	23	25.69	60
Colli	21	23.5	26.14	60
Motor	20	23	$25,24$	43

Table 6.4: Results for Setup 4
(Min = minimum, Mdn = median, Avg = average, Max = maximum)

(a) Time (in milliseconds) to solve CSPs of sizes between 1 and 6 constraints

Size	Min	Mdn	Avg	Max
Any	20	23	$24,80$	63
1	20	23	$22,63$	26
2	20	23	$24,89$	63
3	20	23	$24,58$	57
6	23	26	$27,13$	37

(b) Time to recover from *defect* motor and/or *defect* collision avoidance in milliseconds

Com-ponent	Min	Mdn	Avg	Max
Any	20	41	50.27	132
Motor	20	22	34.40	33
Both	49	67	71.84	132

Setup 3 In a third setup we tried to combine the first two setups by using a higher number of request, namely four, and by introducing a failure after every primitive action. Again, results were very similar to the previous setups.

Setup 4 In a last setup we return to introducing only one failure per component for every procedure. However, we start with eight requests and we increase the number of constraints to eleven, particularly adding constraints for the *transport* action reaching a total of five constraints for that action. Also, the constraints were made more complex by mentioning more than one component. This resulted in CSPs with more constraints and a larger search space for recovery programs. We give results in Table 6.4. While the effect of having more constraints in the CSP is negligible, the recovery time depends on the number of components involved.

6.1.6 Discussion

In this section we presented our approach to self-maintenance for autonomous robots controlled by READYLOG. We modify a given program at run-time using explicitly formulated temporal constraints that relate self-maintenance actions with actions from the task domain. This way, we achieve more robust and enduring operation and take care of maintenance when it is relevant: at execution time. Keeping our approach in one framework allows to use all of READYLOG's features in maintenance and recovery. Up to now, we have not considered any optimization issues in scheduling the maintenance actions, we just apply one possible scheduling returned as a solution to the CSP. It could be worthwhile looking into methods on how to determine which of a set of solutions is optimal. Two extensions could be considered in future work. *Explanation:* Since the robot knows which constraint(s) failed in a particular situation and it probably does not have means to take care of it itself, the robot can at least exhibit to the user what went wrong. *Interaction:* Alternatively, if the robot can not handle a constraint itself (e.g., *no_emergency_off* while *drive*) but knows, that a human user could do, it can simply trigger an interaction, e.g., ask *"Could you please release my emergency button?"*.

6.2 Interactive Autonomy

Apart from taking into account its internal shortcomings, a possibility to increase robustness and flexibility of a service robot is trying to handle the fallibility of a human user. In this section we present a system for flexible command interpretation to facilitate natural human-robot interaction in a domestic service robotics domain. We particularly target settings inspired by the *General Purpose Service Robot* test from the RoboCup@Home competition (Wisspeintner et al., 2009), where a robot is confronted with ambiguous and/or faulty user inputs in form of natural spoken language. The main goal of our approach is to provide a system capable of resolving these ambiguities and of interactively achieving user satisfaction in the form of doing the right thing, even in the face of incomplete, ill-formed, or faulty commands.

We model the processing of natural spoken language input as an interpretation process. More precisely, we first analyse the given utterance syntactically by using a grammar. Then, we cast the interpretation as a planning problem where the single actions available to the planner are to interpret syntactical elements of the utterance. If, in the course of interpreting, ambiguities are detected, the system uses decision-theory to weigh up different alternatives. The system is also able to initiate clarification to resolve ambiguities and to handle errors as to arrive at a successful command interpretation eventually. Since our current high-level control already knows about the robot's capabilities (the actions and the parameters that these actions need), we tightly connect the interpretation with it.

6.2.1 Related Work

We want to build upon the theory of *speech acts* as introduced by Austin (1975) and Searle (1969). Based on these works Cohen and Levesque (1985) already investigated a formal theory of rational interaction. We, restrict ourselves to command interpretation and do not aim for a full-fledged dialogue system. Nevertheless, we follow their formal theory of interpretation and we carry out our work in the context of the situation calculus.

Definite clause grammars for parsing and interpreting natural language have already been used by Beetz et al. (2001). Despite being relatively ad hoc and the fact that the small grammar only covered a constrained subset of English, their system provided a wide spectrum of communication behaviours. However, in contrast to their approach we

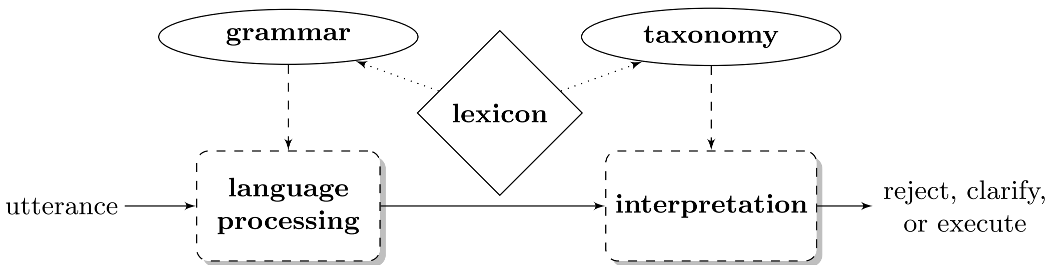


Figure 6.6: General overview of our language interpretation architecture

want to account for incomplete and unclear utterances both by using a larger grammar as well as adding interpretation mechanisms to the system.

Fong et al. (2003b) developed a system on a robot platform that manages dialogues between human and robot. Similar to our approach, input to the system is processed by task planning. However, queries are limited to questions that can either be answered with yes or no or a decimal value. A more advanced system combining natural language processing and flexible dialogue management is reported on by Clodic et al. (2007). User utterances are interpreted as communicative acts having a certain number of parameters. The approach is missing a proper conceptual foundation of objects and actions, though. This makes it hard to adapt it to different platforms or changing sets of robot capabilities.

Görz and Ludwig (2005), on the other hand, built a dialogue management system well-founded by making use of a concept hierarchy formalized in Description Logics (DL). Both, the linguistic knowledge as well as the dialogue management are formalized in DL. This is a very generic method for linking lexical semantics with domain pragmatics. However, this comes with the computational burden of integrating description logics and appropriate reasoning mechanisms. We want to stay within our current representational framework, that is, the situation calculus and Golog, and we opt to exploit the capabilities to reduce computational complexity with combining programming and planning.

6.2.2 Flexible Command Interpretation

As mentioned before, we cast the language processing of spoken commands on a domestic service robot as an interpretation process. We decompose this process into the following steps. First, the acoustic utterance of the user is being transformed into text via a speech recognition component which is not discussed here but could be done similar to what we presented in Section 4.1. The transcribed utterance is then passed on for syntactic analysis by a grammar. After that, the interpretation starts, possibly resolving ambiguities and generating intermediate responses. If the utterance could be interpreted successfully, it is executed, otherwise it is rejected. The overall architecture is shown in Figure 6.6. We now present the individual steps in more detail.

6.2.2.1 Syntactical Language Processing

Given the textual form of the user utterance, the first thing we do is a syntactical analysis. This syntactic operation uses a grammar. Since the entirety of the English language is not context-free (Shieber, 1985) and the targeted application domain allows for a

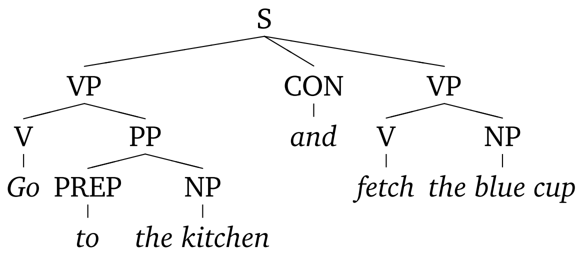


Figure 6.7: Syntax tree for the utterance "*Go to the kitchen and fetch the blue cup*".

reasonable restriction, we confine ourselves to directives. Directives are utterances that express some kind of request. Following Ervin-Tripp (1976) there are six types of directives:

1. Need statements, e.g., “I need the blue cup.”
2. Imperatives, e.g., “Bring me the blue cup!”
3. Imbedded[1] imperatives, e.g., “Could you bring me the blue cup?”
4. Permission directives, e.g., “May I please have the blue cup?”
5. Question directives, e.g., “Have you got some chewing gum?”
6. Hints, e.g., “I have run out of chewing gum.”

Ervin-Tripp characterizes question directives and hints as being hard to be identified as directives even for humans. Moreover, permission directives are mostly used only when the asker is taking a subordinate role, which is not the case of a human instructing a robot. That is why we restrict ourselves to a system that can handle need statements, imperatives and imbedded imperatives only.

A Grammar for English Directives To make the robot understand the user's command, for any of these directives we distill the *essence* of the utterance. To eventually arrive at this, we first perform a purely syntactic processing of the utterance. An analysis of several syntax trees of such utterances revealed structural similarities that we intend to capture with a grammar. An example for a syntax tree is given in Figure 6.7.
Using common linguistic concepts, the main structure elements are verb (V), auxiliary verb (AUX), verb phrase (VP), noun phrase (NP), conjunction (CON), preposition (PREP), and prepositional phrase (PP). A verb phrase consists of a verb and a noun phrase, a noun phrase is a noun, possibly having a determiner in front of it. A prepositional phrase is a preposition and a noun phrase. We further introduce a structure element *object phrase* which is a noun phrase, a prepositional phrase, or concatenations of the two. Multiple verb phrases can be connected with a conjunction. What is more, commands to the robots may be prefixed with a salutation. Also, for reasons of politeness, the user can express courtesy by saying “please”. Putting all this together, we arrive at a base grammar that can be expressed in Extended Backus-Naur Form (EBNF) (Scowen, 1993) as shown in Figure 6.8.
In addition to the base grammar we need a base lexicon that provides us with the vocabulary for elements such as prepositions, auxiliary verbs, courtesies, conjunctions,

[1]Ervin-Tripp (1976) uses the word *imbedded* where *embedded* could be used just as well. We stick to the notation of the original paper.

```
                    s —> salutation utterance | utterance
%
            utterance —> needstatement | imperative | imbedded_imperative
%
        needstatement —> np vp | needphrase vp
           imperative —> vp
  imbedded_imperative —> aux np vp
           needphrase —> "i" prompt "you to"
% verb phrase
                   vp —> vp' | vp' conjunction vp
                  vp' —> verb | verb obp | courtesy vp'
% object phrase
                  obp —> np | pp | np obp | pp obp
% noun phrase
                   np —> noun | pronoun | determiner noun
% propositional phrase
                   pp —> prep np
```

Figure 6.8: Base grammar in EBNF

determiners, and pronouns. To generate a system that is functional in a specific setting, we further need a lexicon containing all verbs for the capabilities of the robot as well as all the objects referring to known entities in the world. This depends on the particular application, though. That is why we couple this to the domain specification discussed later. The base grammar, the base lexicon, and the domain specific lexicon then yield the final grammar that is used for syntactical processing.
Since we are only interested in the core information, the most relevant parts of the utterance are verbs, objects, prepositions, and determiners. We can drop auxiliary verbs, filler words, courtesies, and alike without losing any relevant information. Doing so, we finally arrive at an internal representation of the utterance in a prefix notation depicted below, that we use for further processing.

```
[and, [[Verb, [objects, [[Preposition, [Determiner,Object]],...]] ]], ...]
```

The list notation contains the keyword *and* to concatenate multiple verb phrases and it uses the keyword `objects` to group the object phrase. If an utterance is missing information we fill this with `nil` as a placeholder. For the utterance "*Bring the cup to the kitchen*" the essence would be constructed as follows:

$$[\,and,\ \underbrace{[\ \underbrace{[bring,\ [objects,\ \underbrace{[\ \underbrace{[\ \overbrace{nil}^{preposition}\ ,[the, cup]]}_{1st\ object},\ \underbrace{[\ \overbrace{to}^{preposition}\ ,[the, kitchen]]}_{2nd\ object}\ ]]}_{object\ list}\ ]}_{action\ structure},\ \ldots\ ]}_{list\ of\ verb\ phrases/requests}\ ]$$

6.2.2.2 Planning Interpretations

After syntactic pre-processing of an utterance into the internal representation, the system uses decision-theoretic planning to arrive at the most likely interpretation of the utterance, given the robot's capabilities. The interpretation is supposed to match the request with one of the abilities of the robot (called a skill) and to correctly allocate the parameters that this skill requires.

In order to do that, first, we need to identify the skill that is being addressed. We are going about this from the verb which has been extracted in the syntactical processing, possibly leaving ambiguities on which skill is referred to by the verb. Second, the objects mentioned in the utterance need to be mapped to entities in the world that the robot knows about. Finally, a skill typically has parameters, and the verb extracted from the utterance has (multiple) objects associated to it. Hence, we need to decide which object should be assigned to which parameter. To make things worse, it might very well be the case that we have either too many or too few objects in the utterance for a certain skill.
We cast understanding the command as a process where the single steps are interpretation actions, that is, interpreting the single elements of the utterance. At this point READYLOG and its ability to perform decision-theoretic planning comes into play. The overall interpretation can be modelled as a planning problem. The system can choose different actions (or actions with different parameters) at each stage. Since we want to achieve an optimal interpretation, we make use of decision-theoretic planning. That is to say, given an optimization theory, we try to find a plan, i.e. a sequence of actions, that maximizes the expected reward.

Domain Specification During the interpretation process we need to access the robot's background knowledge. We organize this knowledge to capture generic properties and to make individual parts available to (only) those components which need them. Three types of information are distinguished: *linguistic*, *interpretation*, and *system*. The linguistic information contains everything that has to do with natural language while interpretation information is used during the interpretation process and system information features things like the specific system calls for a certain skill. The combination of these three types is then what makes the connection from natural language to robot abilities. We (again) use ideas from Gu and Soutchanski (2008) to structure our knowledge within our situation calculus-based representation.
In an ontology, for every *Skill* we store a *Name* as an internal identifier that is being assigned to a particular skill during the interpretation. A skill further has a *Command* which is the denotation of the corresponding system call of that skill. *Synonyms* is a list of possible verbs in natural language that may be used to refer to that skill. *Parameters* is a list of objects that refer to the arguments of the skill, where *Name* again is a reference used in the interpretation process, *Attributes* is a list of properties such as whether the parameter is numerical or string data. *Significance* indicates whether the parameter is optional or required, and *Preposition* is a (possibly empty) list of prepositions that go with the parameter. For the information on entities in the world (e.g. locations and objects) we use a structure *Object* which again has a *Name* as an internal identifier used during the interpretation. *Attributes* is a list of properties such as whether the object "is a location" or if it "is portable". *Synonyms* is a list of possible nouns that may refer to the object and *ID* is a system related identifier that uniquely refers to a particular object. The taxonomy trees for *Skills* and *Entities* are shown in Figure 6.9.

Basic Action Theory Now that we have put down the domain knowledge on skills and objects, we still need to formalize the basic action theory for our interpretation system. We therefore define three actions, namely *interpret_action*, *interpret_object*, and *assign_argument*. For all three we need to state precondition axioms and successor

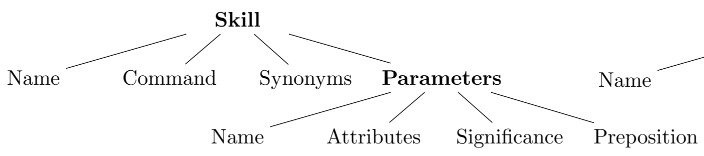


Figure 6.9: Taxonomy trees for the information used for command interpretation

state axioms. We further need several fluents, that describe the properties of the interpretation domain we operate in. Let us take a look at those fluents first. We use the fluents *spoken_verb* (s) and *spoken_objects* (s) to store the verb and the list of objects extracted in the syntactic processing. Further, we use the fluents *assumed_action* (s) and *assumed_objects* (s) to store the skill and the list of objects that we assume to be addressed by the user, respectively. Both these fluents are nil in the initial situation S_0 since no interpretation has taken place so far. The fluent *assumed_arguments* (s) contains a list of pairings between parameters and entities. Finally, *finished* (s) indicates whether the interpretation process is finished.

