Dissertation

Hierarchical Memory Organization of Multimodal Robot Skills for Plan-based Robot Control

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
an der Fakultät für Mathematik, Informatik und Naturwissenschaft
der Universität Hamburg

eingereicht beim Department Informatik von

Martin Weser

aus Hamburg

Mai 2009

Gutachter:	Prof. Dr. Jianwei Zhang
	Prof. Dr. Bernd Neumann
	Prof. Dr. Bo Zhang
Vorsitzender:	Prof. Dr. Christopher Habel
Tag der Disputatuion:	11.11.2009

Abstract

Today, autonomous service robots are still far from being part of our everyday life. This is hard to understand if one considers the remarkable success of robots in automation industries. In controlled environments the robots' speed, accuracy and reliability by far exceed human capabilities. So the question arises: Why can robots not perform simple tasks in natural human environments? One reason for this inability is the lack of versatility of robot behaviors. Many engineers focus on increasing the performance of specific robot tasks and build robot hardware for certain applications to perceive and manipulate the environment more and more skillfully. In contrast, only few researchers investigate the versatility of robot hardware and software for different tasks.

Another reason for the limited success of service robots is their lack of cognitive competence and flexibility. The ability to reason about tasks and problems is a key property for the application of robots in environments that cannot be preconceived completely by the developers. However, artificial intelligence (AI) techniques, that have remarkable success in several areas, are rarely applied to embodied robots, which may be down to the different representational formats of robotics and AI. Embodied robots inherently deal with continuous data while AI traditionally employs symbolic world representations.

The core work of this thesis concerns the integration of different results from robotics and AI research into one coherent system that marks the state of the art of autonomous intelligent service robots.

Integration takes place along two dimensions: First, several robot sensor and actuator modalities are integrated at robot skill level. While most sensors are developed with particular regard to certain applications and thus show different properties in the workspace, accuracy and measured physical modality, they can complement one another and provide a useful coherent picture of the robot's environment if combined properly. As a result of this thesis, it will be shown that robot perception can particularly gain from multi-sensor integration if sensors are actively focused on objects of interest.

Second, representation and planning techniques from the area of AI will be integrated with the developed multi-modal robot skills to establish a complete robot system from low-level perception and action to abstract reasoning and planning. Specifically, basic robot skills will be organized hierarchically into more abstract robot tasks which can be used on the planning level to decompose complex instructions into executable parts.

Basic robot skills constitute the interface between planning and execution layer. The abstraction level of these skills determines the complexity of the two layers. Although every layered robot architecture has to deal with the question of which task should be achieved at the skill level and which task requires deliberative planning, it has not been addressed explicitly in the literature. In this thesis, properties of basic robot skills are discussed in detail, which leads to a number of arguments that constitute a guideline for the design of such basic skills in the context of layered robot architectures.

The integration of multiple sensor and actuator modalities with state of the art AI planning techniques provides the necessary means for reliable robot behavior in complex situations. It will be shown that the proposed architecture leads to extended autonomy and makes abstract robot tasks possible.

The TAMS service robot is used to demonstrate the system integration and to validate the results of this thesis. It consists of a mobile platform, a robot arm equipped with a three-finger hand and a rich sensor system. This setup enables the robot to perceive its environment and to perform everyday tasks autonomously. Endowing the robot with planning capabilities and integrating results from previous investigations is part of this work.

Kurzfassung

Betrachtet man den großen Erfolg von Robotern in industriellen Umgebungen, so scheint es schwer verständlich, dass mobile Service Roboter noch nicht zu unserem alltäglichen Umfeld gehören. Tatsächlich sind Roboter noch weit davon entfernt, in natürlichen Umgebungen selbstständig operieren zu können. In kontrollierten Umgebungen hingegen übersteigt die Geschwindigkeit, Genauigkeit und Zuverlässigkeit von Robotern menschliche Möglichkeiten bei weitem. Es stellt sich die Frage: Weshalb können Roboter nicht einfache Aufgaben in natürlichen Umgebungen ausführen? Eine Grund hierfür ist mangelnde Flexibilität heutiger Roboter. Während viele Wissenschaftler an einer verbesserten Umsetzung spezieller Roboteraufgaben arbeiten und Hard- und Software im Hinblick auf bestimmte Anwendungen entwickelt werden, ist die Vielseitigkeit künstlicher Systeme meist von geringem Interesse. Ein weiterer Grund für den ausbleibenden Erfolg von Service-Robotern ist der Mangel an kognitiven Fähigkeiten. Die Fähigkeit, komplexe Aufgaben und Probleme zielorientiert zu bearbeiten, ist eine wesentliche Voraussetzung für die Verwendung von Robotern in Umgebungen, die durch den Entwickler nicht vollständig vorhergesehen werden können. Obwohl Techniken der künstlichen Intelligenz (KI) auf verschiedensten Gebieten bemerkenswerte Erfolge erzielen, werden sie bis heute selten in realen Robotern eingesetzt.

Zentraler Bestandteil der vorliegenden Dissertation ist die Integration unterschiedlicher Ergebnisse aus der Robotik und der KI in ein kohärentes System, welches vielseitig einsetzbar ist und den aktuellen Stand der Technik in der autonomen Servicerobotik widerspiegelt.

Unterschiedliche Sensoren und Aktuatoren und deren Ansteuerung werden zu elementaren Roboterfähigkeiten zusammengefasst, so dass eine sinnvolle Ergänzung der unterschiedlichen Eigenschaften in Genauigkeit, Arbeitsraum oder gemessener physikalischer Eigenschaft erziehlt wird. Um ein vollständiges Robotersystem zu realisieren, werden weiterhin Repräsentations- und Planungsmethoden aus dem Gebiet der KI in das bestehende System integriert. Hierfür werden die elementaren Roboterfähigkeiten hierarchisch zu abstrakteren Aufgaben organisiert, welche auf Planungsebene genutzt werden, um komplexe Anweisungen in ausführbare Teilaufgaben zu zerlegen.

Wie in dieser Dissertation gezeigt wird, können Ergebnisse der Perzeption insbesondere durch aktives Fokussieren der Sensoren auf relevante Objekte verbessert werden. Sowohl auf Planungs- als auch auf Ausführungsebene werden nicht nur Sensoren verwendet, um Aktionen zu leiten und zu überwachen, sondern Aktionen werden eingesetzt, um Ergebnisse der Perzeption zu verbessern. Diese Umkehr des Abhängigkeitsverhältnisses von Perzeption und Aktion wurde in diesem Umfang bisher nicht gezeigt.

Die Integration verschiedener Sensor- und Aktuatormodalitäten mit Planungsmethoden der KI liefert grundlegende Bausteine für zuverlässiges Verhalten von Robotern in komplexen Situationen.

Der TAMS Serviceroboter wird als experimentelle Plattform genutzt, um die Realisierbarkeit des vorgeschlagenen Systems zu demonstrieren und die Ergebnisse der vorliegenden Arbeit zu validieren. Die mobile Plattform, welche mit einem Roboterarm, einer Roboterhand und einem umfassenden Sensorensystem ausgestattet ist, bietet die nötigen Vorraussetzungen für autonome Bearbeitung alltäglicher Aufgaben. In dieser Arbeit wird gezeigt, dass die für diese Arbeit entwickelte Architektur die Autonomie des Roboters erhöht und die Bearbeitung von abstrakten Aufgaben ermöglicht. Den Roboter um die Fähigkeit des Planens zu erweitern und dabei die Ergebnisse früherer Arbeiten zu ergänzen ist Bestandteil dieser Arbeit.

Acknowledgment

Wie lieb und luftig perlt die Blase
Der Witwe Klicko in dem Glase!
Wilhelm Busch, 1872

I wish to thank…

- … my supervisors Prof. Dr. Jianwei Zhang and Prof. Dr. Bernd Neumann for giving me the most valuable feedback even on short notice and sometimes in the most adverse circumstances. I also want to thank Jianwei for making my work possible by providing an excellent research environment.
- … CINACS Tsinghua Supervisor Prof. Bo Zhang for reviewing this dissertation.
- … my colleagues from the TAMS group for providing professional support and personal advice in all the time of research for and writing of this thesis. Specifically I want to thank Lu, Norman, Andreas, Bernd, Martin, Hannes, Dennis, Houxiang, Mohammed …
- … my colleagues from the CINACS graduate research group at the Informatics Dept.: Sascha, Tian, Pat, Cengiz, Christian and Dominik.
- … my colleagues from the Department of Systems Neuroscience, University Medical Center Hamburg-Eppendorf and the Institute for Biological Psychology and Neuropsychology, University of Hamburg, for interesting and inspiring discussions. In particular I thank Mario, Patrick, Andreas, Inga, Jens, Sabrina.
- … my parents and my friends for supporting me even if I disregarded them during the last few month. Thank you Eva, Helmut, Miri, Bo, Tanja, BenG, Sarah, Manu, Christopher, Sabrina, Nele, Tim, Wolle, …

This work is funded by the DFG German Research Foundation (grant 1247) – International Research Training Group CINACS (Cross-modal Interactions in Natural and Artificial Cognitive Systems).

The experimental environment, including the mobile service robot TASER, is property of the Technical Aspects of Multimodal Systems Group, Department of Informatics at the University of Hamburg.

Contents

List of Figures

1 Introduction

Failure of existing rules
is the prelude to a search for new ones.
Thomas S. Kuhn, 1996

A common conception of *robots* regards them as super-intelligent beings that assist humans in almost every situation or even try to take over the world. This idea is influenced by science-fiction literature or films and is far beyond the actual abilities of today's robots.

The term robot has quite a broad meaning. It can refer to a software that crawls web pages, a manipulator arm, an artificial insect, an autonomous airplane and many other technical systems that somehow act more or less purposefully. If we look at the term *service robot*, we slightly approach what ordinary people consider as robots. The purpose of service robots is actually to assist humans in recurring and tedious everyday tasks. However, people with knowledge of the field know that we are far from building beneficial systems for largely unstructured human environments.

The project presented in this thesis contributes to the ambitious goal to develop service robots that autonomously achieve complex tasks in largely unrestricted environments. This requires equipping the robot with strong reasoning capabilities on the one hand and the ability to reliably perceive and manipulate its environment on the other. The robot TASER (TAMS Service Robot) will serve as a testbed for the demonstration of implemented robot abilities.

The following three exemplary complex robot tasks will be utilized throughout the thesis to illustrate the process of generating goal-directed embodied behavior as well as to demonstrate the advances of the proposed system:

Pick-up and delivery services A service robot should be able to deliver objects from and to different places. This includes perception of the object of interest, grasping capabilities, and the ability to place the object on a surface or to hand it over to a human principal. Issues on advanced human-robot interaction will be excluded since this is an area of active research on its own.

Navigation in challenging environments The robot must be able to navigate safely in a dynamic environment, possibly with the presence of humans. Navigation includes path planning, moving through narrow passages and perceiving and opening doors.

Inventory taking The robot should be able to detect objects and keep track of their number, location and configuration. This includes objects that are of special interest to the robot, i.e. objects that can be manipulated or that may influence the execution of certain robot actions.

These three exemplary robot tasks require competence in perception, manipulation, mobility and planning and provide a comprehensive testbed for service robots.

1.1. Grand challenges on the way towards autonomous service robots

A robot that can autonomously and reliably achieve the above tasks would clearly mark the state of the art in robot research. Considering the precision, speed and reliability of robots in manufacturing industries, it seems to be absurd that such simple tasks in natural environments are still challenging. Several reasons prevent robots from acting intelligently in real-world scenarios. The four grand challenges in robot research which are described in the following comprise major difficulties in building autonomous service robots:

1. A major problem in mobile service robotics is the robot's uncertainty of its own location relative to external objects and of the nature of the environment in general. What is the use of high-precision manipulation (industrial robots easily achieve an accuracy of less than 1mm) if the location of the target object is unknown? Reliable perception is a key property of robots that can be avoided in industrial environments due to precise modeling of a constant and known environment.
2. Another great challenge in service robotics is the system integration. To build a robot with mobile manipulation capabilities, several disciplines have to be combined: control-theory, computer science, mathematics, electronics and mechatronics, to name but a few. Informatics, as a particular case, encompasses kinematics, human-robot interaction, localization, manipulation, perception, machine learning, artificial intelligence etc.
3. Task-level learning is still in its infancy. Machine learning methods rarely go beyond parameter adaption. In relatively small domains, these methods are successfully applied, examples are pattern recognition and nonlinear motor control. However, autonomous mental development, which is learning of new behaviors from scratch, is still far from possible.
4. Probably the most important challenge may be characterized as embodied artificial intelligence. How to endow a robot with intelligent reasoning capabilities? Complex tasks given by human instructors have to be decomposed into subtasks

that can be achieved by the robot, and goal-directed decisions on the future course of actions have to be made. Since it is impossible to foresee all situations at programming time that a robot will potentially encounter within its work, a robot needs the ability to flexibly adapt its own behavior to new tasks and environments.

A rather general problem in robotic research is the difficulty to exchange tools and algorithms among research groups. Robot software fameworks, e.g. Player/Stage (Gerkey et al., 2003), propose to standardize robot development environments to allow exchange and comparison of components. However, due to very diverse technical prerequisites, many issues from disciplines listed in challenge two have to be addressed for every new robot platform. Similarly, results from service robot research are hardly comparable if no consistent experimental environment is used. And last but not least, elaborate robot hardware is still expensive. Comparably few institutions have the opportunity to investigate multi-modal robot platforms with rich sensor and actuator capabilities. Luckily, we are one of them.

1.2. Contribution

It goes without saying that one thesis cannot solve all challenges of service robot research. However, when working with a complete robot platform, one will have to deal with these issues at least peripherally. For example, system integration (Challenge 2) must be addressed every time a new component is introduced and every time an existing component will be used in a way that was not foreseen by the initial developers.

The central objective of the project, which led to the findings described in this thesis, is to enable a robot to achieve a variety of complex and abstract tasks autonomously. To achieve this objective, several established methods and elaborated libraries for robot control will be used. The main contributions of this thesis concern Challenge 2 and Challenge 4. The contributions are described as follows.

1. Every plan-based robot control architecture has to comprise an interface between deliberative planning in the symbolic domain and embodied execution of robot behaviors in the continuous domain. Assuming that the deliberative component triggers programs that actually generate behavior, there is necessarily a point at which these programs appear as atomic from the deliberative point of view. This leads to the research question of an appropriate abstraction level for atomic robot skills to minimize the system's overall complexity. Although every plan-based robot control architecture has to deal with this issue, it has not been addressed individually in the literature. The discussion on an appropriate abstraction level for atomic robot skills in Chapter 4 leads to several criteria that have to be considered while designing plan-based robot control architectures. These guidelines are of general applicability for service robot scenarios beyond the implementation testbed TASER.

2. Current robot systems provide several sensors that show complementary properties. Usually, they are used for dedicated purposes, e.g. laser range scanners and sonars are used for localization, cameras for object recognition, and force sensors to support grasping operations. Due to non-overlapping workspaces, the integration of robot sensors for a common purpose is rarely seen. In Chapter 5 it will be demonstrated how perception can be improved by purposefully focusing robot sensors on common objects of interest. This reverses the natural relation of sensors and actuators: sensors are not exclusively used to guide actuators, but actuators can also assist sensors in deliberatively initiated perception with specific attentional focus.

3. Symbolic planning has been investigated for years in the traditional AI community. Remarkable results have been achieved, especially with configurable hierarchical task net (HTN) planners. In spite of this, many research labs still develop dedicated robot planners with specific focus on temporal constraints, continuous representation of sensor data and uncertainty management. Chapter 6 investigates the question of whether traditional AI planners can be used for plan-based robot control. It will be shown that the organization of robot skills into a hierarchical memory representation allows for the planning of robot control sequences with an HTN planner.