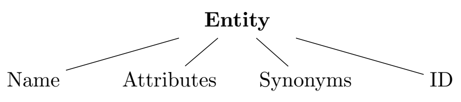
Let us now turn to the three interpretation actions. The precondition axiom for *interpret_action* states that *interpret_action* (k) is only possible if we are not done with interpreting yet and the word k actually is a synonym of the verb spoken.

$$\begin{aligned} &Poss(interpret\_action(k),\ s) \equiv \\ &\qquad \neg finished(s) \\ &\quad \wedge \quad synonym(spoken\_verb(s),\ k) \end{aligned}$$

Similarly, *interpret_object* (e) is possible for an entity e only if we are not finished and the object (from *spoken_object* (s)) is a synonym appearing for e.

$$\begin{aligned} &Poss(interpret\_object(e),\ s) \equiv \\ &\qquad \neg finished(s) \\ &\quad \wedge \quad synonym(\ second(\ second(\ first(spoken\_object(s))\ )\ ),\ e) \end{aligned}$$

The functions *first* and $second$ are helper functions that do not belong to the basic action theory. They are used to return the first and second element of list, respectively. Finally, the precondition axiom for *assign_argument* for an entity e and parameter p checks whether the interpretation process is not finished and there is no entity assigned to the parameter yet. Further, p needs to be a parameter of the assumed skill and we either have no preposition for the object or the preposition we have matches the preposition associated with the parameter. Lastly, the attributes associated to parameter p need to



be a subset of the attributes for the entity.

$$
\begin{aligned}
& Poss(assign\_argument(p),\ s) \equiv \\
& \qquad \neg finished(s) \\
& \quad \wedge \quad (p,\ second(\ first(assumed\_objects)) \notin assumed\_arguments(s) \\
& \quad \wedge \quad parameter(assumed\_action(s),\ p,\ \_) \\
& \quad \wedge \quad [\ first(\ first(assumed\_objects(s))\ ) = nil \\
& \qquad\quad \vee\ preposition(assumed\_action(s),\ p,\ first(first(assumed\_objects)))\ ] \\
& \quad \wedge \quad parameter\_attributes(assumed\_action(s),\ p,\ A1) \\
& \quad \wedge \quad entity\_attributes(second(\ first(assumed\_objects)\ ),\ A2) \\
& \quad \wedge \quad A1 \subset A2
\end{aligned}
$$

To allow for aborting the interpretation process we additionally introduce an action *reject* which is always possible.
After detailing the preconditions of actions, we now lay out how these actions change the fluents introduced above. The fluents *spoken_verb* and *spoken_objects* contain the essence of the utterance to be interpreted. The effect of the action *interpret_action* (k) is to reset the fluent *spoken_verb* to `nil` and to set the fluent *assumed_action* to the assumed skill k.

$$
\begin{aligned}
& \gamma_{spoken\_verb}(y, a, s) \equiv \\
& \qquad (a = interpret\_action(k) \wedge y = nil) \\
& \gamma_{assumed\_action}(y, a, s) \equiv \\
& \qquad (a = interpret\_action(k) \wedge y = k)
\end{aligned}
$$

The action $interpret\_object(e)$ iteratively removes the first object (in a list of multiple objects) from the fluent $spoken\_objects$ and adds it to the fluent $assumed\_objects$ along with its preposition (if available).

$$
\begin{aligned}
& \gamma_{spoken\_objects}(y, a, s) \equiv \\
& \qquad (a = interpret\_object(e) \wedge y = pop\_first(spoken\_objects(s)))
\end{aligned}
$$

The action $assign\_argument(p)$ removes the object from the fluent $assumed\_objects$ and it adds the pair (p, e) for parameter p and entity e to the fluent $assumed\_arguments$.

$$
\begin{aligned}
& \gamma_{assumed\_objects}(y, a, s) \equiv \\
& \qquad (a = assign\_argument(p) \wedge y = pop\_first(assumed\_objects(s))) \vee \\
& \qquad (a = interpret\_object(e) \wedge \\
& \qquad\quad y = assumed\_objects(s) \oplus (\underbrace{first(first(spoken\_objects(s)))}_{preposition\ of\ first\ object}, e))
\end{aligned}
$$

$$
\begin{aligned}
& \gamma_{assumed\_arguments}(y, a, s) \equiv \\
& \qquad (a = assign\_argument(p) \wedge \\
& \qquad\quad y = assumed\_arguments(s) \oplus (p, \underbrace{second(first(assumed\_objects(s)))}_{name\ of\ first\ entity}))
\end{aligned}
$$

Finally, the fluent *finished* is set to *true* if either the action was *interpret_action* and there are no more objects to process (i.e. *spoken_objects* is empty) or the action was *assign_argument* and there are no more objects to assign (i.e. *assumed_objects* is empty). Further it is also set to *true* by the action *reject*.

$$\begin{aligned}
&\gamma^{+}_{finished}(a, s) \equiv \\
&\quad (a = interpret\_action \land spoken\_objects = \emptyset) \lor \\
&\quad (a = assign\_argument \land assumed\_objects = \emptyset) \lor \\
&\quad (a = reject)
\end{aligned}$$

From the effect axioms given above we can compile the successor state axioms.

Programs Using the basic action theory described above, the overall interpretation process can now be realized with READYLOG programs as follows. In case of multiple verb phrases we process each separately. For each verb phrase, we first interpret the verb. Then, we interpret the objects before we assign them to the parameters of the skill determined in the first step. The procedures to do so are

```
proc $interpret\_verbphrase$
  $solve$( {
    ( pickBest( $var$, $AllActions$,
        $interpret\_action(var)$ )
    | $reject$ )
    while $\neg finished$ do
      $interpret\_objectphrase$ endwhile
  }, $horizon$, $reward\_function$ )
endproc
```

with

```
proc $interpret\_objectphrase$
  (pickBest( $var$, $AllEntities$,
             $interpret\_object(var)$ )
  | $reject$)
  if $finished$ then $nil$
  else
    (pickBest( $var$, $AllParams$,
               $assign\_argument(var)$ )
    | $reject$)
  endif
endproc
```

where *AllActions*, *AllEntities*, and *AllParams* are sets of all skills of the robot, all entities known to the robot, and all parameters of a skill in the robot's domain specification, respectively. We consider more intelligent selection methods than taking all items available in the evaluation. Recall that the *solve*-statement initiates decision-theoretic planning, where the non-deterministic construct **pickBest**$(var, VarSet, prog)$ evaluates the program *prog* with every possibility for *var* in *VarSet* using the underlying optimization theory given mainly by the reward function, which rates the quality of resulting situations. To design an appropriate reward function situations that represent better interpretations need to be given a higher reward than those with a not so good interpretation. A possible reward function could be to give a reward of 10 if the assumed action

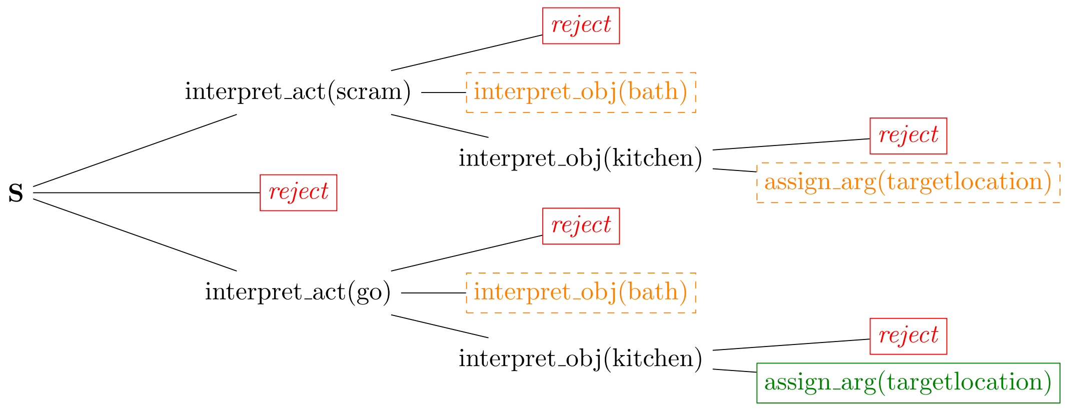


Figure 6.10: Interpretation of the essence `S = [`*and*`, [[move, [`*objects*`, [[to, [the,kitchen]] ]]] ] ]` derived from the utterance "Move to the kitchen"

is not *nil* and one could further add the difference between the number of assigned arguments and the total number of parameters required by the selected skill. Doing so results in situations with proper parameter assignment being given higher reward than those with fewer matches. If two possible interpretations have the same reward, one can either enquire with the user on which action to take or simply pick one of them at random.

Example Consider the exemplary utterance "Move to the kitchen." After syntactical processing we have the internal representation `[`*and*`, [[move, [`*objects*`, [[to, [the,kitchen]] ]]] ] ]`. Using the program given above and a small basic action theory as introduced before, one of the skills available to the robot that has *go* as a synonym may be *goto* which is stored in *assumed_action* by the action *interpret_action*. Then, $interpret\_object(kitchen)$ will assume *kitchen* as the object (along with the preposition *to*). However, it could also interpret "move" as bringing some object somewhere which leads to a lower reward, because a parameter slot remains unassigned. Trying to assign arguments for the skill *goto* may succeed since *kitchen* is an entity that has the *Location* attribute as would naturally be required for the target location parameter of a *goto* skill. Comparing the rewards for the different courses of interpretation the system will pick the interpretation with the highest reward, which is executing the $goto(kitchen)$ skill.

6.2.2.3 Clarification and Response

Things might not always go as smooth as in our example above. To provide a system that has capabilities beyond a pure interface to translate utterances to system calls we therefore include means for clarification if the utterance is missing information.
If the verb is missing, our grammar from the syntactical processing will already fail to capture to utterance. Hence, we only consider missing objects for clarification in the following. We propose to model clarification as an iterative process where the user is enquired for each missing object. To generate the appropriate questions to the user we make use of the information that has been extracted from the utterance already and of the information stored in the ontology. Assuming that we know about the skill that is

being addressed we can look up the parameters required. Using a template that repeats the user's request as far as it has been interpreted we can then pose an accurate question and offer possible entities for the missing objects.
Consider that the user said "*Go!*" missing the required target location. So the target location is what we want to enquire about. This can be achieved with using a generic template as follows:

"you want me to [assumed_action] [assumed_arguments].
[preposition] which [attribute] ? [list of entities]"

where *[preposition]* is the preposition associated to the parameter in question and *[attribute]* is one of the attributes associated to the parameter. Only including one of the parameter's attributes seems incomplete, but suits the application, since it still leads to linguistically flawless responses. Including *[assumed_arguments]* in the response indicates what the system has already managed to interpret and additionally reminds the user of his original request. The system would respond to the utterance "*Go!*" from above with "*You want me to go. To which location? kitchen or bath?*", which is exactly what we want.
To avoid annoying the user we put a limit on the number of entities to propose to the user. If the number of available entities exceeds, say, three we omit it from the question. Moreover, to improve on the response we add what we call "unspecific placeholders" to the domain ontology. So for locations we might add "*somewhere*" and for portable thing we might add "*something*" which are then used in the response at the position of a missing object.
There might be cases where information is not missing but instead is either wrong or the skills available to the robot do not allow for execution. Our system should provide information on rejecting faulty or non-executable requests. Depending on the type of error, we propose the following templates for explanation.

1. "*I cannot [spoken_verb].*" if the verb could not be matched with any skill, i.e. $spoken\_verb \neq nil$.
2. "*I do not know what [next spoken_object] is.*" if the object could not be matched with any entity known to the robot, i.e. $spoken\_objects \neq nil$.
3. "*I cannot [assumed_action] [preposition] [next assumed_object].*" if the object could not be assigned to a parameter of the skill that is being addressed, i.e. $assumed\_objects \neq nil$.

Note that *[next some_list]* retrieves the next element from *some_list*. Also note that the fluent values we mentioned above are sound given our basic action theory since the action *reject* sets the fluent *finished* to true and leaves the other fluents' values as they were when the utterance was rejected.

6.2.3 Experimental Evaluation

To investigate the performance of our system we evaluate it along two dimensions, namely understanding and responsiveness.

Table 6.5: Survey results by sentence type

type	absolute frequency	relative frequency
imperatives	114	87%
imbedded imperatives	6	5%
need-statements	2	2%
hints	4	3%
wh-questions	3	2%
others	3	2%

6.2.3.1 Understanding

The aim of our approach was to provide a system that is able to react to as many commands for a domestic service robot given in natural language as possible. With the generic grammar for English directives our approach is able to handle more utterances than previous approaches based on finite state grammars such as (Doostdar et al., 2008). To evaluate how far off we are from an ideal natural language interface we conducted a user survey. The survey was carried out on-line with a small group of (about 15) predominantly tech-savvy students. A short description of the robot's capabilities was given and participants were asked to provide us with sample requests for our system. Participants took the survey without any assistance, except the task description.

We received a total of 132 submissions. Firstly, we are interested in the general structure of the answers to see whether our grammar is appropriate. Therefore, Table 6.5 shows the submissions itemized by sentence type.

Syntactically speaking, the grammar can cover imperatives, imbedded imperatives and need-statements, which make for 92.37% of the survey results. However, some of these utterances do not possess the verb-object-structure we assumed in our system. For example, "Make me a coffee the way I like it" contained an adverbial ("the way I like it") which we did not account for neither in the grammar nor in the interpretation process. It is technically possible to treat adverbials as entities and thus incorporate such utterances. A better founded approach, however, would be to introduce the concept of adverbials to our system as a special case of objects that modify the mode of a skill. We leave this for future work, though. Still, 77.01% of the survey entries provide the assumed modular verb-object-structure and can therefore be processed by our system successfully.

To test the resilience against erroneous utterances we tested the system's response to the set of utterances given in Table 6.6. In case that an object is missing that is required as a parameter by a skill (as in E1) the system will enquire for clarification with offering possible entities. To be able to handle unspecific objects we included those in our grammar and we treat them just like missing objects and initiate a clarification procedure. Prepositions help in assigning objects to parameter slots of a skill. With only one parameter as in the utterance E3 we do not require the preposition in order to come to a successful termination of our interpretation process. With multiple parameters that

Table 6.6: Types of erroneous utterances

id	example utterance	problem
E1	“fetch”	object missing
E2	“go somewhere”	unspecific object
E3	“go the kitchen”	missing preposition
E4	“collect the bath room”	nonsense
E5	“smurf”	unknown word
E6	“the cup i need” or “the cup”	ill-formed syntax

are identical in all their attributes we would need additional information, though. An utterance can be nonsense when the objects do not match the attributes required for a parameter as specified in our ontology. In cases such as the one in E4 the system rejects the utterance since “the bath room” is not a “portable object” as required for the “collect” skill. This is then also mentioned in an explanation given to the user. Words that do not occur in our lexicon cannot be processed. Hence, the system will fail when confronted with unknown words. When the system is confronted with ill-formed syntax, it fails at the syntactical processing stage. This is because the grammar cannot handle utterances with unknown constructions.