A practical contribution of this work is the enhancement of the capabilities of the service robot TASER. This robot platform is used to demonstrate the multi-modal integration of sensors and actuators in a variety of robot skills. For the implemented skill library, symbolic descriptions will be defined which will ground a hierarchy of abstract robot tasks. Based on this hierarchical memory, a planner will be used to endow the robot with the capability to achieve complex tasks by executing sequences of skills in a goal-directed manner. Experiments with the real robot will prove the feasibility of this approach and demonstrate the extended autonomy of the system while achieving complex tasks.

1.3. About this work

This work benefited largely from an excellent interdisciplinary research environment. Findings from natural cognitive information processing, which where taught at the Department of Systems Neuroscience, University Medical Center Hamburg-Eppendorf and the Institute for Biological Psychology and Neuropsychology, University of Hamburg, have influenced decisions on the implementations of multi-sensor interaction. The idea of understanding perception as a deliberative act, for example, arose from the insight that the lateral geniculate nucleus (LGN) in the human brain receives strong feedback connections from the primary visual cortex rather than just processing information from the retina in a bottom-up manner. Journal clubs, colloquia and uncountable discussions with colleagues from the area of Neuroscience and Psychology put my work into an interdisciplinary perspective.

The international cooperation with the Tsinghua University in Beijing, specifically with the State Key Laboratory of Intelligence Technology and Systems, has also influenced the progress of this thesis positively. Parts of my investigations and implementations of algorithms have been done during a three-month research exchange program. I also want to mention three CINACS summer schools that provided an excellent opportunity for scientific and cultural exchange.

1.3.1. A note for the reader

The list of references used in this work is ordered alphabetically and without any preferences. HTTP links are referenced only at places where no written source of information is available. Since HTTP links tend to be comparatively short-lived, they are not considered as a replacement for official references. Instead of being listed in the bibliography, they are marked as footnotes within the text as additional information.

Java and Jini are trademarks of Sun Microsystems, Inc., USA; Roblet and genRob are trademarks of Hagen Stanek, Germany. Although these terms are not marked throughout this work, the author respects the rights of the holders.

1.4. Thesis structure

The following chapter introduces the experimental robot platform which is used within this work to demonstrate the feasibility of the proposed approach. The robot's sensors and actuators will be described along with a brief introduction of the software architecture which is used to control the robot.

Chapter 3 provides the fundamentals of autonomous robot action generation. The purpose of this chapter is twofold: first it reviews the current state of the art in all related areas. It points out several issues that are not investigated exhaustively and provide further research opportunities. This leads to the second purpose of putting the present work into context. Section 3.7 contrasts a summary of the state of the art with the proposed approach.

Chapter 4 introduces the notion of atomic robot skills as it is used in this thesis. Robot actions, primitive operators and robot control programs will be defined and discussed, which establishes the basis for the subsequent chapters. A detailed discussion on the requirements and properties of atomic robot skills leads to several considerations that have to be made when designing and implementing a layered robot architecture.

The results of Chapter 4 guide the definition of a library of robot skills for mobile manipulators. The implementation of this skill library in the form of robot control programs for the service robot TASER will be described in Chapter 5 along with the symbolic representations which will be used later on for deliberative planning of goal-directed robot action sequences. The specific emphasis of this chapter is on demonstrating the benefit of multi-modal integration on the skill level. The use of physical movements to support robot perception is inherent to most of the implemented robot skills that focus on perception.

Chapter 6 investigates the integration of an AI-based planning system into the robot control architecture. Specifically, JSHOP2 will be used to demonstrate HTN planning in the service robot domain. By the end of this chapter, the service robot TASER will be able to achieve complex service tasks autonomously by executing goal-directed action sequences.

Experiments that demonstrate and validate the proposed approach will be described in Chapter 7. Proactive perception of complex objects using several sensor modalities and the support of physical robot actions will be used to present the functionality of the implemented overall system including symbolic planning and reactive execution.

The main contributions of this thesis are summarized in Chapter 8. The thesis concludes with a discussion of the limitations of the proposed approach and future research directions.

2 Experimental Platform

Intelligence is the faculty of making artificial objects,
especially tools to make tools.
Henri Bergson, 1859-1941

This chapter describes the multi-modal service robot TASER (TAms SErvice Robot). The robot is used as the experimental platform for the implementation, demonstration and evaluation of the developments and findings from this project. Although some important contributions of this thesis have general applicability, large parts of the presented work are directly dependent on the robot. A large part of the time which has been spent to make this project possible was used for maintenance of the robot hardware and further development of the controller software. TASER's sensors and actuators are indispensable for all developed robot actions, and specifically for the final demonstration scenarios which are presented in Chapter 7.

The robot platform is based on the MP-L655 platform of Neobotix, which belongs to GPS GmbH. It consists of a rectangular base that provides space for eight lead acid batteries, power electronics, the host PC and hardware controllers for several components. The batteries supply a main power of 48 Volts with the total power of 3.84 kWh. This guarantees an overall independent working time of approximately eight hours. Mobility of the robot platform is realized with three passive wheels and two drive wheels. To allow the investigation of robot manipulation with a human-like workspace, a tower is installed on top of the base. This tower is designed to host two robot arms with similar measurements to a human upper-body. In the current version, only one arm is installed. Figure 2.1 gives an idea of the shape of the platform as well as the scales of the main components.

The robot comprises several independent hardware devices which are described in the following sections. They are organized into robot actuators (Section 2.1) and robot sensors (Section 2.2). Section 2.3 presents the software environment which combines the control of sensors and actuators while encapsulating unwanted details of network communication. The software environment actually states the interface which is used to implement high-level robot action and perception routines in Chapter 5.

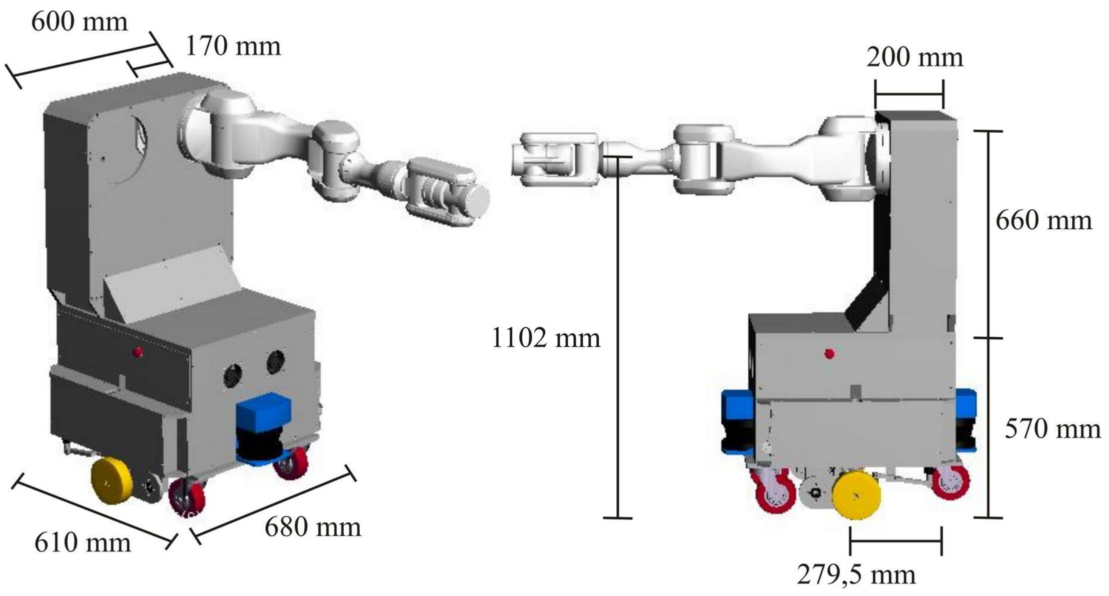


Figure 2.1.: The robot platform with one arm mounted.