6.2.3.2 Responsiveness

To evaluate the performance of our system in terms of speed, we evaluated the system using the following domain. The example agent has four different skills: getting lost (no parameter), going somewhere (1 parameter), moving an object to some location (2 parameters) and moving an object from some location to some location (3 parameters). Additionally, our domain contains different entities with appropriate attributes: a kitchen (location), a bath (location), a coffee cup (portable object) and a football trophy (decoration). Some of the synonyms for skills and entities are ambiguous, namely (1) “go” may refer to “get lost” as well as to “go somewhere”, (2) “move” may refer to “get lost”, “go somewhere”, “move something somewhere” or “move something from somewhere to somewhere”, and (3) “cup” may refer to the coffee cup as well as to the football trophy.
We tested four different versions of the system with different requests involving various degrees of complexity using the following utterances:

(i) “scram”

(ii) “go to the kitchen”

(iii) “could you please move the cup to the kitchen”

(iv) “go to the kitchen and move the cup to the bath room”

(v) “i need you to move the cup from the bath room to the kitchen”

Utterance (i) is a very simple request. It addresses a skill with no parameters and the used synonym “scram” is unambiguous. The skill addressed in utterance (ii) involves one parameter and the used synonym “go” is ambiguous. Utterance (iii) involves a skill

Table 6.7: Response times in different test scenarios

	i	ii	iii	iv	v
base	0.08 s	0.28 s	2.37 s	2.67 s	9.06 s
action pre-selection	0.08 s	0.24 s	2.10 s	2.29 s	7.15 s
entity pre-selection	0.06 s	0.19 s	2.01 s	2.16 s	7.41 s
parameter pre-selection	0.09 s	0.19 s	1.06 s	1.20 s	4.05 s
action + entity	0.05 s	0.16 s	1.70 s	1.85 s	6.07 s
entity + parameter	0.05 s	0.13 s	0.99 s	1.10 s	3.75 s
action + parameter	0.09 s	0.13 s	0.71 s	0.83 s	2.52 s
full combination	0.07 s	0.10 s	0.68 s	0.76 s	2.35 s

with two parameters and the synonym “move” is also ambiguous. Utterance (iv) is the combination of utterances (ii) and (iii) linked with an “and”. The skill requested in utterance (v) has three parameters and the synonym “move” is again ambiguous.

The depth of the search tree spanned in the planning process depends on the number of objects. For example, the depth of the search tree for utterance (i) is exactly 1 while the depth of the search tree for utterance (v) is 7. Note that utterance (iv) involves two distinct search trees, since it contains two independent verb phrases which are interpreted separately.

The five utterances were tested with the following versions of the system. First, we used the base system as described in Section 6.2.2, it does not include any explicit performance improvements speed-wise. The first row of Table 6.7 shows the performance of the base system.

Improvements Second, we considered systems incorporating different pre-selection methods. For each interpretation step (interpreting action, entity and parameter), we can pre-select the candidates that may be considered by the appropriate interpretation action. This can lead to considerably lower branching factors.

The pre-selection process for *interpret_action* involves two criteria: synonym and parameter count. This means that candidates are eliminated from the list if the spoken verb is not one of the candidates’ synonyms or if the number of parameters the candidate provides is lower than the number of spoken objects. This is due to the fact that we want every spoken object to be assigned to a parameter slot, so we only have to consider skills that provide a sufficient amount of parameter slots. If we would also consider skills with fewer parameters, we would have to drop parts of the user’s utterance. One could argue that reducing the set of available skills is a restriction from a theoretical point of view. However, ignoring elements that where uttered could easily frustrate the user. Hence, the restriction only has little practical relevance. The second row of Table 6.7 illustrates the performance of the base system plus *action pre-selection*.

Entities are pre-selected just by checking whether the spoken object is one of the entity’s synonyms. The third row of Table 6.7 shows the response times including the base system plus *entity pre-selection*.

Table 6.8: Response times (in seconds) depending on the two types of difficulty

# of	**tree**	**#actions/#entities**			
objects	**depth**	**1/1**	**1/5**	**5/1**	**5/5**
1	3	0.15	0.32	0.48	1.27
2	5	0.47	0.96	1.61	3.50
3	7	2.54	4.83	7.40	13.92
4	9	18.77	34.00	39.72	68.19
5	11	153.40	267.55	154.97	276.20

Pre-selecting parameters involves checking the attributes and the preposition of the corresponding candidate. Hence, the attributes of the parameter slot have to be a subset of the entities attributes, and if a preposition was provided along with the spoken object or entity, respectively, then it has to match the preposition required by the parameter. The fourth row of Table 6.7 lists response times of the base system plus *parameter pre-selection*. Rows five, six and seven illustrate the performance of different pairs of the three pre-selection methods. The last row shows the performance of the system including all three enhancements. As we can see, the full combination yields an improvement except for utterance i where the difference is negligible. The relative improvement of the enhancements increases with the complexity of the utterances. That is to say, the more complex the utterance, the more the speed-ups pay off.

Altogether, the complexity of the search tree is affected by the different branching factors at each level, and the depth which depends on the number of spoken objects. The branching factor at the first level depends on the number of actions that have the spoken verb as a synonym. The branching factor at the second level depends on the number of entities that have the spoken object as a synonym. At the third level the branching factor depends on the number of parameters of the respective skill. We further evaluated our optimized system by varying the two complexity factors independently.

Along the rows of Table 6.8 we varied the number of spoken objects. Along the columns we varied the number of actions that have the spoken verb as a synonym and the number of entities that have the spoken object as a synonym. The number of parameters of the appropriate skill are not varied, since this number already depends on the amount of spoken objects. In this test scenario the parameters of a skill became distinguishable for the system by providing distinct prepositions for each parameter. Different entities became distinguishable through their attributes and the skills were distinguishable by the number of parameters. So we had five skills with one, two, three, four and five parameters, respectively.

Table 6.8 shows, that the number of spoken objects has a greater influence on the computation time than has ambiguity. This is indicated by the last two rows which only contain measurements greater than 10 seconds. That is unacceptable for fluent human-robot interaction. We can also observe that action pre-selection performs very well in this test scenario. All tests in the last row address a skill with five parameters. In this test scenario there was no other skill involving five or more parameters. As a consequence, the action pre-selection can rule out the other four skill candidates which implies nothing less than reducing the branching factor of the top node from 5 to 1 and

Table 6.9: Response times with different lexicons

	small lexicon	large lexicon
utterance i	0.07 sec	0.08 sec
utterance ii	0.10 sec	0.14 sec
utterance iii	0.68 sec	0.90 sec
utterance iv	0.76 sec	1.15 sec
utterance v	2.35 sec	2.51 sec

thus reducing the computation time by a factor of approximately 5. This also results in comparable computation times for the combinations 1/1 (153.40 sec) and 5/1 (154.97 sec) as well as 1/5 (267.55 sec) and 5/5 (276.20 sec).
Finally, we analysed whether the lexicon size poses a computational problem. For this purpose, we added 50,000 nouns to the lexicon and used the full combination test setup from Table 6.7. Now, Table 6.9 indicates that the additional computational effort to process the utterances with a large lexicon plays no significant role.

6.2.3.3 Discussion

An important point towards successful human-robot interaction with respect to the user's patience is the system's reaction time. The average human attention span (for focused attention, i.e. the short-term response to a stimulus) is considered to be approximately eight seconds (Cornish and Dukette, 2009). Therefore, the time we require to process the utterance of a user and react in some way must not exceed eight seconds. Suitable reactions are the execution of a request, rejection, or to start a clarification process.
Hence, the question whether computation times are reasonable is in fact the question whether the computation times exceed eight seconds. Nonetheless, the answer is not as easy as the question. The optimized system performs well in a realistic test scenario as shown by the last row of Table 6.7. In turn, complex test scenarios can lead to serious problems as Table 6.8 indicated. However, we saw that ambiguity is a smaller problem than the length[2] of an utterance. Skills that have more than three parameters are rare in the field of mobile service robots. In fact, the skills with four or five parameters we used in the tests of Table 6.8 needed to be created artificially for lack of realistic examples.

6.2.4 Summary

We presented a system for interpreting commands issued to a domestic service robot using decision-theoretic planning. The proposed system allows for a flexible matching of utterances and robot capabilities and is able to handle faulty or incomplete commands by using clarification. It is also able to provide explanations in case the user's request cannot be executed and is rejected. The system covers a broader set of possible requests than existing systems with small and fixed grammars. Also, it performs fast enough to prevent annoying the user or loosing his or her attention.

[2]By the length of an utterance, we mean the number of spoken objects.

A possible extension of the approach could be to include a list of the n most probable interpretations and to verify with the user on which of these should be executed. Moreover, properly integrating the use of adverbials as qualifiers for nouns both in the grammar and the interpretation process would further improve the system's capabilities.

6.3 Discussion

In this chapter we presented two approaches towards increasing the robustness and flexibility of our domestic service robot. In Section 6.1 we tried to make the robot take care of maintaining its internal workings by explicitly formulating the dependencies between actions on the task level and the internal state of the robot. This is done by means of an online transformation process on the program, i.e. a plan generated by READYLOG, right before its execution. In Section 6.2 we tried to account for the user's fallibility by casting the interpretation of a user utterance as a planning process. The goal is to map the elements of the utterance to a skill available on the robot and its parameters, respectively. To weigh different alternatives that appear during the interpretation we employ an optimization theory that uses a reward function which favours more complete mappings. Both approaches increase robustness and flexibility of our service robot, one towards internal failures on the robot, the other against imprecisions in human-robot interaction. With the two methods in place we increase the potential time that a robot can operate autonomously once it is deployed.
Although there exist certain restrictions in our attempts, they still contribute towards increasing the resilience of a robot against internal and external failures. Thus, they lay ground for building domestic service robots that feature long-term autonomy. The self-maintenance could be improved, for example, by using an optimization theory to schedule the maintenance actions. Adding explanations when a failure cannot be prevented or including the human user in the maintenance for things that the robot cannot do itself would add the the power of the method. Also, step-wise dropping the assumptions we made provides opportunities for further research. Namely, investigating whether and how constraints between two actions on the task level are possible without invalidating the high-level program. The natural language interpretation could be enhanced to deal with adverbials. Rating several possible interpretations and confirming with the user which one is the intended meaning could further increase the flexibility and lead to an even higher user satisfaction. The method itself could be used with other grammars, for example, to interpret spatial references like the ones used in Section 5.5.5.

7 Conclusion and Future Work

In this chapter we conclude the thesis. We summarize our contributions and highlight our achievements. Afterwards, we propose topics for potential future work.

7.1 Summary and Conclusions

In this work, we contributed to the integration of qualitative reasoning and human-robot interaction for intelligent domestic service robots. First, we provided an existing mobile robot system with a basic set of capabilities to be deployed in domestic domains, namely safe collision avoidance, semantic mapping and flexible object recognition. Then we extended the robot with means for affective human-robot interaction by adding robust speech recognition, one-step face detection, recognition and learning as well as modular gesture recognition. Additionally, the robot was endowed with speech synthesis and a virtual facial display. We successfully applied the interactive system in a scenario where its ability for deliberation was a benefit. Next, we integrated a powerful mechanism for representing and using qualitative notions in high-level programs in general and for representing and reasoning with qualitative positional information in particular. Finally, we implemented a novel form of self-maintenance that allows for recovering from internal errors in the robot system. Furthermore, we added a flexible method for interpreting spoken commands that is able to cope with imprecise, incomplete and even faulty utterances.

Basic Components
Starting with a former soccer playing robot platform, we prepared the base system for an application in domestic service robotics. Apart from removing any soccer-specific hardware equipment we paid particular attention to building a local navigation system with collision avoidance that allows for safe operation in human environments. A mapping mechanism was designed and implemented that allows to construct and to maintain semantically enriched representations of the environment the robot operates in. These representations can be used throughout the system, starting from global localization with a metric map, over path planning with a topological map to using attributes within the high-level control such as where to look for people or where certain activities might take place. The semantic mapping scheme is a valuable addition to the robot software system. It simplifies the process of keeping the robot's world model up to date and it makes it easier to maintain the models used throughout the robot's modules in one central location. We still lack an extension that automatically detects changes

in the environment and modifies the corresponding model accordingly, though. We endowed the robot with means for recognition of known objects by straightforwardly using existing mechanisms as well as by proposing a novel architecture for combining such methods to be able to recognize unspecific or yet unseen objects. The object recognition system is very flexible and it is designed to be able to adapt to changes on the robot such as adding a new sensor or a new algorithm to detect objects. This kind of evolution is anticipated and can very easily be accounted for. Using available technology the robot is also capable of executing basic manipulation tasks. Our extensions to the base system constitute a solid foundation to develop an intelligent interactive domestic service robot.

Human-Robot Interaction
We presented the domestic service robotics domain and introduced RoboCup@Home as an initiative to evaluate and benchmark systems in that domain. While we only briefly sketched the effort of the @Home initiative, the constantly growing number of participants and the increase in performance indicate that it implements its objectives successfully. To enable natural human-robot interaction we presented approaches in three important modes of interaction, namely speech, faces, and gestures. Robustly recognizing spoken commands is a must for safe and convenient operation of a domestic service robot. Our approach first detects speech that is directed towards the robot. Then it combines two decoders using different language models and matches their output. This increases the system's robustness against false positive recognitions and thus makes for a reliable input method, especially in noisy environments. It provides sufficiently accurate results and it is computationally efficient. Specifying new input to be recognized can be done by just providing an appropriate finite state grammar. To detect and to recognize people we proposed a one-step real-time system for integrated face detection, recognition and learning using random forests. Random forests are particularly suited in our application because they prevent overfitting while retaining high classification accuracy. Further, they have training times that are much smaller than that of comparable approaches. This allows for deploying the system on our service robot leaving enough of the limited computing resources for other tasks. Also response times are kept very low. This makes for a solid component that we can use for responsive human-robot interaction. Since a significant amount of information is conveyed using non-verbal signals and because gestures are a convenient mode to instruct a robot we proposed an architecture for and presented a prototypical implementation of a modular gesture recognition system. The recognition is split into steps each of which provides information that is used for the next step but that can be used on its own as well. First, hand detection is done using skin color extracted from face detection. Then, the detected hand is verified and its posture is classified. At this stage static gestures like pointing can already be utilized to instruct the robot. Dynamic gestures such as waving are then recognized by first tracking the hand position over time. The trajectory is matched against a set of known gesture trajectories using a modified version of a recognizer originally designed for one-stroke hand-written gestures. Although our system suffers from shortcomings in the posture recognition it still allows to build gesture enabled human-robot interaction applications. Due to the modular design we can realize improvements easily by exchanging any of the components with an improved

version. The set of interactive capabilities is complemented by modules for speech synthesis as well as for people detection and tracking. Furthermore, an interactive video screen is used to display an artificial face that can express basic emotions and that illustrates, for instance, the robot's current focus of attention. The monitor can also be used to display instructions that a human user needs to follow in a particular task and to receive user commands by touch input. The additional components make for an even more natural interaction which is important for successful applications of a service robot in human environments.
In an interactive demonstration scenario, we showed the application of the robot in a domestic service setting. In a home-like environment the robot's task is to help setting the table. It uses its newly added methods for human-robot interaction and its sophisticated logic-based high-level control to complete the task, in this case re-ordering a set of three cups according to instructions given by the human user with using a minimal number of movements. This is done with the help of speech recognition, face detection and pointing gesture recognition as well as by employing decision-theoretic optimization in the planning of moving the cups. The demonstration illustrates the successful extension of our robot with different modes for interaction. It also shows the benefits of deliberation in the high-level control of the robot.