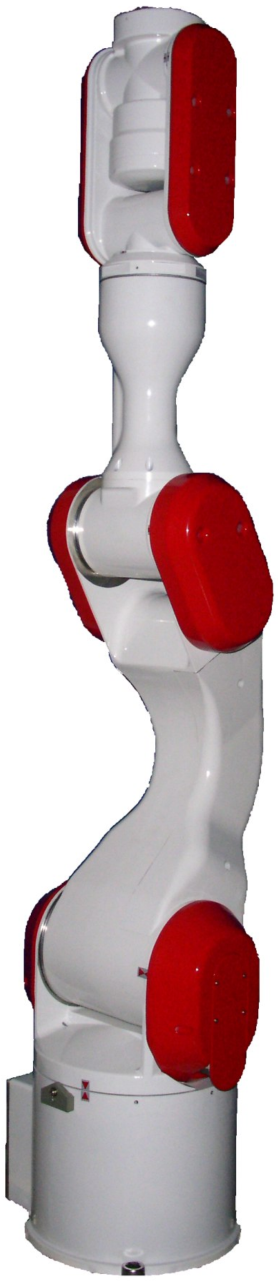
2.1. Robot actuators

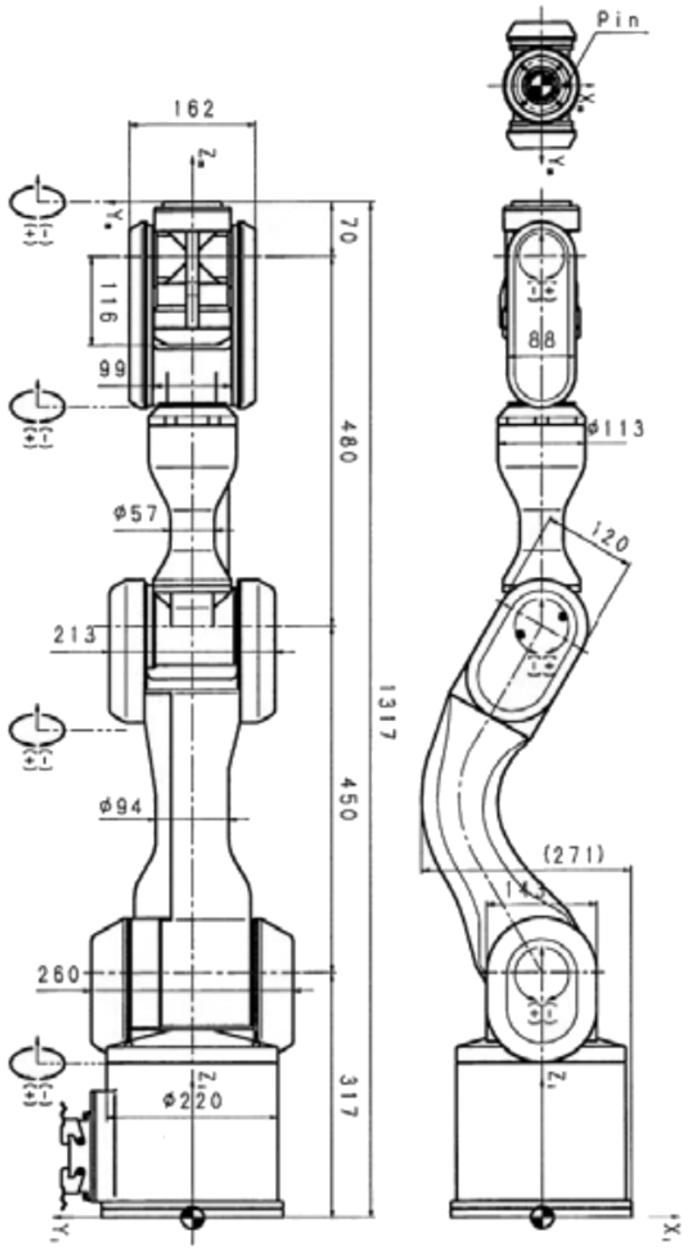
Actuators are the parts of the robot hardware that are used to change the external world physically. Without actuators, a robot would degenerate to a passive observer. The following sections describe actuators of the service robot TASER.

2.1.1. Robot arm

The service robot TASER is designed for two PA10-6C robot arms from Mitsubishi Heavy Industries (MHI) as shown in Figure 2.2. In the current version of TASER only one arm is mounted. The arm has a total length of 1317 mm and six degrees of freedom. In this it is comparable with a human arm. This is useful for experiments in environments that are designed for humans. The measures of the segments of the arm are shown in Figure 2.2(b). The payload of the arm is specified with 10 kg, however, it is decreased due to the low working voltage of 48 Volts instead of 100 Volts as recommended by the manufacturer.

The PA10 series is controlled using ARCNET. Since the PA10-6C sends and receives data via special fiber-optics, an MHI-optical board is used together with an ARCNET card from Contemporary Controls. As control software, the Robot Control C Library (RCCL), written by (Lloyd and Hayward, 1992) is used. It was modified as described in (Scherer, 2004) to enable the control of PA10 series manipulators.


(a)



(b)

Figure 2.2.: The PA10-6C robot arm. Pictured in the colors of the University of Hamburg and as a technical diagram. Source: Mitsubishi Heavy Industries

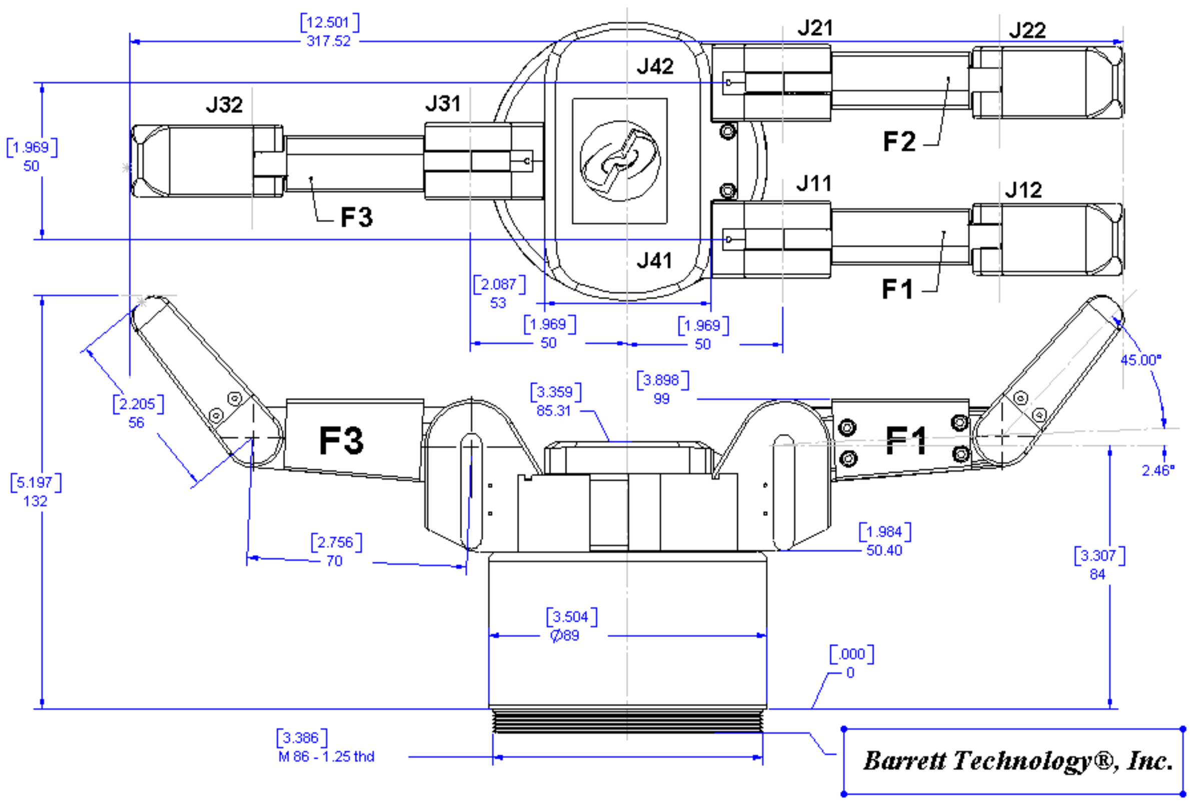


Figure 2.3.: Measures of the BarrettHand: the unit is [inch]: 1 inch = 25.4 mm. Source: Barrett Technology, Inc.

2.1.2. Three finger hand

The robot is equipped with a three finger hand which is mounted as end effector of the PA10-6C robot arm. A the time when the robot platform was set up (by the end of 2003) the BarrettHand was the only commercially available multi-finger hand.

The BarrettHand possess three fingers with a maximum span width of 317.5 mm as shown in Figure 2.3. The two joints of each finger are driven with one motor via a special gear box which is called TorqueSwitch^{TM}. This mechanism allows a constrained control of two joints with only one motor and gear cable (Townsend, 2000). If the first link of a finger (joint J11. . . J31 in Figure 2.3) encounters resistance, the motor force will be transmitted to the second joint (J12. . . J32). Otherwise, the second joints are not actively moved. The payload of each finger is 2 kg.

The spread angle of finger one and two are controlled simultaneously and symmetrically. This means that the spread angles of these fingers are always equal.

2.1.3. Mobile platform

The mobile platform consists of two differential drive wheels and three free wheels for stabilization of the heavy-weight system. The setup of the wheels is shown in Figure 2.1. The advantage of this setup is that turning in place is possible. This allows for the convenient execution of complicated locomotion maneuvers. The disadvantage is that the drive wheels may loose grip in rough terrains. However, this drawback can be neglected since the intended area of application is an office environment which usually provides quite plane floors.

The motors for the drive wheels are accessed via a Controller Area Network (CAN). The mobile platform is controlled by a C/C++ application developed in (Scherer, 2004) and enhanced for TASER. It provides a TCP/IP interface to trigger motion commands.

2.1.4. Pan-tilt unit

The robot platform possess a pan-tilt unit (PTU) which serves as a flexible mount for the stereo vision system which is described in Section 2.2.2. It is of the type PTU-46-17.5 produced by Directed Perception[1]. The ranges of operation are 47° down and 31° up in the tilt direction and symmetrical ±159° in the pan direction. Maximum payload is 1.81 kg and maximum speed is 300° per second (Acuity Research, 2000).

The PTU is connected to the host computer via an RS-232 port and consumes 7.5 Watt in the low-power mode with unregulated 11-37 Volts DC.

2.2. Robot sensors

Different sensor modalities complement one another and enable the robot to perceive its surroundings to a degree that allows autonomous robot actions in largely unstructured environments.

The following sections describe sensors that measure signals of the external world over long distances, i.e. laser range sensors and cameras. Then, the force sensors on the robot hand are described.

2.2.1. Laser range scanner

The robot is equipped with two SICK LMS200 laser range scanners. Each range scanner is set to scan an area of 180° with an angular resolution of 0.5°. The scanning distance is restricted to eight meters. A special purpose RS422-to-Ethernet adapter is used to connect the sensor to the host PC. The adapter is based on a Rabbit Semiconductor

[1] `http://www.dperception.com`, accessed on 25 April 2009

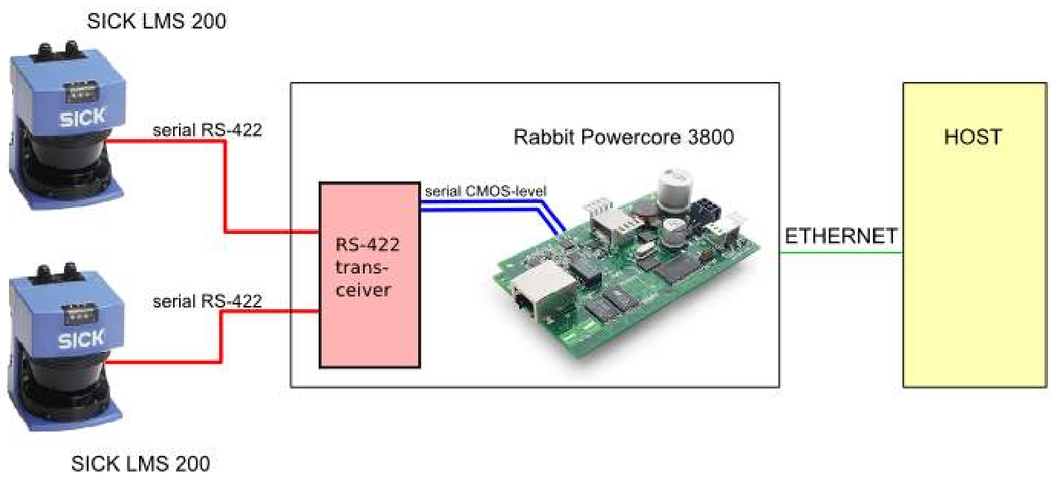


Figure 2.4.: SICK LMS200 laser range scanner and structure of the connection to the host PC. Picture: (Bistry et al., 2007b).

PowerCore 3800 module with a Rabbit 3000 8 bit microprocessor. Additional parts are RS-422 transceivers and status-LEDs. The laser range scanner and the structure of the adapter is shown in Figure 2.4. See (Bistry et al., 2007a) for a detailed description of the adapter. Due to the large interest in the system, the source code for the Rabbit 3800 is published under GPLv3. It can be downloaded at the TAMS website[2]. The laser range scanners are used within this project for robot localization and perception of certain objects.

2.2.2. Cameras

Cameras are similar to laser range scanners in that they provide information on the environment over large distances. Thus they are specifically suitable to detect objects or properties of objects which are not directly under manipulation.

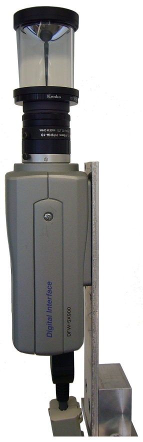
An omni-directional vision system and a stereo camera system have been installed on the robot TASER. Both camera systems are connected to the robot via IEEE DC1394 firewire connections. They are described in the following.

Omni-directional vision system

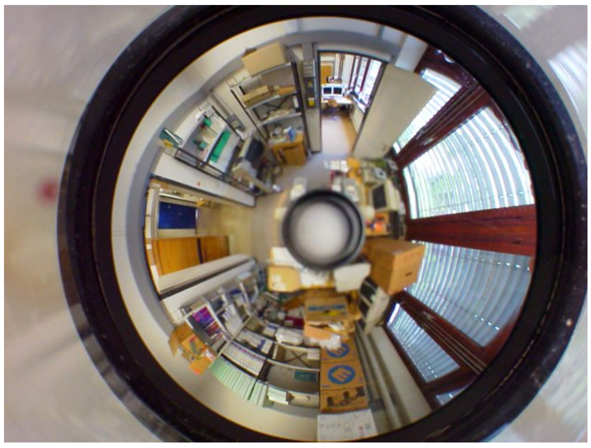
The omni-directional vision system is based on hyperboloidal mirrors of the type Panorama Eye[3]. To obtain an omni-directional image, a camera is installed vertically under a mirror. The obtained Bird's eye view is transformed then into a 360 degree panoramic image (Chahl and Srinivasan, 1997). The setup is shown in Figure 2.5(a).

[2] http://tams.informatik.uni-hamburg.de/research/robotics/service_robot/hardware/, accessed on 25 April 2009

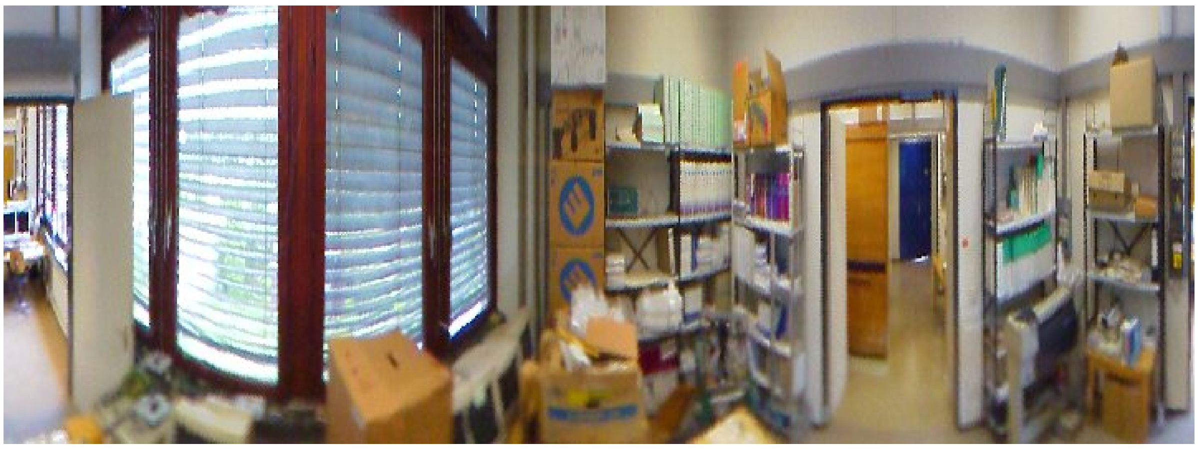
[3] http://www.accowle.com/english/index.html, accessed on 25 April 2009



(a) Setup of omni-directional vision system.