Qualitative Representations and Reasoning
Humans frequently use qualitative notions in their communication. For a domestic service robot to be useful to humans it must be able to make sense of such notions accordingly. Using a semantics based on fuzzy sets we integrated qualitative fluents in the situation calculus such that we are able to transparently use these fluents in our high-level control programs. Also, we extended our high-level control with fuzzy controllers, a mechanism that is popular to allow specifying control tasks with a simple set of human-like if-then-else rules. The qualitative fluents allow for using human notions in specifying the behaviour of the robot. We can use them in regular READYLOG programs but we can also use them to implement a rulebase for more reactive tasks that focus on control. In both cases we facilitate using human expertise more directly. Since positional information is of particular importance in HRI we presented an extension of our qualitative representation for the spatial domain. We use an existing method to represent positional information in a qualitative fashion. It has a direct correspondence to the Euclidean space. In combination with our approach of using fuzzy logic with fuzzyfication and defuzzyfication we exploit this connection between qualitative and quantitative values to use Euclidean geometry for reasoning with qualitative positional information. The tight integration with our logic-based mechanism for the high-level control of the robot behaviour allows for a seamless and effective application in domestic settings. The proper formalization of and reasoning with spatial information is an asset in joint human-robot scenarios. Because positional information appears so often in human-robot communication its integration greatly improves the naturalness and usability of the robot.

Robustness and Flexibility
Finally, we presented two efforts towards more robust and more flexible controllers for domestic service robots. A robot for complex tasks is a complex system itself. To

account for possible failures in the robot's inner workings we endow it with a form of self-maintenance. We formulate requirements that every task action which the robot can execute has with respect to the robot's state in terms of the state of its components such as software modules. Right before executing an action the set of constraints associated with that action is checked and if necessary a recovery program is constructed and executed as to restore a failure-free configuration of the system. Despite our restrictions on which actions the robot can take to perform self-maintenance we can cover a large portion of common failures with our approach. This certainly improves the robustness of the system and allows for longer autonomous operation. We not only try to account for possible failures on the internal side but we also prepare to flexibly handle human fallibility in receiving commands. To this end, we cast the interpretation of possibly erroneous spoken commands given to the robot as a natural language interpretation process. Using decision-theoretic planning we try to map what was said to what the robot is able to do. In case of underspecified or faulty requests, the robot asks for clarification or rejects execution. With our flexible natural language interpretation we account for many ordinary failures in receiving commands. Not only can we make up for imprecisions caused by the human factor but we can also remedy faulty input due to problems in the signal processing of the robot. This added resilience is another improvement of the robustness of the system and allows for interactive long-term operation.

Overall, our extensions of the base system provide a solid basis for research in domestic service robotics. The individual modules for human-robot interaction may be outperformed by specialized solutions but add up to a respectable set of interactive capabilities for a service robot that can communicate with humans in domestic environments. Our efforts to bridge the gap between human and robot representations simplify both specification of the robot behaviour as well as interaction with the robot. Especially our proper integration of representing and reasoning with qualitative positional information greatly adds to the naturalness and usability of the robot. Our approach to self-maintenance increases the robustness against internal faults. The flexible interpretation of natural language commands raises the resilience against common faults in receiving external input. In summary, our efforts represent a selected but highly valuable set of steps to integrate qualitative reasoning and human-robot interaction for intelligent domestic service robots.

7.2 Future Work

We have laid ground for further research on a cognitive service robot for domestic domains. We first concentrated on building a strong base system, and on extending it with basic means for human-robot interaction. Then we focused on extensions of the high-level control system to handle human-like representations in general and qualitative spatial information in particular. Lastly, we investigated methods to increase the robustness and flexibility of the interactive robotic system.

As for the base system, we did not really put much effort into mobile manipulation except for providing a baseline to work with. Hence, this largely remains an issue for further research for the robot platform discussed here. The semantic mapping scheme could be extended by a component that recognizes changes in the environment and that

adapts the internal representations according to those changes automatically. This could be done in a joint effort to equip the robot with a method for simultaneous localization and mapping (SLAM), but it could also just borrow ideas from the field.
Since this thesis was not meant to be an exercise in any of the particular methods of HRI there remains room for improvement with each of them. The speech recognition so far is limited to work reliably with close speech. We imagine it to be worthwhile to examine the system's performance for far-field speech, especially when we integrate filter methods such as beam forming for on-board microphones in combination with sound-source localization (Calmes et al., 2007a) and in combination with other modalities like we already investigated earlier with laser-based object detection (Calmes et al., 2007b) and face detection (Schiffer et al., 2014). In the face processing system the accuracy of the face detection component was low. We plan to improve on this by experimenting with aggregation techniques similar to what has been done in the AdaBoost cascade (Jones and Viola, 2003). Besides detecting faces and recognizing identities we could also analyse each face further such as to recognize gaze or emotions. This would enable using joint attention in the interaction and to incorporate the emotional state of the human in the robot's behaviour. The recognition of different postures of the hand was suboptimal in our approach. An important next step is to fully integrate the "*four box*" features from (Kölsch and Turk, 2004b) in our random forest implementation. Future work further includes the integration of improvements on the preciseness of pointing gestures as presented in (Breuer et al., 2010). So far, new gestures have to be defined manually. Preferable would be if a human user could introduce new gestures to the robot in an interactive fashion.
We investigated how to achieve human-like representations and control as part of our logic-based high-level control. Our focus was on qualitative positional information where we exploited the correspondence of a particular calculus with Euclidean space. It would be interesting to see how other spatial calculi such as the RCC could be integrated. In our model of qualitative positional information we assumed that a fixed set of categories with our unit distance and orientation relation is appropriate for all possible contexts. We would like to drop this assumption and learn which categories are required in what context by interacting with a human instructor and also learn their design, such as in (Robinson, 2000). This would also allow for adapting to different parameterizations with different users.
Also with our approaches to increase robustness and flexibility some interesting research questions remain open. When something in the inner workings of the robot went wrong, we might offer an explanation on what that was since we know which constraints could not be satisfied anyway. In some cases, the robot might not be able to recover from a fault on its own. However, with the help of a human the situation might be cleared up. Initiating appropriate interaction in such cases would certainly be worth looking at. Our command interpretation so far can handle a restricted set of directives. Extending the grammar to cover more commands, for instance, by integrating adverbials for nouns would improve the system's capabilities. Also, applying the idea of interpretation not only to directives but also to other speech such as to positional information as discussed in Section 5.5.5 is an interesting direction for future work.
Lastly, we extended our high-level control towards incorporating the means of human-robot interaction that we deployed on our robot. Fully considering the human user in the high-level control by modelling his or her behaviour was not considered and is a

very large area for research. Mixed-initiative planning and joint actions come to mind as challenges here.

The overall problem of developing domestic service robots is of course much broader than what has been discussed in this thesis. However, even if many issues remain to be solved or to be improved, we made several important contributions to integrate high-level reasoning and human-robot interaction for domestic service robots.

Bibliography

Alami, R., Clodic, A., Montreuil, V., Sisbot, E. A., and Chatila, R. (2005). Task planning for human-robot interaction. In *Proceedings of the 2005 Joint Conference on Smart objects and ambient intelligence (sOc-EUSAI'05)*, pages 81–85, New York, NY, USA. ACM. Cited on page 16.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843. Cited on pages 140 and 141.

Arbib, M. (1998). Schema theory. In *The handbook of brain theory and neural networks*, pages 830–834. MIT Press. Cited on page 99.

Aristotle (1944). *Aristotle in 23 Volumes*, volume 21, translated by H. Rackham. Harvard University Press, Cambridge, MA. Cited on page 1.

Asensio, J. R., Montiel, J. M. M., and Montano, L. (1999). Goal directed reactive robot navigation with relocation using laser and vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 2905–2910. Cited on page 22.

Asimov, I. (1950). *I, Robot*. Gnome Press, New York. Cited on page 1.

Austin, J. L. (1975). *How to Do Things with Words*. Harvard University Press, 2 edition. Cited on page 157.

Axenbeck, T., Bennewitz, M., Behnke, S., and Burgard, W. (2008). Recognizing complex, parameterized gestures from monocular image sequences. In *Proceedings of the 8th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 687–692. Cited on page 83.

Bacchus, F., Halpern, J. Y., and Levesque, H. J. (1999). Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111(1–2):171–208. Cited on page 59.

Balch, T. and Yanco, H. A. (2002). Ten years of the AAAI mobile robot competition and exhibition: looking back and to the future. *AI Magazine*, 23(1):13–22. Cited on page 12.

Baumgartner, T. (2011). Visual gesture recognition on a mobile robot. Bachelor thesis, RWTH Aachen University, Aachen, Germany. Cited on page 6.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359. Similarity Matching in Computer Vision and Multimedia. Cited on page 33.

Beetz, M. (2001). Structured reactive controllers. *Journal of Autonomous Agents and Multi-Agent Systems*, 2(4):25–55. Cited on page 16.

Beetz, M., Arbuckle, T., Belker, T., Cremers, A. B., and Schulz, D. (2001). Integrated plan-based control of autonomous robots in human environments. *IEEE Intelligent Systems*, 16(5):56–65. Cited on pages 16 and 157.

Beetz, M., Jain, D., Mösenlechner, L., and Tenorth, M. (2010). Towards performing everyday manipulation activities. *Robotics and Autonomous Systems*, 58(9):1085–1095. Cited on page 16.

Beetz, M., Stulp, F., Radig, B., Bandouch, J., Blodow, N., Dolha, M., Fedrizzi, A., Jain, D., Klank, U., Kresse, I., Maldonado, A., Marton, Z., Mosenlechner, L., Ruiz, F., Rusu, R., and Tenorth, M. (2008). The Assistive Kitchen – A demonstration scenario for cognitive technical systems. In *The 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1–8. Cited on page 15.

Behnke, S. (2006). Robot competitions - ideal benchmarks for robotics research. In *Proceedings of the 2006 IROS Workshop on Benchmarks in Robotics Research*. IEEE/RSJ. Cited on page 12.

Bekey, G. A. (2005). *Autonomous Robots: From Biological Inspiration to Implementation and Control (Intelligent Robotics and Autonomous Agents)*. The MIT Press. Cited on page 1.

Belle, V. (2008). Detection and recognition of human faces using random forests for a mobile robot. Master's thesis, RWTH Aachen University, Aachen, Germany. Cited on page 6.

Belle, V., Deselaers, T., and Schiffer, S. (2008). Randomized trees for real-time one-step face detection and recognition. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR'08)*, pages 1–4. IEEE Computer Society. Cited on pages 5, 63, 83, and I.

Bellman, R. E. (1957). *Dynamic programming*. Rand research study. Princeton University Press, Princeton, NJ. Cited on page 40.

Bennewitz, M., Faber, F., Joho, D., and Behnke, S. (2007). Fritz - a humanoid communication robot. In *Proceedings of the 16th IEEE International Symposium on Robot and Human interactive Communication (RO-MAN)*, pages 1072–1077. Cited on page 15.

Van den Bergh, M., Carton, D., de Nijs, R., Mitsou, N., Landsiedel, C., Kuehnlenz, K., Wollherr, D., Van Gool, L., and Buss, M. (2011). Real-time 3D hand gesture interaction with a robot for understanding directions from humans. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 357–362. Cited on pages 82 and 83.

Bhatt, M. and Loke, S. (2008). Modelling dynamic spatial systems in the situation calculus. *Spatial Cognition & Computation*, 8(1-2):86–130. Cited on page 100.

Bhatt, M., Rahayu, J. W., and Sterling, G. (2006). Qualitative spatial reasoning with topological relations in the situation calculus. In Sutcliffe, G. and Goebel, R., editors, *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 713–718. AAAI Press. Cited on page 101.

Black, A., Taylor, P., and Caley, R. (2001). The festival speech synthesis system version

1.4.2. Technical report, University of Edinburgh. Cited on page 90.

Black, A. W. and Lenzo, K. A. (2001). Flite: a small fast run-time synthesis engine. In *4th ISCA Tutorial and Research Workshop on Speech Synthesis*. Cited on page 90.

Black, A. W. and Taylor, P. A. (1997). The Festival Speech Synthesis System: System documentation. Technical Report HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK. Avaliable at http://www.cstr.ed.ac.uk/projects/festival/. Cited on page 90.

Bloch, I. (2006). Spatial reasoning under imprecision using fuzzy set theory, formal logics and mathematical morphology. *International Journal of Approximate Reasoning*, 41(2):77–95. Advances in Fuzzy Sets and Rough Sets. Cited on page 101.

Bloch, I. and Saffiotti, A. (2003). On the representation of fuzzy spatial relations in robot maps. In Bouchon-Meunier, B., Foulloy, L., and Yager, R., editors, *Intelligent Systems for Information Processing*, pages 47–57. Elsevier, NL. Cited on page 101.

Böhme, H.-J., Wilhelm, T., Key, J., Schauer, C., Schröter, C., Groß, H.-M., and Hempel, T. (2003). An approach to multi-modal human–machine interaction for intelligent service robots. *Robotics and Autonomous Systems*, 44(1):83–96. Best Papers of the Eurobot '01 Workshop. Cited on page 14.

Bohren, J., Rusu, R. B., Jones, E. G., Marder-Eppstein, E., Pantofaru, C., Wise, M., Mösenlechner, L., Meeussen, W., and Holzer, S. (2011). Towards autonomous robotic butlers: Lessons learned with the pr2. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5568–5575. IEEE. Cited on page 15.

Bolloju, N. (1996). Formulation of qualitative models using fuzzy logic. *Decision Support Systems*, 17(4):275–298. Cited on page 41.

Borenstein, J. and Koren, Y. (1991). The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 3(7):278–288. Cited on pages 21 and 22.

Bosch, A., Zisserman, A., and Munoz, X. (2007). Image Classification using Random Forests and Ferns. In *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV'07)*, pages 1–8. Cited on page 74.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, pages 144–152, New York, NY, USA. ACM. Cited on page 79.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-Theoretic Planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11(1):94. Cited on pages 41 and 60.

Boutilier, C., Reiter, R., Soutchanski, M., and Thrun, S. (2000). Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 355–362, Menlo Park, CA. AAAI Press. Cited on pages 60 and 61.

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1(Q2):1—15. Cited on page 83.

Breazeal, C. L. (2003). Toward sociable robots. *Robotics and Autonomous Systems*, 42(3–4):167–175. Socially Interactive Robots. Cited on page 63.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. Cited on pages 39, 74, and 76.

Brenner, M. (2007). Situation-aware interpretation, planning and execution of user commands by autonomous robots. In *The 16th IEEE International Symposium on Robot and Human interactive Communication (RO-MAN 2007)*, pages 540–545. Cited on page 16.

Breuer, T., Giorgana Macedo, G. R., Hartanto, R., Hochgeschwender, N., Holz, D., Hegger, F., Jin, Z., Müller, C., Paulus, J., Reckhaus, M., Álvarez Ruiz, J. A., Plöger, P. G., and Kraetzschmar, G. K. (2012). Johnny: An autonomous service robot for domestic environments. *Journal of Intelligent & Robotic Systems*, 66(1-2):245–272. Cited on pages 15 and 16.

Breuer, T., Ploeger, P. G., and Kraetzschmar, G. K. (2010). Precise Pointing Target Recognition for Human-Robot Interaction. In *Domestic Service Robots in the Real World. Workshop at SIMPAR 2010*, pages 229–240, Darmstadt, Germany. Cited on page 177.

Brick, T. and Scheutz, M. (2007). Incremental natural language processing for hri. In *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*, HRI ’07, pages 263–270, New York, NY, USA. ACM. Cited on page 14.

Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, volume 1, pages 341–346. Cited on page 21.

Bruce, A., Nourbakhsh, I., and Simmons, R. (2002). The role of expressiveness and attention in human-robot interaction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 4138–4142. Cited on page 91.

Bruce, J. and Veloso, M. (2006). Safe multi-robot navigation within dynamics constraints. *Proceedings of the IEEE, Special Issue on Multi-Robot Systems*, 94(7):1398–1411. Cited on page 22.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1–2):3–55. Cited on pages 14, 32, and 57.

Calmes, L., Lakemeyer, G., and Wagner, H. (2007a). Azimuthal sound localization using coincidence of timing across frequency on a robotic platform. *Journal of the Acoustical Society of America*, 121(4):2034–2048. Cited on page 177.