(b) Raw omni-directional image.



(c) 360 degree panoramic view.

Figure 2.5.: The omni-directional vision system: the setup, raw omni-directional image and resulting 360 degree panoramic view.

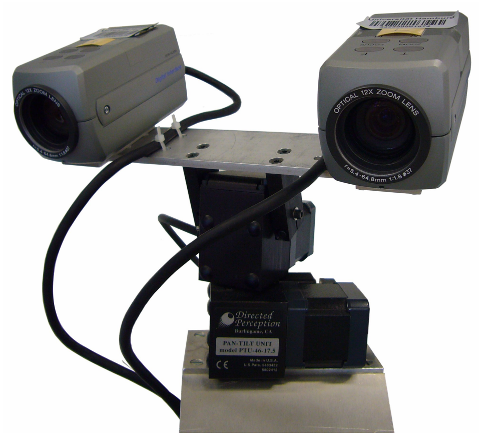


Figure 2.6.: stereo vision system mounted on a pan-tilt unit.

A Sony DFW-SX 900 camera in combination with a FUJINON 9mm/F1.4-F16 HF9HA-1B lens is used to capture the images. The camera is an IEEE DC1394 camera with SXGA resolution of 1280x960 pixels. With this resolution, the maximum frame rate is 7.5 frames per second in YUV422. A captured raw image and the resulting panoramic view is shown in Figure 2.5.

The omnidirectional vision system is mounted on top of the robot at a height of approximately 1.95 m.

Stereo camera system

The stereo camera system is composed of two Sony DFW-VL 500 cameras with 12x optical zoom. They are mounted on a rack with 110 mm distance between their the optical axes. The setup of the stereo camera system mounted on a pan-tilt unit (see Section 2.1.4) is shown in Figure 2.6. The cameras are IEEE DC1394 cameras with a maximum resolution of 640x480px at a frame rate of 30 frames per second in YUV411.

2.2.3. Force sensors

Each finger of the BarrettHand (see Section 2.1.2) is equipped with a resistance strain gauge (RSG). The RSG is built in the first link of the finger and measures forces that act on the gear cable.

Since the sensors do not provide force values in Newton units, they have to be calibrated in a set of experiments utilizing an external weighting machine. Furthermore, the provided force values are dependent on the fingers' position. This, and the fact that the fingers are controlled by requesting motion speed rather than desired forces, requires a second calibration step in which velocities of the fingers are mapped to forces that are applied on the fingertip. For details of the calibration process see (Baier-Löwenstein, 2008).

2.3. TASER's control architecture

In order to keep the control software maintainable, many complex multi-modal robots hide low-level details of the system in a hardware abstraction layer (HAL). Task-level programs are then implemented based on this HAL. However, some newly implemented complex robot tasks may require access to low-level hardware control as well as to functionality provided by the HAL. This calls for a software architecture that is capable of providing encapsulated high-level functionality as well as low-level access to parts of the robot hardware.

For the service robot TASER the Roblet architecture is used to unify hardware control modules into one coherent software environment. Both abstract interfaces that hide hardware details as well as access to low-level sensor and actuator control are provided by Roblet-Servers. As the Roblet architecture is completely based on Java, native hardware drivers have to be wrapped with the Java native interface (JNI). Roblet-Servers then provide access to this components for workstations distributed in a network. For details of the Roblet architecture see e.g. (Baier et al., 2006).

Robot hardware components that are encapsulated via JNI and Roblet-Servers are:

- two laser range scanners
- three IEEE DC1397 cameras
- TCP-based control of the drive wheels
- the pan-tilt unit
- the PA10-6C robot arm
- the BarrettHand with force sensors
- monitoring services of internal parameters (temperature, voltage)

In addition to hardware control, some Roblet-Servers have been implemented to provide software functionality, such as path planning and image processing. An overview of the control architecture is given in Figure 2.7. In the current version of TASER, there are more than 10 Roblet-Servers up and running; many of these have been developed within this project.

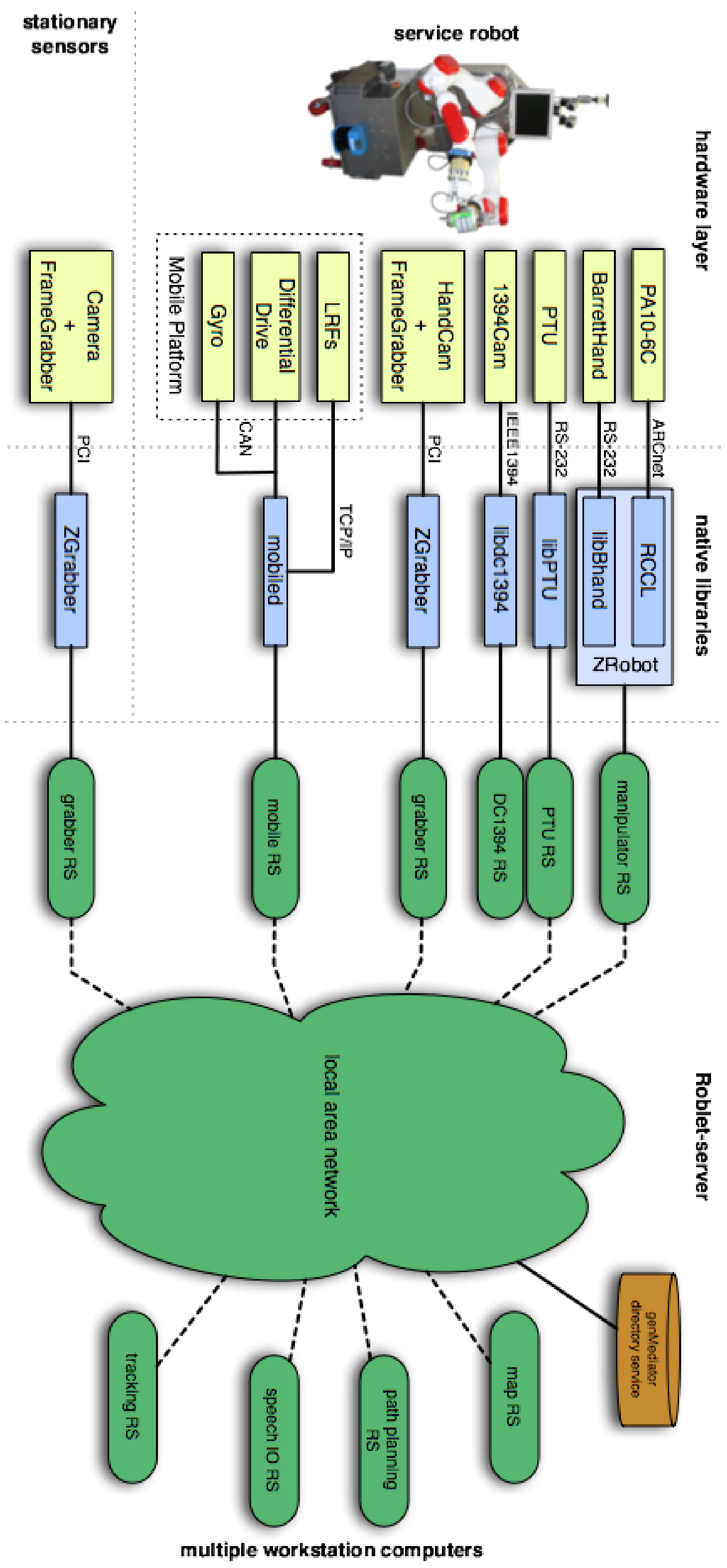


Figure 2.7.: Control architecture of TASER: robot hardware (yellow) is controlled by native libraries (blue) that are wrapped into Roblet-Servers (green).