Calmes, L., Wagner, H., Schiffer, S., and Lakemeyer, G. (2007b). Combining Sound Localization and Laser-based Object Recognition. In Tapus, A., Michalowski, M., and Sabanovic, S., editors, *Papers from the AAAI Spring Symposium (AAAI-SS 2007)*, pages 1–6, Stanford, CA. AAAI Press. Cited on pages 90 and 177.

Čapek, K. (1920). *R.U.R. – Rossum’s Universal Robots*. Aventinum, Prague. Cited on page 1.

Chen, X., Ji, J., Jiang, J., Jin, G., Wang, F., and Xie, J. (2010). Developing high-level cognitive functions for service robots. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS’10)*, volume 1, pages 989–

996, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems. Cited on pages 15 and 16.

Chen, X., Jiang, J., Ji, J., Jin, G., and Wang, F. (2009). Integrating NLP with Reasoning about Actions for Autonomous Agents Communicating with Humans. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 2 of *WI-IAT'09*, pages 137–140, Washington, DC, USA. IEEE Computer Society. Cited on page 16.

Chen, X., Jin, G., Ji, J., Wang, F., and Xie, J. (2011). Kejia project: Towards integrated intelligence for service robots. Technical report, Multi-Agent Systems Lab, Department of Computer Science and Technology, University of Science and Technology of China, Heifei, China. Cited on page 16.

Chen, X., Lu, D., Chen, K., Chen, Y., and Wang, N. (2014). Kejia: The intelligent service robot for robocup@home 2014. Technical report, Multi-Agent Systems Lab, Department of Computer Science and Technology, University of Science and Technology of China, Heifei, China. Cited on page 15.

Chen, X., Sui, Z., and Ji, J. (2013). Towards metareasoning for human-robot interaction. In Lee, S., Cho, H., Yoon, K.-J., and Lee, J., editors, *Intelligent Autonomous Systems 12*, volume 194 of *Advances in Intelligent Systems and Computing*, pages 355–367. Springer Berlin Heidelberg. Cited on page 15.

Ciocarlie, M., Hsiao, K., Leeper, A., and Gossow, D. (2012). Mobile manipulation through an assistive home robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5313–5320. Cited on page 15.

Claßen, J., Hu, Y., and Lakemeyer, G. (2007). A Situation-Calculus Semantics for an Expressive Fragment of PDDL. In *AAAI'07: Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 956–961. AAAI Press. Cited on page 141.

Clementini, E., Felice, P. D., and Hernandez, D. (1997). Qualitative representation of positional information. *Artificial Intelligence*, 95(2):317–356. Cited on pages 44, 45, 48, 49, 99, 100, 118, 122, 123, and 127.

Clodic, A., Alami, R., Montreuil, V., Li, S., Wrede, B., and Swadzba, A. (2007). A study of interaction between dialog and decision for human-robot collaborative task achievement. In *Proc. of the International Symposium on Robot and Human interactive Communication (RO-MAN'07)*, pages 913–918. IEEE. Cited on page 158.

Cohen, P. R. and Levesque, H. J. (1985). Speech acts and rationality. In *Proc. of the 23rd Annual Meeting on Association for Computational Linguistics*, pages 49–60. Cited on page 157.

Cohn, A. G., Bennett, B., Gooday, J., and Gotts, N. M. (1997). Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, 1:275–316. Cited on page 98.

Cohn, A. G. and Hazarika, S. M. (2001). Qualitative Spatial Representation and Reasoning: An Overview. *Fundamenta Informaticae*, 46(1-2):1–29. Cited on page 98.

Cornish, D. and Dukette, D. (2009). *The Essential 20: Twenty Components of an Excellent Health Care Team*. RoseDog Books. Cited on page 171.

Correa, M., Pavez, M., Olave, G., Tampier, C., Retamal, C., Pairo, W., Bernuy, F., Herrmann, D., Verschae, R., Loncomilla, P., Martínez, L., Daud, O., and Ruiz-del Solar, J. (2014). UChile HomeBreakers 2014 Team Description Paper. Technical report, Department of Electrical Engineering - Advanced Mining Technology Center, Universidad de Chile, Santiago, Chile. Cited on page 14.

Correa, M., Ruiz-del Solar, J., Verschae, R., Lee-Ferng, J., and Castillo, N. (2010). Real-time hand gesture recognition for human robot interaction. In Baltes, J., Lagoudakis, M. G., Naruse, T., and Ghidary, S. S., editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 46–57. Springer Berlin Heidelberg. Cited on page 82.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. Cited on page 79.

Dautenhahn, K. (2013). Human-robot interaction. In Soegaard, M. and Dam, R. F., editors, *The Encyclopedia of Human-Computer Interaction*. The Interaction Design Foundation, Aarhus, Denmark, 2 edition. Cited on page 14.

Dautenhahn, K., Woods, S. N., Kaouri, C., Walters, M. L., Koay, K. L., and Werry, I. P. (2005). What is a robot companion - friend, assistant or butler? In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1192–1197. Cited on page 14.

Dedieu, D., Cadenat, V., and Soueres, P. (2000). Mixed camera-laser based control for mobile robot navigation. In *Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, volume 2, pages 1081–1086. Cited on page 22.

Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328. Cited on page 30.

Deselaers, T., Criminisi, A., Winn, J., and Agarwal, A. (2007). Incorporating on-demand stereo for real time recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, USA. Cited on pages 75 and 77.

Diankov, R. and Kuffner, J. (2008). Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA. Cited on page 34.

Dominey, P. F. and Boucher, J.-D. (2005). Developmental stages of perception and language acquisition in a perceptually grounded robot. *Cognitive Systems Research*, 6(3):243–259. Cited on page 99.

Doostdar, M. (2011). Multi-modal multi-target people tracking on a mobile robot. Diplomarbeit, RWTH Aachen University, Aachen, Germany. Cited on page 6.

Doostdar, M., Schiffer, S., and Lakemeyer, G. (2008). Robust speech recognition for service robotics applications. In *Proceedings of the International RoboCup Symposium 2008 (RoboCup 2008)*, volume 5399 of *LNCS*, pages 1–12. Springer. Best Student Paper Award. Cited on pages 5, 63, 167, and I.

Droeschel, D., Stückler, J., Holz, D., and Behnke, S. (2011). Towards joint attention for a domestic service robot - person awareness and gesture recognition using time-of-flight cameras. In *Proceedings of the IEEE International Conference on Robotics and*

Automation (ICRA), pages 1205–1210. Cited on page 83.

Drury, J. L., Yanco, H. A., and Scholtz, J. (2005). Using competitions to study human-robot interaction in urban search and rescue. *interactions*, 12(2):39–41. Cited on page 13.

Dubois, D. and Prade, H. (1998). An introduction to fuzzy systems. *Clinica Chimica Acta*, 270(1):3–29. Cited on pages 42, 43, 101, and 102.

Dutta, S. (1990). Qualitative spatial reasoning: A semi-quantitative approach using fuzzy logic. In Buchmann, A. P., Günther, O., Smith, T. R., and Wang, Y.-F., editors, *Proceedings of the First Symposium on Design and Implementation of Large Spatial Databases (SSD '89)*, volume 409 of *Lecture Notes in Computer Science*, pages 345–364. Springer. Cited on page 41.

Dylla, F., Ferrein, A., Jacobs, S., Lakemeyer, G., Richterich, C., and Schiffer, S. (2002a). Collision Avoidance in Real-Time with Look-Ahead (CARLA). In *Proceedings of the 3rd Workshop on Environment and Motion Modelling (UB 2002)*. Cited on pages 20 and I.

Dylla, F., Ferrein, A., and Lakemeyer, G. (2002b). Acting and deliberating using GOLOG in robotic soccer – a hybrid architecture. In *Proceedings of the 3rd International Cognitive Robotics Workshop (CogRob02) at AAAI-02*, pages 29–36. AAAI Press. Cited on page 20.

Dylla, F. and Kreutzmann, A. (2010). Agent Control by Adaptive Neighborhoods. In Bhatt, M., Guesgen, H., and Hazarika, S., editors, *Proceedings of the International Workshop on Spatio-Temporal Dynamics, co-located with the European Conference on Artificial Intelligence (ECAI-10)*, pages 55–60. ECAI Workshop Proceedings., and SFB/TR 8 Spatial Cognition Report Series. Cited on page 100.

Dylla, F. and Moratz, R. (2005). Exploiting qualitative spatial neighborhoods in the situation calculus. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., and Barkowsky, T., editors, *Spatial Cognition IV. Reasoning, Action, and Interaction*, volume 3343 of *Lecture Notes in Computer Science*, pages 304–322. Springer. Cited on page 100.

Ekman, P. and Friesen, W. V. (1971). Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2):124–129. Cited on page 91.

Engleberg, I. N. and Wynn, D. R. (2006). *Working in Groups: Communication Principles and Strategies*. Allyn & Bacon, 4 edition. Cited on page 81.

Ervin-Tripp, S. (1976). Is Sybil there? The structure of some American English directives. *Language in Society*, 5(01):25–66. Cited on page 159.

Faber, F., Bennewitz, M., Eppner, C., Görög, A., Gonsior, C., Joho, D., Schreiber, M., and Behnke, S. (2009). The humanoid museum tour guide Robotinho. In *The 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 891–896. Cited on page 15.

Falcone, E., Gockley, R., Porter, E., and Nourbakhsh, I. (2003). The personal rover project: The comprehensive design of a domestic personal robot. *Robotics and Autonomous Systems*, 42(3–4):245–258. Socially Interactive Robots. Cited on page 14.

Farhadi, A., Endres, I., Hoiem, D., and Forsyth, D. (2009). Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1778–1785. Cited on page 33.

Feil-Seifer, D., Skinner, K., and Matarić, M. J. (2007). Benchmarks for evaluating socially assistive robotics. *Interaction Studies*, 8(3):423–439. Cited on page 12.

Ferrein, A. (2010a). golog.lua: Towards a non-prolog implementation of GOLOG for embedded systems. In Hoffmann, G., editor, *Proceedings of the AAAI Spring Symposium on Embedded Reasoning*, (SS-10-04), pages 20–28. AAAI Press. Cited on page 61.

Ferrein, A. (2010b). Robot controllers for highly dynamic environments with real-time constraints. *KI - Künstliche Intelligenz*, 24(2):175–178. Cited on page 60.

Ferrein, A., Fritz, C., and Lakemeyer, G. (2004). On-line decision-theoretic golog for unpredictable domains. In Biundo, S., Frühwirth, T., and Palm, G., editors, *KI 2004: Advances in Artificial Intelligence*, volume 3238 of *Lecture Notes in Computer Science*, pages 322–336. Springer Berlin Heidelberg. Cited on page 20.

Ferrein, A., Fritz, C., and Lakemeyer, G. (2005). Using GOLOG for deliberation and team coordination in robotic soccer. *KI Künstliche Intelligenz*, 19(1):24–43. Cited on page 20.

Ferrein, A. and Lakemeyer, G. (2008). Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems*, 56(11):980–991. Cited on pages 16, 20, 35, 53, 60, 61, 93, 95, 102, and 110.

Ferrein, A., Lakemeyer, G., and Schiffer, S. (2006). AllemaniACs@Home 2006 Team Description. In *Proceedings CD RoboCup 2006*. RoboCup Federation. Cited on pages 6 and I.

Ferrein, A., Schiffer, S., and Lakemeyer, G. (2008). A Fuzzy Set Semantics for Qualitative Fluents in the Situation Calculus. In *Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA'08)*, volume 5314 of *Lecture Notes in Computer Science*, pages 498–509. Springer. Cited on pages 5, 97, and I.

Ferrein, A., Schiffer, S., and Lakemeyer, G. (2009). Embedding fuzzy controllers into golog. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'09)*, pages 894–899. IEEE. Cited on pages 5, 97, and I.

Ferrein, A. and Steinbauer, G. (2010). On the way to high-level programming for resource-limited embedded systems with GOLOG. In Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., and von Stryk, O., editors, *Proceedings of the 2nd International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2010)*, volume 6472 of *Lecture Notes in Computer Science*, pages 229–240. Springer. Cited on page 61.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208. Cited on page 56.

Finzi, A. and Orlandini, A. (2005). Human-robot interaction through mixed-initiative planning for rescue and search rovers. In Bandini, S. and Manzoni, S., editors, *AI*IA 2005: Advances in Artificial Intelligence*, volume 3673 of *Lecture Notes in Computer Science*, pages 483–494. Springer Berlin Heidelberg. Cited on page 143.

Finzi, A. and Pirri, F. (2004). Flexible interval planning in concurrent temporal golog. In *Working notes of the 4th Int. Cognitive Robotics Workshop*. Cited on page 140.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17:760–772. Cited on page 21.

Fong, T., Nourbakhsh, I., and Dautenhahn, K. (2003a). A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3–4):143–166. Socially Interactive Robots. Cited on page 63.

Fong, T., Thorpe, C., and Baur, C. (2003b). Collaboration, dialogue, human-robot interaction. In *Robotics Research*, volume 6 of *Springer Tracts in Advanced Robotics*, pages 255–266. Springer. Cited on page 158.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33. Cited on pages 21, 22, and 28.

Fox, D., Burgard, W., Thrun, S., and Cremers, A. B. (1998). Position Estimation for Mobile Robots in Dynamic Environments. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI'98/IAAI'98)*, pages 983–988, Menlo Park, CA, USA. American Association for Artificial Intelligence. Cited on page 30.

Francke, H., Ruiz-del Solar, J., and Verschae, R. (2007). Real-time hand gesture detection and recognition using boosted classifiers and active learning. In *Proceedings of the 2nd Pacific Rim Conference on Advances in Image and Video Technology*, pages 533–547. Springer. Cited on pages 82 and 84.

Freksa, C. (1992). Using Orientation Information for Qualitative Spatial Reasoning. In A. U. Frank, I. Campari, U. F., editor, *Theories and methods of spatio-temporal reasoning in geographic space*, pages 162–178. Springer, Berlin. Cited on page 99.

Freksa, C. and Zimmermann, K. (1992). On the utilization of spatial structures for cognitively plausible and efficient reasoning. In *IEEE International Conference on Systems Man and Cybernetics*, pages 261–266, Chicago. Cited on pages 98 and 99.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. System Sci.*, 55:119–139. Cited on page 83.

Gates, B. (2007). A Robot in Every Home. *Scientific American*, 296:58–65. Cited on pages 2 and 7.

De Giacomo, G., Lespérance, Y., and Levesque, H. J. (1997). Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In *Proceedings of the Fifteenth International Joint Conference on AI (IJCAI'97)*, pages 1221–1226, Nagoya. Cited on page 58.

De Giacomo, G., Lespérance, Y., and Levesque, H. J. (2000). ConGolog, A Concurrent Programming Language based on the Situation Calculus. *Artificial Intelligence*, 121(1-2):109–169. Cited on pages 59, 60, 141, and 146.

De Giacomo, G. and Levesque, H. (1999). An incremental interpreter for high-level programs with sensing. In Levesque, H. J. and Pirri, F., editors, *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, Artificial Intelligence, pages 86–102. Springer Berlin Heidelberg. Cited on pages 58 and 59.

De Giacomo, G., Levesque, H., and Sardiña, S. (2001). Incremental Execution of Guarded Theories. *ACM Transactions on Computational Logic (TOCL)*, 2(4):495–525. Cited on pages 59 and 60.

Goodrich, M. A. and Schultz, A. C. (2007). Human-robot interaction: A survey. *Found. Trends Hum.-Comput. Interact.*, 1(3):203–275. Cited on page 63.

Görz, G. and Ludwig, B. (2005). Speech Dialogue Systems – A Pragmatics-Guided Approach to Rational Interaction. *KI–Künstliche Intelligenz*, 10(3):5–10. Cited on page 158.

Goth, G. (2011). I, domestic robot. *Communications of the ACM*, 54(5):16–17. Cited on page 15.