Fundamentals of Autonomous Robot Action Generation

3

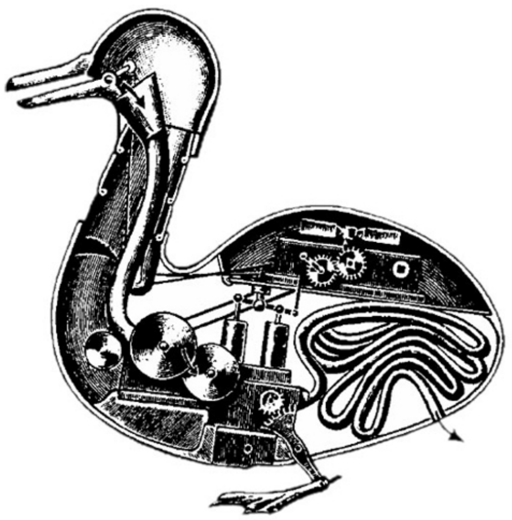
If you can't explain it simply,
you don't understand it well enough.
Albert Einstein, 1879-1955

Every time when significant progress was made in the history of computing, people claimed that artificial computation will soon exceed human capabilities. Several events in history can be seen as a starting point of computation. Regardless of whether it was Charles Babbage, who described his analytical engine in 1835[1], Alan Turing's paper on universal computers (Turing, 1936) or John von Neumann's architecture of a computer that uses the same memory both to store programs and data – people where impressed by the (theoretical) capabilities of the upcoming generation of computing machines. Nowadays, people realize that intelligence is more than just analytical computation. It is unlikely that anybody believes that the next considerable increase of computational capabilities will endow computers with true intelligence.

Similarly, people attested mysterious autonomy to the first robots, e.g. the "Digesting Duck" by Jacques de Vaucanson in 1737 (Figure 3.1(a)), and believed they soon would be seeing artificial companions. The first industrial robots made people think that human labor would soon become dispensable. Robot control has made, superficially speaking, remarkable progress. For example, the pole balancing problem[2] has been solved (Moore, 1990), robots are capable of juggling with three balls (Bühler et al., 1990) and the famous humanoid ASIMO (Figure 3.1(b)) can dance, walk and step on stairs (Chestnutt et al., 2007). However, autonomy in complex unstructured environments and the ability to solve challenging problems is still very limited. Today, we know that some tasks that still cannot be done by machines most probably will not be solved in the near future.

[1]Babbage's analytical machine was the plan of a general-purpose programmable computer based on mechanics driven by a steam engine.

[2]In the pole balancing problem a pole is mounted flexibly on a movable cart. The task is to balance the pole solely by moving the cart. Is is solved using closed sensor-actuator loops with parameters learned through reinforcement learning techniques.



(a) The digesting duck by Jacques de Vaucanson, 1737

(b) The humanoid robot ASIMO by Honda Motor Co. Inc., 2008

Figure 3.1.: State of the art of robot research 1737 and now.

Although traditional AI has achieved impressive results, its application to autonomous robots is still rare. One reason is the diverse representation of actions, objects and events in robot control programs on the one hand and in AI reasoning systems on the other. AI reasoners usually adopt a state space approach that is associated most prominently with the work of (Newell and Simon, 1975). Robot control programs in contrast process sensor signals and generate motor commands that are naturally continuous.

Supposably, the most important step on the way to build truly versatile robots that act in unstructured environments is combining results from these different research directions. This chapter's intention is to put the work described in this thesis into context. The following two sections provide an overview of the current state of the art in robot control, AI-based planning and the approaches to combine both in embodied robot systems. The combination of these lines of research lead inevitably to the problem of how to relate internal data structures with the external world. Approaches to overcome the so-called symbol grounding problem are discussed in Section 3.3. The step from versatile service robots to systems that autonomously learn new behaviors obviously requires computational learning techniques. Although robot learning is only of minor importance for this thesis, for the sake of completeness an overview of learning techniques is provided in Section 3.5. This chapter concludes with an overview of robot systems that exemplify the current state of technological progress. Major findings from the literature-review given in this chapter are summarized in Section 3.7.

3.1. Robot control

Dexterous robot movements that adapt flexibly to the external world are profoundly complicated. In biological systems, actions naturally adapt to the environment to a certain degree due to the compliance of all body parts. In contrast, artificial limbs are usually rigid and have to be controlled precisely into exact positions. Robot compliance has to be introduced artificially later on via force control. An exception are actuators driven by air muscles that are innately compliant, like products from the Shadow Robot Company[3]. This line of research becomes more and more important in automated engineering applications that are specifically designed for safe human-robot interaction (Klug et al., 2008).

The following sections address the question of how robot actuators are made to move. The simplest form is to set motors to certain values with no regard to the external world. In this case, the only sensors that are involved provide proprioceptive information about the current motor state. In most cases, however, the robot's actions should have an impact or comply to the external world. Thus, the action effects and external physical signals e.g. forces, distances or positions, have to be measured and fed back into the control process. To achieve safe robot behavior, usually several control levels are applied.

The basic idea of control loops is explained in the next section. Section 3.1.2 discusses how more complex robot behavior can be achieved without explicit world modeling and reasoning. Robot control applications that require specific control algorithms which utilize elaborated world models are discussed in Section 3.1.3.

3.1.1. Control theory

Control theory is an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems. Simply speaking, its goal is to adjust to or hold a system output close to a reference value that is given by higher-level processes. Therefore, the system state is measured and compared to the reference value which is fed into a controller as an error signal. The controller generates an input value to change the state of the controlled system. The control loop states the basic principle of control theory; it is shown in Figure 3.2.

Control theoretic approaches as described here are usually used for the low level control of actuators. They are often implemented in hardware to ensure real-time capabilities. The reference values that drive the actuator's controller are actually used to affect the system's behavior on larger time scales.

[3] http://www.shadowrobot.com, accessed on 25 April 2009

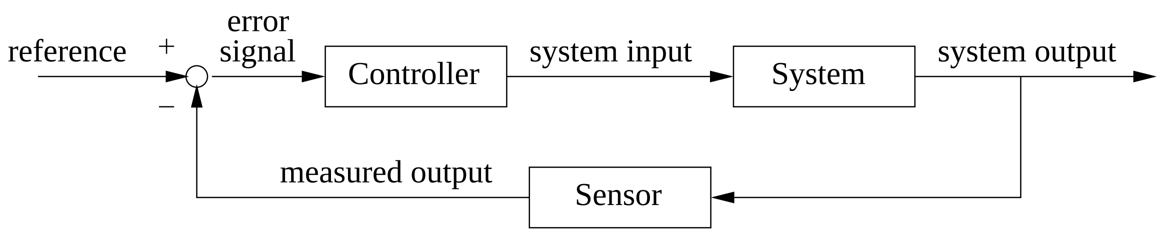


Figure 3.2.: The concept of a feedback loop to control the behavior of a dynamic system. The sensed value is subtracted from the reference value to create an error signal.

The measured system output and the reference value can in general be of arbitrary complexity: the speed of a motor may be controlled by a simple linear controller while a controller that holds a car on the lane uses several sensors, internal models and nonlinear filtering to predict the effect of the introduced system input. Frequently used controllers are the proportional-integral-derivative (PID) controller, the neural network based controller and the fuzzy controller (Zadeh, 1965). A hierarchy of controllers may be used to achieve complex robot behavior as described in the next section. See e.g. (Kilian, 2005) for an excellent overview of modern control techniques.

3.1.2. Behavior-based systems

The underlying idea of this line of research is that the behavior of a robot at a given moment is to be controlled by the situation at that moment. Behavior is seen as a matter of stimuli and response. Elaborated mechanisms for building world models and making inferences from them are to be avoided. Although (Braitenberg, 1984) introduced this idea earlier, it is associated most prominently with the work of (Brooks, 1986; Brooks and Stein, 1994).