Graf, B., Hans, M., and Schraft, R. D. (2004). Care-O-bot II – Development of a Next Generation Robotic Home Assistant. *Autonomous Robots*, 16(2):193–205. Cited on page 14.

Graf, B., Reiser, U., Hägele, M., Mauz, K., and Klein, P. (2009). Robotic home assistant Care-O-bot ® 3 - product vision and innovation platform. In *Proceedings of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 139–144. Cited on page 14.

Grandhi, S. A., Joue, G., Mittelberg, I., and Jarke, M. (2010). Designing Touchless Gesture-Based Interfaces for Human Computer Interaction: Insights from co-verbal gestures. In *SIGHCI 2010 Proceedings*. Cited on page 89.

Grosskreutz, H. (2000). Probabilistic projection and belief update in the pGOLOG framework. In *Proceedings of the 2nd Cognitive Robotics Workshop (CogRob'00) at the 14th European Conference on Artificial Intelligence (ECAI'2000)*, pages 34–41. Cited on page 60.

Grosskreutz, H. (2002). *Towards More Realistic Logic-Based Robot Controllers in the GOLOG Framework*. PhD thesis, RWTH Aachen University. Cited on page 59.

Grosskreutz, H. and Lakemeyer, G. (2001). On-Line Execution of cc-Golog Plans. In *IJCAI*, pages 12–18. Cited on page 60.

Grosskreutz, H. and Lakemeyer, G. (2003). cc-Golog - An Action Language with Continous Change. *Logic Journal of the IGPL*, 11(2):179–221. Cited on page 60.

Gu, Y. and Soutchanski, M. (2008). Reasoning about Large Taxonomies of Actions. In Fox, D. and Gomes, C. P., editors, *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, volume 2, pages 931–937. AAAI Press. Cited on pages 146 and 161.

Guo, G., Li, S. Z., and Chan, K. (2000). Face recognition by support vector machines. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000 (FG '00)*, page 196, Washington, DC, USA. IEEE Computer Society. Cited on pages 75 and 79.

Hähnel, D., Burgard, W., and Lakemeyer, G. (1998). GOLEX: Bridging the gap between logic (GOLOG) and a real robot. In Herzog, O. and Günter, A., editors, *KI-98: Advances in Artificial Intelligence*, volume 1504 of *Lecture Notes in Computer Science*, pages 165–176. Springer Berlin Heidelberg. Cited on page 57.

Hanebeck, U., Fischer, C., and Schmidt, G. (1997). Roman: a mobile robotic assistant for indoor service applications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 518–525. Cited on page 14.

Harrison, J. R. (1996). Theorem proving with the real numbers. Technical Report UCAM-CL-TR-408, University of Cambridge, Computer Laboratory. Cited on page 104.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107. Cited on page 20.

Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., and Schölkopf, B. (1998). Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28. Cited on page 79.

Hegel, F., Gieselmann, S., Peters, A., Holthaus, P., and Wrede, B. (2011). Towards a typology of meaningful signals and cues in social robotics. In *Proceedings of the 20th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 72–78. Cited on page 14.

Helmer, S., Meger, D., Viswanathan, P., McCann, S., Dockrey, M., Fazli, P., Southey, T., Muja, M., Joya, M., Little, J., Lowe, D. G., and Mackworth, A. K. (2009). Semantic robot vision challenge: Current state and future directions. In *Proceedings of the IJCAI 2009 Workshop on Competitions in Artificial Intelligence and Robotics*. Cited on page 13.

Hernandez, D. (1991). Relative Representation of Spatial Knowledge: The 2-D Case. In Mark, D. M. and Frank, A. U., editors, *Cognitive and Linguistic Aspects of Geographic Space*, pages 373–385. Kluwer, Dordrecht. Cited on pages 44, 99, and 122.

Hernandez, D., Clementini, E., and Felice, P. D. (1995). Qualitative distances. In Kuhn, W. and Frank, A., editors, *Spatial Information Theory: a theoretical basis for GIS*, number 988 in LNCS, pages 45–58. Springer-Verlag. Cited on pages 44, 99, and 122.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 278–282. Cited on pages 38 and 83.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844. Cited on page 38.

Homer (1924). *The Iliad*. Harvard University Press, Cambridge, MA. with an English Translation by A.T. Murray, Ph.D. in two volumes. Cited on page 1.

Honda (2002). We're building a dream, one robot at a time. Advertisement in Time Magazine, pages 52–53. Cited on page 2.

Honda (2003). We're building a dream, one robot at a time. Advertisement in Smithsonian Magazine, back cover. Cited on page 2.

Hoppe, N. (2011). Flexible command interpretation on a mobile robot using readylog. Diplomarbeit, RWTH Aachen University, Aachen, Germany. Cited on page 6.

Howard, R. A. (1960). *Dynamic programming and Markov processes*. MIT Press, Cambridge, MA. Cited on pages 40 and 41.

Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y., and Rosenfeld, R. (1993). The SPHINX-II speech recognition system: an overview. *Computer Speech and Language*, 7(2):137–148. Cited on pages 66 and 67.

Hüttenrauch, H. and Severinson Eklundh, K. (2002). Fetch-and-carry with CERO: observations from a long-term user study with a service robot. In *Proceedings of the 11th IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN*, pages 158–163. Cited on page 14.

Inamura, T., Tan, J., Sugiura, K., Nagai, T., and Okada, H. (2014). Development of robocup@home simulation towards long-term large scale hri. In Behnke, S., Veloso, M., Visser, A., and Xiong, R., editors, *RoboCup 2013: Robot World Cup XVII*, volume 8371 of *Lecture Notes in Computer Science*, pages 672–680. Springer Berlin Heidelberg. Cited on page 15.

Isermann, R. (1998). On fuzzy logic applications for automatic control, supervision, and fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(2):221–235. Cited on page 101.

Jacobs, S., Ferrein, A., and Lakemeyer, G. (2005). Controlling Unreal Tournament 2004 bots with the logic-based action language GOLOG. In Young, R. M. and Laird, J. E., editors, *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, pages 151–152. AAAI Press. Cited on page 61.

Jacobs, S., Ferrein, A., Schiffer, S., Beck, D., and Lakemeyer, G. (2009). Robust Collision Avoidance in Unknown Domestic Environments. In *Proceedings of the International RoboCup Symposium 2009 (RoboCup 2009)*, volume 5949 of *LNCS*, pages 116–127. Springer. Cited on pages 5 and I.

Ji, J. and Chen, X. (2011). Induction in nonmonotonic causal theories for a domestic service robot. In *Proceedings of the 21st International Conference on Inductive Logic Programming (ILP 2011)*. Cited on page 16.

Ji, J., Sui, Z., Jin, G., Xie, J., and Chen, X. (2013). Simulation competitions on domestic robots. In Chen, X., Stone, P., Sucar, L., and van der Zant, T., editors, *RoboCup 2012: Robot Soccer World Cup XVI*, volume 7500 of *Lecture Notes in Computer Science*, pages 166–177. Springer Berlin Heidelberg. Cited on page 15.

Jones, M. and Rehg, J. (2002). Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96. Cited on pages 82 and 84.

Jones, M. and Viola, P. (2003). Face recognition using boosted local features. In *Proceedings of International Conference on Computer Vision*. Cited on pages 74, 75, 76, 77, 78, 79, 84, and 177.

Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transaction on Robotics and Automation*, 12(4):566–580. Cited on page 21.

Khatib, M., Bouilly, B., Simeon, T., and Chatila., R. (1997). Indoor navigation with uncertainty using sensor-based motions. In *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA-97)*, pages 3379–3384. Cited on page 21.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997a). RoboCup: The

Robot World Cup Initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 340–347, New York, NY, USA. ACM. Cited on page 9.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., and Matsubara, H. (1997b). RoboCup: A challenge problem for AI. *AI magazine*, 18(1):73–85. Cited on page 13.

Kitano, H. and Tadokoro, S. (2001). RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems. *AI Magazine*, 22(1):39–52. Cited on page 13.

Koenig, S. and Likhachev, M. (2002). Improved fast replanning for robot navigation in unknown terrain. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA-02)*, pages 968–975. Cited on page 22.

Kölsch, M. and Turk, M. (2004a). Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration. In *IEEE Workshop on Real-Time Vision for Human-Computer Interaction at Conf. on Computer Vision and Pattern Recognition*, page 158. Cited on pages 83 and 88.

Kölsch, M. and Turk, M. (2004b). Robust hand detection. In *Proceedings of the 6th IEEE International Conference on Automatic Face and Gesture Recognition*, pages 614–619. Cited on pages 82, 83, 87, 88, 90, and 177.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97. Cited on page 91.

Kumar, A. (2008). Incorporating cohort information for reliable palmprint authentication. In *Proceedings of the Sixth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP'08)*, pages 583–590. Cited on page 86.

Lakani, S. R. (2013). A flexible object recognition system for domestic domains. Master's thesis, RWTH Aachen University, Aachen, Germany. Cited on pages 6 and 33.

Lamel, L., Rabiner, L., Rosenberg, A., and Wilpon, J. (Aug 1981). An improved endpoint detector for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing]*, 29(4):777–785. Cited on page 67.

Large, F., Sekhavat, S., Shiller, Z., and Laugier, C. (2002). Towards real-time global motion planning in a dynamic environment using the NLVO concept. In *Proceedings of the 2002 IEEE/RSJ International Conference on Robots and Systems (IROS)*, pages 607–612. Cited on page 21.

Lemai, S. and Ingrand, F. (2004). Interleaving temporal planning and execution in robotics domains. In *AAAI'04:Proceedings of the 19th National Conference on Artifical Intelligence*, pages 617–622. AAAI Press / The MIT Press. Cited on pages 16, 142, and 146.

Lespérance, Y., Tam, K., and Jenkin, M. (2000). Reactivity in a logic-based robot programming framework. In Jennings, N. R. and Lespérance, Y., editors, *Intelligent Agents VI. Agent Theories, Architectures, and Languages*, volume 1757 of *Lecture Notes in Computer Science*, pages 173–187. Springer Berlin Heidelberg. Cited on page 143.

Levesque, H., Pirri, F., and Reiter, R. (1998). Foundations for a calculus of situations. *Electronic Transactions of AI (ETAI)*, 2(3–4):159–178. Cited on pages 53 and 56.

Levesque, H. and Reiter, R. (1998). High-level robotic control: Beyond planning. a position paper. In *AIII 1998 Spring Symposium: Integrating Robotics Research: Taking the Next Big Leap*. Cited on page 2.

Levesque, H. J. and Lakemeyer, G. (2008). Cognitive Robotics. In van Harmelen, F., Lifschitz, V., and Porter, B., editors, *Handbook of Knowledge Representation*, chapter 23, pages 869–886. Elsevier. Cited on page 2.

Levesque, H. J., Reiter, R., Lespérance, Y., Lin, F., and Scherl, R. B. (1997). GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31(1–3):59–84. Cited on pages 35, 53, 56, 60, 102, and 110.

Lin, Q., Lubensky, D., Picheny, M., and Rao, P. S. (1997). Key-phrase spotting using an integrated language model of n-grams and finite-state grammar. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH-1997)*, pages 255–258. Cited on page 67.

Liu, H. (2009). A fuzzy qualitative framework for connecting robo qualitative and quantitative representations. *IEEE Transactions on Fuzzy Systems*, 16(6):1522–1530. Cited on page 100.

Liu, H., Brown, D. J., and Coghill, G. M. (2008). Fuzzy qualitative robot kinematics. *IEEE Transactions on Fuzzy Systems*, 16(3):808–822. Cited on page 100.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110. Cited on page 33.

Macho, D., Padrell, J., Abad, A., Nadeu, C., Hernando, J., McDonough, J., Wolfel, M., Klee, U., Omologo, M., Brutti, A., Svaizer, P., Potamianos, G., and Chu, S. (6-6 July 2005). Automatic speech activity detection, source localization, and speech recognition on the chil seminar corpus. *IEEE Int. Conf. on Multimedia and Expo, 2005 (ICME 2005)*, pages 876–879. Cited on page 67.

Marée, R., Geurts, P., and Wehenkel, L. (2007). Random subwindows and extremely randomized trees for image classification in cell biology. *Workshop of Multiscale Biological Imaging, Data Mining and Informatics*. Cited on page 74.

Matas, J. and Obdržálek, Š. (2004). Object Recognition Methods Based on Transformation Covariant Features. In *Proceedings of the XII European Signal Processing Conference (EUSIPCO)*, pages 1333–1336. Cited on page 39.

McCarthy, J. (1963). Situations, Actions, and Causal Laws. Technical Report Memo 2, AI Lab, Stanford University, California, USA. Published in Semantic Information Processing, ed. Marvin Minsky. Cambridge, MA: The MIT Press, 1968. Cited on pages 35, 53, and 102.

McCarthy, J. and Hayes, P. J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press. reprinted in McC90. Cited on pages 53 and 54.

McDermott, D. and Davis, E. (1984). Planning routes through uncertain territory. *Artificial Intelligence*, 22(2):107–156. Cited on page 101.

Mendel, J. (1995). Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377. Cited on pages 42 and 101.

Meystel, A., Guez, A., and Hillel, G. (1986). Minimum time path planning for a robot. In *Proc. of the 1986 IEEE International Conference on Robotics and Automation (ICRA'86)*, pages 1678–1687. Cited on page 24.

Minguez, J. and Montano, L. (2004). Nearness Diagram Navigation (ND): Collision Avoidance in Troublesome Scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59. Cited on page 21.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics (Part C)*, 37(3):311–324. Cited on page 83.

Montemerlo, D., Roy, N., and Thrun, S. (2003). Perspectives on standardization in mobile robot programming: the Carnegie Mellon Navigation (CARMEN) Toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2436–2441. Cited on page 30.

Moratz, R., Dylla, F., and Frommberger, L. (2005). A relative orientation algebra with adjustable granularity. In *Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments (IJCAI 05)*. Cited on page 99.

Moravec, H. P. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 116–121. Cited on pages 20 and 23.

Müller, R., Röfer, T., Lankenau, A., Musto, A., Stein, K., and Eisenkolb, A. (2000). Coarse qualitative descriptions in robot navigation. In *Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, pages 265–276, London, UK. Springer-Verlag. Cited on page 99.

Murphy, R. R. (2000). *Introduction to AI Robotics*. MIT Press, Cambridge, MA, USA, 1st edition. Cited on page 1.

Murray, D. and Little, J. (2000). Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171. Cited on page 22.

Musto, A., Stein, K., Eisenkolb, A., Röfer, T., Brauer, W., and Schill, K. (2000). From motion observation to qualitative motion representation. In *Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*, pages 115–126, London, UK. Springer-Verlag. Cited on pages 99 and 100.

Nakamura, T., Araki, T., Nagai, T., and Iwahashi, N. (2011). Grounding of word meanings in latent dirichlet allocation-based multimodal concepts. *Advanced Robotics*, 25(17):2189–2206. Cited on page 15.

Nardi, D., Dessimoz, J.-D., Dominey, P. F., Gossow, D., Holz, D., Iocchi, L., Kraetzschmar, G., Mahmoudi, F., Olufs, S., Rascon, C., Rybski, P. E., Savage, J., Schiffer, S., Sugiura, K., Wachsmuth, S., Wisspeintner, T., van der Zant, T., and Yazdani, A. (2014). Robocup@home – rules & regulations. Technical report, RoboCup@Home Technical Committee. Cited on pages 9 and 11.

Nass, C., Steuer, J., and Tauber, E. R. (1994). Computers are social actors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 72–78, New York, NY, USA. ACM. Cited on page 14.

Nickel, K. and Stiefelhagen, R. (2007). Visual recognition of pointing gestures for human–robot interaction. *Image and Vision Computing*, 25(12):1875–1884. Cited on page 83.

Niemueller, T., Ferrein, A., Beck, D., and Lakemeyer, G. (2010a). Design principles of the component-based robot software framework fawkes. In Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., and von Stryk, O., editors, *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, volume 6472 of *Lecture Notes in Computer Science*, pages 300–311. Springer Berlin Heidelberg. Cited on pages 18 and 93.

Niemueller, T., Ferrein, A., and Lakemeyer, G. (2010b). A lua-based behavior engine for controlling the humanoid robot nao. In Baltes, J., Lagoudakis, M., Naruse, T., and Ghidary, S., editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 240–251. Springer Berlin Heidelberg. Cited on page 20.

Niemueller, T., Schiffer, S., Lakemeyer, G., and Rezapour-Lakani, S. (2013). Life-long learning perception using cloud database technology. In *Proceedings of the 2013 IROS Workshop on Cloud Robotics*. Cited on pages 5, 33, 34, and I.

Nilsson, N. J. (1984). Shakey the robot. Technical Report 323, DTIC Document. Cited on page 1.

Nordvik, J.-P., Smets, P., and Magrez, P. (1988). Fuzzy qualitative modeling. In Bouchon, B., Saitta, L., and Yager, R., editors, *Proceedings of the 2nd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems: Uncertainty and Intelligent Systems (IPMU-88)*, volume 313 of *Lecture Notes in Computer Science*, pages 231–238. Springer. Cited on page 41.

Okada, H., Iwahashi, N., Tan, J. T. C., Sugiura, K., Nagai, T., Nakamura, T., Hagiwara, Y., and Inamura, T. (2014). Team er@sers 2014 in the @home league team description paper. Technical report, Tamagawa University, Tokyo and Kyoto University and University of Tokyo and National Institute of Information and Communications Technology and The University of Electro-Communications and National Institute of Informatics, JAPAN. Cited on page 15.

Padrell, J., Macho, D., and Nadeu, C. (2005). Robust Speech Activity Detection Using LDA Applied to FF Parameters. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, volume 1, pages 557–560. Cited on page 67.

Panin, G., Klose, S., and Knoll, A. (2009). Real-time articulated hand detection and pose estimation. In *Proceedings of the 5th International Symposium on Advances in Visual Computing (ISVC '09)*, pages 1131–1140, Berlin, Heidelberg. Springer-Verlag. Cited on page 83.

Paredes, R., Perez-Cortes, J., Juan, A., and Vidal, E. (2001). Local representations and a direct voting scheme for face recognition. In *Workshop on Pattern Recognition in Information Systems*, pages 71–79, Setúbal, Portugal. Cited on page 74.

Park, C.-B. and Lee, S.-W. (2011). Real-time 3D pointing gesture recognition for mobile robots with cascade HMM and particle filter. *Image and Vision Computing*, 29(1):51–63. Cited on page 83.

Parlitz, C., Baum, W., Reiser, U., and Hägele, M. (2007). Intuitive human-machine-interaction and implementation on a household robot companion. In Smith, M. and Salvendy, G., editors, *Human Interface and the Management of Information. Methods, Techniques and Tools in Information Design*, volume 4557 of *Lecture Notes in Computer Science*, pages 922–929. Springer Berlin Heidelberg. Cited on page 14.

Passino, K. M. and Yurkovich, S. (1998). *Fuzzy Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition. Cited on pages 43, 101, and 115.

Petti, S. and Fraichard, T. (2005). Safe motion panning in dynamic environemts. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and System (IROS-05)*, pages 2210–2215. Cited on page 21.

Pham, M. and Cham, T. (2007). Fast training and selection of haar features using statistics in boosting-based face detection. *IEEE 11th International Conference on Computer Vision*, pages 1–7. Cited on pages 75 and 78.

Piaget, J. (1955). The construction of reality in the child. *Journal of Consulting Psychology*, 19(1):77. Cited on page 99.

Pineau, J., Montemerlo, M., Pollack, M., Roy, N., and Thrun, S. (2003). Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3–4):271–281. Socially Interactive Robots. Cited on page 2.

Pirri, F. and Reiter, R. (1999). Some Contributions to the Metatheory of the Situation Calculus. *Journal of the ACM*, 46(3):325–361. Cited on pages 55, 56, and 144.

Plato (1967). *Plato in Twelve Volumes*, volume 3, translated by W.R.M. Lamb. Harvard University Press, Cambridge, MA. Cited on page 1.

Plöger, P.-G., Pervölz, K., Mies, C., Eyerich, P., Brenner, M., and Nebel, B. (2008). The DESIRE Service Robotics Initiative. *KI*, 22(4):29–32. Cited on page 15.

del Pobil, A. (2006). Why do We Need Benchmarks in Robotics Research? In *Proceedings of the 2006 IROS Workshop on Benchmarks in Robotics Research*. IEEE/RSJ. Cited on page 12.

Pommerening, F., Wölfl, S., and Westphal, M. (2009). Right-of-Way Rules as Use Case for Integrating GOLOG and Qualitative Reasoning . In *Proceedings of the 32nd Annual German Conference on AI (KI 2009)*, pages 468–475. Springer-Verlag New York Inc. Cited on page 100.

Prassler, E., Zöllner, M., Bischoff, R., Burgard, W., Haschke, R., Hägele, M., Lawitzky, G., Nebel, B., Plöger, P., and Reiser, U., editors (2012). *Towards Service Robots for Everyday Environments*, volume 76 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg. Cited on page 15.

Preparata, F. P. and Shamos, M. I. (1985). *Computational geometry: An Introduction*. Springer-Verlag, New York, NY, USA. Cited on page 46.

Puterman, M. (1994). *Markov Decision Processes: Discrete Dynamic Programming*. Wiley, New York, USA. Cited on pages 39 and 41.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106. Cited on page 37.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA. Cited on page 37.

Quinlan, J. R. (1996). Bagging, Boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI)*, pages 725–730. AAAI Press. Cited on page 39.

Randell, D. A., Cui, Z., and Cohn, A. (1992). A Spatial Logic Based on Regions and Connection. In Nebel, B., Rich, C., and Swartout, W., editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 165–176. Morgan Kaufmann, San Mateo, California. Cited on page 98.

Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, USA. Cited on page 54.

Reiter, R. (1996). Natural actions, concurrency and continuous time in the situation calculus. In *In Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, pages 2–13, Cambridge, Massachusetts, U.S.A. Cited on page 147.

Reiter, R. (2001). *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, Massachusetts. Cited on pages 35, 53, 55, 56, and 112.

Rentzeperis, E., Stergiou, A., Boukis, C., Souretis, G., Pnevmatikakis, A., and Polymenakos, L. (2006). An Adaptive Speech Activity Detector Based on Signal Energy and LDA. In *3rd Joint Workshop on Multi-Modal Interaction and Related Machine Learning Algorithms*. Cited on page 67.

Renz, J. and Nebel, B. (2007). Qualitative spatial reasoning using constraint calculi. *Handbook of Spatial Logics*, pages 161–215. Cited on page 98.

Robinson, V. B. (2000). Individual and multipersonal fuzzy spatial relations acquired using human-machine interaction. *Fuzzy Sets and Systems*, 113(1):133–145. Cited on pages 138 and 177.

Rosheim, M. E. (1994). *Robot Evolution: The Development of Anthrobotics*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition. Cited on page 1.

Rosheim, M. E. (1997). In the footsteps of leonardo. *Robotics Automation Magazine*, 4(2):12–14. Cited on page 1.

Rowley, H., Baluja, S., and Kanade, T. (1998a). Rotation invariant neural network-based face detection. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, page 963, Washington, DC, USA. IEEE Computer Society. Cited on page 75.

Rowley, H. A., Baluja, S., and Kanade, T. (1998b). Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38. Cited on pages 75 and 78.

Ruhi Sarikaya, J. H. L. H. (1998). Robust Speech Activity Detection in the Presence of Noise. In *Proc. of the 5th Int. Conf. on Spoken Language Processing, (Inc. the 7th Australian Int. Speech Science and Technology Conf.* Cited on page 67.

Ruiz-del Solar, J., Mascaró, M., Correa, M., Bernuy, F., Riquelme, R., and Verschae, R. (2010). Analyzing the human-robot interaction abilities of a general-purpose social robot in different naturalistic environments. In Baltes, J., Lagoudakis, M., Naruse, T., and Ghidary, S., editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science*, pages 308–319. Springer Berlin Heidelberg. Cited on page 14.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition. Cited on pages 37 and 38.

Rusu, R., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3D recognition and pose using the Viewpoint Feature Histogram. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2155–2162. IEEE. Cited on page 33.

Rybski, P. E., Forbes, J., Burhans, D. T., Dodds, Z., Oh, P., Scheutz, M., and Avanzato, B. (2007). The AAAI 2006 mobile robot competition and exhibition. *AI Magazine*, 28(2):101–114. Cited on page 12.

Sabanovic, S., Michalowski, M., and Simmons, R. (2006). Robots in the wild: observing human-robot social interaction outside the lab. In *9th IEEE International Workshop on Advanced Motion Control*, pages 596–601. Cited on page 12.

Saffiotti, A. (1997). Fuzzy logic in autonomous robotics: behavior coordination. In *Proceedings of the Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97)*, pages 573–578 vol.1. IEEE Computer Society Press. Cited on page 100.

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., and Fujimura, K. (2002). The intelligent asimo: system overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2478–2483. Cited on page 14.

Saxe, D. and Foulds, R. (1996). Toward robust skin identification in video images. In *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, page 379. Cited on pages 82 and 84.

Schermerhorn, P., Kramer, J., Brick, T., Anderson, D., Dingler, A., and Scheutz, M. (2006a). DIARC: A testbed for natural human-robot interaction. In Avanzato, B., Rybski, P., and Forbes, J., editors, *Proceedings of the AAAI Workshop on the AAAI Mobile Robot Competition and Exhibition*, volume Technical Report WS-06-15, pages 45–52, Menlo Park, California. The AAAI Press. Cited on page 14.

Schermerhorn, P., Kramer, J., Middendorff, C., and Scheutz, M. (2006b). DIARC: A testbed for natural human-robot interaction. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, volume 2, pages 1972–1973. The AAAI Press. Cited on page 14.

Scheutz, M., Schermerhorn, P., Kramer, J., and Anderson, D. (2007). First steps toward natural human-like hri. *Autonomous Robots*, 22(4):411–423. Cited on page 63.

Schiffer, S. (2005). A qualitative worldmodel for autonomous soccer agents in the readylog framework. Diploma thesis, Knowledge-based Systems Group, Computer Science Department, RWTH Aachen University, Aachen, Germany. Cited on page 97.

Schiffer, S., Baumgartner, T., Beck, D., Maleki-Fard, B., Niemueller, T., Schwering, C., and Lakemeyer, G. (2012a). robOCD: Robotic Order Cups Demo – An Interactive Domestic Service Robotics Demo. In Wölfl, S., editor, *Poster and Demo Session at the*

35th German Conference on Artificial Intelligence (KI 2012), pages 150–154. Cited on pages 5, 63, 93, and I.

Schiffer, S., Baumgartner, T., and Lakemeyer, G. (2011a). A modular approach to gesture recognition for interaction with a domestic service robot. In Jeschke, S., Liu, H., and Schilberg, D., editors, *Intelligent Robotics and Applications*, volume 7102 of *Lecture Notes in Computer Science*, pages 348–357. Springer Berlin / Heidelberg. Cited on pages 5, 63, and I.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2006a). Football is coming home. In Chen, X., Liu, W., and Williams, M.-A., editors, *Proceedings of the 2006 International Symposium on Practical Cognitive Agents and Robots (PCAR'06)*, pages 39–50, New York, NY, USA. ACM. Cited on pages 5, 6, and I.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2006b). Qualitative world models for soccer robots. In Wölfl, S. and Mossakowski, T., editors, *Qualitative Constraint Calculi, Workshop at KI 2006, Bremen*, pages 3–14. Cited on pages 97 and 102.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2007). AllemaniACs@Home 2007 Team Description. In *Proceedings CD RoboCup 2007*. Cited on pages 6 and I.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2010a). Fuzzy Representations and Control for Domestic Service Robots in Golog. In Iocchi, L., del Solar, J. R., and van der Zant, T., editors, *Domestic Service Robots in the Real World. Workshop Proceedings of the International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2010)*, pages 183–192, Darmstadt, Germany. Cited on pages 5, 97, and I.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2011b). Fuzzy representations and control for domestic service robots in golog. In Jeschke, S., Liu, H., and Schilberg, D., editors, *Proceedings of the Fourth International Conference on Intelligent Robotics and Applications (ICIRA 2011)*, volume 7102 of *Lecture Notes in Computer Science*, pages 241–250. Springer Berlin / Heidelberg. Cited on pages 5, 97, and I.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2012b). CAESAR – An Intelligent Domestic Service Robot. *Journal of Intelligent Service Robotics. Special Issue on Artificial Intelligence in Robotics: Sensing, Representation and Action*, pages 1–15. Cited on page I.

Schiffer, S., Ferrein, A., and Lakemeyer, G. (2012c). Reasoning with Qualitative Positional Information for Domestic Domains in the Situation Calculus. *Journal of Intelligent and Robotic Systems. Special Issue on Domestic Service Robots in the Real World.*, 66(1–2):273–300. Cited on pages 5, 97, and II.

Schiffer, S., Goeckel, T., Lakemeyer, G., and Wagner, H. (2014). ViCToR – Combining Sound-Source Localization and Face Detection for a Video Conferencing Tool Robot. Technical report, Knowledge-based Systems Group and Institute of Biology II, RWTH Aachen University, Aachen, Germany. Cited on page 177.

Schiffer, S., Hoppe, N., and Lakemeyer, G. (2012d). Flexible command interpretation on an interactive domestic service robot. In Filipe, J. and Fred, A., editors, *Proceedings of the 4th International Conference on Agents and Artificial Intelligence (ICAART 2012)*, volume 1 - Artificial Intelligence, pages 26–35. SciTePress. Best Student Paper Award. Cited on pages 6 and II.

Schiffer, S., Hoppe, N., and Lakemeyer, G. (2013a). Natural language interpretation for an interactive service robot in domestic domains. In Filipe, J. and Fred, A., editors, *Agents and Artificial Intelligence*, volume 358 of *Communications in Computer and Information Science*, pages 39–53. Springer Berlin Heidelberg. Cited on pages 6 and II.

Schiffer, S. and Lakemeyer, G. (2008). AllemaniACs@Home 2008 Team Description. In *Proceedings CD RoboCup 2008*. Cited on pages 6 and I.

Schiffer, S. and Lakemeyer, G. (2011). AllemaniACs@Home 2011 Team Description. Technical report, Knowledge-based Systems Group, RWTH Aachen University. Cited on pages 6 and I.

Schiffer, S., Niemueller, T., Doostdar, M., and Lakemeyer, G. (2009). AllemaniACs@Home 2009 Team Description. In *Proceedings CD RoboCup 2009*. Cited on pages 5, 6, 77, and I.

Schiffer, S., Niemueller, T., Lakani, S. R., and Lakemeyer, G. (2013b). A Flexible Object Recognition System with Hierarchical Classifiers. Technical report, Knowledge-based Systems Group, RWTH Aachen University, Aachen, Germany. Cited on pages 33 and 34.

Schiffer, S., Wortmann, A., and Lakemeyer, G. (2010b). Self-Maintenance for Autonomous Robots controlled by ReadyLog. In Ingrand, F. and Guiochet, J., editors, *Proceedings of the 7th IARP Workshop on Technical Challenges for Dependable Robots in Human Environments*, pages 101–107, Toulouse, France. Cited on pages 6 and II.

Schiffer, S., Wortmann, A., and Lakemeyer, G. (2010c). Self-Maintenance for Autonomous Robots in the Situation Calculus. In Lakemeyer, G., Levesque, H. J., and Pirri, F., editors, *Cognitive Robotics*, number 10081 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. Cited on pages 6 and II.

Schockaert, S., Cornells, C., De Cock, M., and Kerre, E. (2006). Fuzzy spatial relations between vague regions. In *Proc. 3rd International IEEE Conference on Intelligent Systems*, pages 221–226. Cited on page 101.

Schultz, A. C. (2001). The 2000 AAAI mobile robot competition and exhibition. *AI Magazine*, 22(1):67. Cited on page 13.

Schwarz, M., Stückler, J., Droeschel, D., Gräve, K., Holz, D., Schreiber, M., and Behnke, S. (2014). NimbRo@Home 2014 Team Description. Technical report, Computer Science Institute VI: Autonomous Intelligent Systems, Rheinische Friedrich-Wilhelms-Universität Bonn. Cited on page 15.

Scowen, R. (1993). Extended BNF – generic base standards. In *Proc. of the Software Engineering Standards Symposium*, pages 25–34. Cited on page 159.

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, London. Cited on page 157.

Seib, V., Giesen, J., Grüntjens, D., and Paulus, D. (2013). Enhancing human-robot interaction by a robot face with facial expressions and synchronized lip movements. In Skala, V., editor, *Proceedings of the 21st International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 70–77. Cited on page 15.

Seib, V., Gossow, D., Vetter, S., and Paulus, D. (2011). Hierarchical multi-robot coordination. In Ruiz-del Solar, J., Chown, E., and Plöger, P., editors, *RoboCup 2010: Robot Soccer World Cup XIV*, volume 6556 of *Lecture Notes in Computer Science*, pages 314–323. Springer Berlin Heidelberg. Cited on page 15.

Seib, V., Memmesheimer, R., Jakowlewa, T., Nagel, D., Gard, N., Dünnebier, D., Eckert, A., Kreckel, E., Kreutz, A., Mykhalchyshyna, I., Rettler, G., Sattler, F., Veith, A., Knauf, M., Peters, A., and Paulus, D. (2014). RoboCup 2014 - homer@UniKoblenz (Germany). Technical report, University of Koblenz-Landau. Cited on page 14.

Seraji, H. and Howard, A. (2002). Behavior-based robot navigation on challenging terrain: A fuzzylogic approach. In *Proc. of the IEEE Transactions on Robotics and Automation (ICRA-02)*, pages 308–321. Cited on page 21.

Severinson-Eklundh, K., Green, A., and Hüttenrauch, H. (2003). Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, 42(3–4):223–234. Socially Interactive Robots. Cited on page 14.

Seymore, K., Chen, S., Doh, S., Eskenazi, M., Gouvea, E., Raj, B., Ravishankar, M., Rosenfeld, R., Siegler, M., Stern, R., and Thayer, E. (1998). The 1997 CMU Sphinx-3 English Broadcast News transcription system. In *Proceedings of the DARPA Speech Recognition Workshop*. Cited on page 71.

Shannon, C. E. and Weaver, W. (1949). The mathematical theory of communication. Cited on page 37.

Shi, J. and Tomasi, C. (1994). Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600. Cited on pages 83 and 88.

Shieber, S. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343. Cited on page 158.

Siepmann, F., Ziegler, L., Kortkamp, M., and Wachsmuth, S. (2014). Deploying a modeling framework for reusable robot behavior to enable informed strategies for domestic service robots. *Robotics and Autonomous Systems*, 62(5):619–631. Special Issue Semantic Perception, Mapping and Exploration. Cited on page 15.

Simmons, R. (1996). The curvature-velocity method for local obstacle avoidance. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 1, pages 1615–1621. Cited on page 21.

Sproull, L., Subramani, M., Kiesler, S., Walker, J. H., and Waters, K. (1996). When the interface is a face. *Human-Computer Interaction*, 11(2):97–124. Cited on page 91.

Srinivasa, S., Berenson, D., Cakmak, M., Collet Romea, A., Dogar, M., Dragan, A., Knepper, R. A., Niemueller, T. D., Strabala, K., Vandeweghe, J. M., and Ziegler, J. (2012). HERB 2.0: Lessons Learned from Developing a Mobile Manipulator for the Home. *Proceedings of the IEEE*, 100(8):1–19. Cited on page 15.

Srinivasa, S. S., Ferguson, D., Helfrich, C., Berenson, D., Collet, A., Diankov, R., Gallagher, G., Hollinger, G., Kuffner, J., and Weghe, M. (2010). HERB: A home exploring robotic butler. *Autonomous Robots*, 28(1):5–20. Cited on page 15.

Stachniss, C. and Burgard, W. (2002). An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-01)*, pages

508–513. Cited on pages 21, 22, and 26.

Steinfeld, A., Fong, T., Kaber, D., Lewis, M., Scholtz, J., Schultz, A., and Goodrich, M. (2006). Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction (HRI'06)*, pages 33–40, New York, NY, USA. ACM. Cited on page 13.

Strack, A., Ferrein, A., and Lakemeyer, G. (2006). Laser-based localization with sparse landmarks. In Bredenfeld, A., Jacoff, A., Noda, I., and Takahashi, Y., editors, *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 569–576. Springer Berlin Heidelberg. Cited on page 30.

Stückler, J. and Behnke, S. (2009). Integrating indoor mobility, object manipulation, and intuitive interaction for domestic service tasks. In *Proceedings of the 9th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2009)*, pages 506–513. Cited on page 16.

Stückler, J., Droeschel, D., Gräve, K., Holz, D., Schreiber, M., Topalidou-Kyniazopoulou, A., Schwarz, M., and Behnke, S. (2013). Increasing flexibility of mobile manipulation and intuitive human-robot interaction in robocup@home. In Behnke, S., Veloso, M. M., Visser, A., and Xiong, R., editors, *RoboCup 2013: Robot World Cup XVII*, volume 8371 of *Lecture Notes in Computer Science*, pages 135–146. Springer. Cited on page 15.

Sugeno, M. and Yasukawa, T. (1993). A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1(1):7–31. Cited on page 41.

Tam, K. B.-L. (1998). Experiments in High-Level Robot Control Using ConGolog — Reactivity, Failure Handling, and Knowledge-Based Search. Master's thesis, York University, Canada. Cited on page 143.

Taylor, P. A., Black, A., and Caley, R. (1998). The architecture of the festival speech synthesis system. In *The Third ESCA Workshop in Speech Synthesis*, pages 147–151, Jenolan Caves, Australia. Cited on page 90.

Thierfelder, S., Seib, V., Lang, D., Häselich, M., Pellenz, J., and Paulus, D. (2011). Robbie: A message-based robot architecture for autonomous mobile systems. In Heiß, H.-U., Pepper, P., Schlingloff, H., and Schneider, J., editors, *INFORMATIK 2011 - Informatik schafft Communities*, volume 191 of *Lecture Notes in Informatics (LNI)*, page 331. Köllen Druck+Verlag GmbH Bonn. Cited on page 15.

Thrun, S. (2002). Robotic mapping: A survey. In Lakemeyer, G. and Nebel, B., editors, *Exploring Artificial Intelligence in the New Millennium*, pages 1–35. Morgan Kaufmann, San Mateo, CA. Cited on page 30.

Thrun, S. (2004). Toward a framework for human-robot interaction. *Human-Computer Interaction*, 19(1):9–24. Cited on pages 1 and 2.

Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (2000). Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. *The International Journal of Robotics Research*, 19(11):972–999. Cited on page 14.

Tikk, D., Biro, G., Gedeon, T., Koczy, L., and Yang, J. D. (2002). Improvements and critique on sugeno's and yasukawa's qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 10(5):596–606. Cited on page 41.

Triesch, J. and von der Malsburg, C. (1996). Robust classification of hand postures against complex backgrounds. In *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, pages 170–175, Washington, DC, USA. Cited on page 83.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86. Cited on page 74.

Ulrich, I. and Borenstein, J. (2000). Vfh*: Local obstacle avoidance with look-ahead verification. In *Proceedings of the 2000 IEEE/RSJ International Conference on Robotics and Automation*, pages 2505–2511. Cited on page 21.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New-York. Cited on page 79.

Viola, P. A. and Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *Conf. on Computer Vision and Pattern Recognition (CVPR 2001)*, volume I, pages 511–518, Los Alamitos, CA, USA. IEEE Computer Society. Cited on page 82.

Viola, P. A. and Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154. Cited on page 84.

Wachs, J. P., Kölsch, M., Stern, H., and Edan, Y. (2011). Vision-based hand-gesture applications. *Communications of the ACM*, 54(2):60–71. Cited on page 82.

Wang, C.-C. and Wang, K.-C. (2008). Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction. In Lee, S., Suh, I. H., and Kim, M. S., editors, *Recent Progress in Robotics: Viable Robotic Service to Human*, volume 370 of *Lecture Notes in Control and Information Sciences*, pages 317–329. Springer Berlin Heidelberg. Cited on page 83.

Wessel, F., Schlüter, R., Macherey, K., and Ney, H. (Mar 2001). Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288–298. Cited on page 68.

Wijesoma, W., Kodagoda, K., and Balasuriya, A. (2002). A laser and a camera for mobile robot navigation. In *7th International Conference on Control, Automation, Robotics and Vision (ICARCV 2002)*, volume 2, pages 740–745. Cited on page 22.

Wisspeintner, T. and van der Zant, T. (2006). RoboCup@Home - Rules & Regulations. Technical report. Cited on page 11.

Wisspeintner, T., van der Zant, T., Iocchi, L., and Schiffer, S. (2009). RoboCup@Home: Scientific Competition and Benchmarking for Domestic Service Robots. *Interaction Studies. Special Issue on Robots in the Wild*, 10(3):392–426. Cited on pages 5, 9, 10, 13, 118, 157, and I.

Wisspeintner, T., van der Zant, T., Iocchi, L., and Schiffer, S. (2010). RoboCup@Home: Results in Benchmarking Domestic Service Robots. In Baltes, J., Lagoudakis, M., Naruse, T., and Ghidary, S. S., editors, *RoboCup 2009: Robot Soccer World Cup XIII*, volume 5949 of *Lecture Notes in Computer Science (LNCS)*, pages 390–401. Springer Berlin Heidelberg. Cited on pages 5 and I.

Wobbrock, J. O., Wilson, A. D., and Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th ACM* *symposium on User Interface Software and Technology*, pages 159–168. Cited on pages 83, 88, and 89.

Wortmann, A. (2010). Self-maintaining and self-monitoring autonomous systems. Diplomarbeit, RWTH Aachen University, Aachen, Germany. Cited on page 6.

Wrede, B., Kleinehagenbrock, M., and Fritsch, J. (2006). Towards an integrated robotic system for interactive learning in a social context. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1817–1823. Cited on page 15.

Wunderlich, J. and Dylla, F. (2002). Technical description of the AllemaniACs soccer robots. Technical report, LTI / KBSG, RWTH Aachen University. in german. Cited on page 17.

Xie, J., Chen, X., Ji, J., and Sui, Z. (2012). Multi-mode natural language processing for extracting open knowledge. In *Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, volume 02, pages 154–161, Washington, DC, USA. IEEE Computer Society. Cited on page 15.

Yanco, H. A., Drury, J. L., and Scholtz, J. (2004). Beyond Usability Evaluation: Analysis of Human-Robot Interaction at a Major Robotics Competition. *Human-Computer Interaction*, 19:117–149. Cited on page 13.

Yen, J. and Lee, J. (1993). Fuzzy logic as a basis for specifying imprecise requirements. In *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, volume 2, pages 745–749. Cited on page 41.

Yin, P., Criminisi, A., Winn, J., and Essa, I. (2007). Tree-based classifiers for bilayer video segmentation. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*. Cited on page 75.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353. Cited on page 41.

Zadeh, L. A. (1975). The Concept of a Linguistic Variable and its Application to Approximate Reasoning - I. *Information Sciences*, 8(3):199–249. Cited on pages 42 and 101.

Zadeh, L. A. (1989). Knowledge representation in fuzzy logic. *IEEE Transaction on Knowledge and Data Engineering*, 1(1):89–100. Cited on pages 42, 101, and 112.

van der Zant, T. and Iocchi, L. (2011). Robocup@home: Adaptive benchmarking of robot bodies and minds. In Mutlu, B., Bartneck, C., Ham, J., Evers, V., and Kanda, T., editors, *Social Robotics*, volume 7072 of *Lecture Notes in Computer Science*, pages 214–225. Springer Berlin Heidelberg. Cited on pages 11 and 14.

van der Zant, T. and Wisspeintner, T. (2005). RoboCup X: A Proposal for a New League Where RoboCup Goes Real World. In Bredenfeld, A., Jacoff, A., Noda, I., and Takahashi, Y., editors, *RoboCup*, volume 4020 of *Lecture Notes in Computer Science*, pages 166–172. Springer. Cited on pages 9, 13, and 64.

van der Zant, T. and Wisspeintner, T. (2007). *Robotic Soccer*, chapter RoboCup@Home: Creating and Benchmarking Tomorrows Service Robot Applications, pages 521–528. I-Tech Education and Publishing. Cited on pages 9 and 13.

Zhu, X., Yang, J., and Waibel, A. (2000). Segmenting hands of arbitrary color. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, page 446. Cited on page 82.

Ziegler, L., Wittrowski, J., Meyer zu Borgsen, S., and Wachsmuth, S. (2014). ToBI - Team of Bielefeld: The Human-Robot Interaction System for RoboCup@Home 2014. Technical report. Cited on page 15.

Statement of Originality

Many of the ideas and approaches presented in this thesis have been influenced by discussions and by collaboration with members of the Knowledge-based Systems Group led by Professor Gerhard Lakemeyer, the ALLEMANIACS ROBOCUP team, and the ROBOCUP@HOME community. Even if it is hard to precisely define individual contributions I detail my contributions in previous publications in the following.

(Wisspeintner et al., 2010, 2009) I made an equal contribution to the methodology of benchmarking domestic service robots and to analysing the progress of the @HOME league in its first years.

(Schiffer et al., 2006a) The semantic mapping was my exclusive work. Further I was leading the development reported on in this paper.

(Dylla et al., 2002a) I was one of three main developers of the approach.

(Jacobs et al., 2009) I equally added to the deployment, maintenance, formalization and evaluation of the collision avoidance method.

(Niemueller et al., 2013) I provided the idea and the conceptual design of the hierarchical object recognition approach and I was one of the main contributors in the evaluation and the exposition.

(Doostdar et al., 2008) I supported the development and I was mainly responsible for the formalization and the presentation. I was equally involved in the evaluation of the system.

(Belle et al., 2008) I equally contributed to the development and evaluation.

(Schiffer et al., 2011a) I provided the idea for the modular gesture recognition and I mainly contributed to the conceptual design and the presentation.

(Schiffer et al., 2012a) I lead the development and I was the principal investigator managing most organizational and scientific aspects. This also holds for (Ferrein et al., 2006; Schiffer et al., 2007; Schiffer and Lakemeyer, 2008; Schiffer et al., 2009; Schiffer and Lakemeyer, 2011).

(Schiffer et al., 2012b) I was leading the development and the scientific contributions summarized in this paper.

(Ferrein et al., 2008, 2009) I equally contributed to the formal account and integration.

(Schiffer et al., 2010a, 2011b) I was leading the application of previous concepts in the domestic service robotics domain.

(Schiffer et al., 2012c) I contributed the hybrid reasoning approach and I was equally involved in formalizing and fully integrating the approach for qualitative positional information.

(Schiffer et al., 2010c,b) I accounted for the idea of a self-maintenance approach based on on-line program transformation and I was equally involved in its realization.

(Schiffer et al., 2012d, 2013a) I contributed the approach to cast the interpretation of natural language as a decision theoretic planning problem, I was an equal member in the development and lead the formalization.