In *behavior-based systems* (BBS), a layered set of behavior-producing modules are simultaneously active. Each module generates motor commands based on current sensor readings. They may also perform temporal filtering of sensor data that can be seen as short term memory. The actual motor command is a weighted sum of the outputs of all modules. Modules of higher layers can suppress or potentiate lower modules. The classical architecture as proposed by Brooks is shown in Figure 3.3. Such architectures do not make it impossible to write programs that maintain complicated world models, but they discourage the practice. A slogan of this line of research is: Use the world as its own best model (Arkin, 1998).

Robots that are controlled solely by feed forward neural nets that map sensor readings to motor commands may be seen as behavior-based systems as well. They do not maintain explicit world models, in fact their behavior is determined by the current situation that is perceived and fed into the net.

The similarity of BBS to control theoretic approaches is that each module constitutes

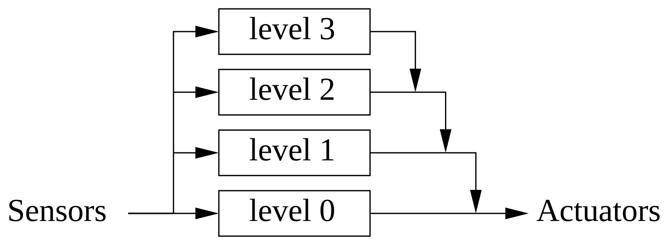


Figure 3.3.: Original subsumption architecture according to (Brooks, 1986). Control is layered with higher level layers subsuming the roles of lower level layers when they whish to take control.

a feedback loop. However, in BBS it is not assumed that reference values are given. Instead, they are coded explicitly in the modules. This kind of robot control is well suited for homogeneous tasks such as approaching a target with a mobile platform or a manipulator. If different tasks are to be achieved these controllers have to be configured, modified or exchanged.

3.1.3. Complex robot control applications

The robot control mechanisms described so far do not focus on certain applications. A generic controller as it is described in Section 3.1.1 may be used to maintain the pressure on a valve or to position a joint at a certain angle. Behavior-based systems may also be used in different contexts, e.g. locomotion guided by laser-range sensors or visually guided manipulation. This flexibility is at the expense of specificity.

If accuracy and correctness of tasks have to be ensured, it is often hard to find provable algorithms for behavior-based systems. It may even be impossible to find suitable controllers for complex tasks with multiple and potentially conflictive solutions. Furthermore, it is not possible to acquire complex world models with the above-mentioned methods. For a number of such problems, the research community developed highly optimized task-specific algorithms. The following paragraphs detail selected robot tasks and their solutions. The selection depends on whether the task can be investigated in isolation and whether established solutions exist.

Self-localization and mapping

Mobile robot localization is the problem of estimating the robot's pose relative to the environment and plays a major role in the success of mobile robot systems. According to (Cox and Wilfong, 1990),

> Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities.

The complexity of mobile robot localization differs in whether the robot's initial position is known (Wei et al., 1994; Borenstein et al., 1996) or not (Fox et al., 1998; Thrun et al., 2001). In recent years, the significantly harder problem of simultaneous localization and mapping (SLAM), i.e. no information about the environment is available, has been tackled. (Montemerlo et al., 2003) implemented the efficient algorithm fastSLAM2.0 which is successfully applied in several robot systems and marks the current state of the art.

Although the problem of self-localization and mapping is widely considered to be solved, there is still much ongoing research aiming at increased efficiency and accuracy. Especially SLAM based on visual sensors received much attention recently, e.g. the IEEE Transactions on Robotics published a special issue on visual SLAM in October, 2008.

Mobile path planning

The goal of path planning is to guide the mobile robot from its initial position to another position i.e. the goal point. The desired path has to fulfill certain criteria such as low energy consumption, shortest overall distance or maximal safety. Supposably, the most common method to find a path for mobile robots is to transform the geometric properties of the robot's circumstances into a directed graph representation. Arbitrary optimality criteria can be coded into the weights of the graph's edges. There exist a number of proven algorithms for path planning on graphs. The A* algorithm (Nilsson, 1982) finds the shortest path on a weighted graph if underestimating heuristics are used. Other common path planning methods are potential fields, grid-based methods or sample and random-tree search. See (Sedgewick, 2001) for an overview of graph planning algorithms.

People tracking and trajectory prediction

For the purpose of human-robot interaction it is important for the robot to know the location of its interaction partner. Furthermore, it is important to predict people's future trajectories for e.g. efficient and unobtrusive path planning or delivery tasks.

(Kleinehagenbrock et al., 2002) proposed people tracking in laser range data and camera images. A similar approach is used in (Kobilarov and Sukhatme, 2006). In an outdoor environment, a robot tracks people in laser range data and omni-directional images with the purpose of following them.

(Weser et al., 2006a) developed a framework to track multiple people simultaneously using information provided by an arbitrary number of tracking modules that implement a certain interface. To show the feasibility of the proposed approach, tracking modules are implemented for an overhead camera and two laser range scanners as sensor modality. It was shown that the future trajectory of tracked persons can be predicted using incrementally learned and generalized motion patterns.

The work of (Bennewitz, 2004) integrates predicted human trajectories into A* path planning. Paths generated for delivery tasks are planned towards the estimated future location of the recipient at delivery time. Other tasks introduce non-interference with predicted human trajectories as additional optimality criteria for the planning algorithm.

Robot arm kinematics

Robot arm kinematics deals with the analytical description of the spatial displacement of the robot arm as a function of time, in particular the relations between the joint-variable space and the position and orientation of the end-effector of a robot arm.

Robot arm kinematics state two fundamental problems: the *direct kinematics* refers to the calculation of the position of the robot arm given the joint variables. If the task is given in a reference coordinate frame, the *inverse kinematics* is applied to find appropriate joint variables. For a detailed problem description and possible solutions see e.g. (Fu et al., 1987).

The inverse kinematics problem is much more frequently used since positions of objects under manipulation are usually known in cartesian coordinates. Still the inverse kinematics problem is developed even further. Since redundant manipulators provide several solutions for most configurations in the coordinate frame, different optimality criteria are explored. (Klanke et al., 2006), for example, propose to decompose the inverse kinematics of a seven degree of freedom (DOF) arm to a redundant 6-DOF problem in a 5D configuration space. Due to explicit parameterization of the arm's redundancy, the algorithm can dynamically react to arbitrary changes of the environment.

This and other examples show that different strategies to handle redundancy are still developed. The kinematics engine that drives the robot arm used in this project is based on the robot control c library (RCCL) by (Lloyd and Hayward, 1992).

3.2. Robot planning

The previous section described how robot actuators can be controlled to generate a desired effect. It is not yet discussed how the robot decides what action should be executed i.e. which effect is desired given an abstract task or goal. The part of the robot's software that breaks down abstract goals or tasks into behaviors that supposably achieve the robot's goals is widely referred to as *deliberative component*.

One possibility to endow the robot with a deliberative component is to predefine solutions for challenging tasks. (von Collani, 2001), for example, coded world states and events in a script representation that is used to generate the events required for complex tasks. Using this practice, the programmer has to foresee every possible combination of robot goals and initial situations.

More flexible is the usage of planners that generate solutions for complex problems based on general rules. The next section summarizes the current state of the art in AI planning. The integration of planning mechanisms with robot systems is not a trivial task. Several assumptions made in traditional artificial intelligence research do not hold if they are applied to embodied robots. Section 3.2.2 reviews specific requirements and techniques for plan-based robot control. How perception, planning and action may be integrated in natural systems is subject to research on cognitive architectures as described in Section 3.2.3. The combination of several hardware devices and algorithms that may require real-time performance, storage of large data sets and high computational belong to the field of robot software systems. Robot software systems are discussed in Section 3.2.4.

3.2.1. Traditional AI planning

The problem-space hypothesis (Newell and Simon, 1975) assumes that problems can be described using a state space and a set of discrete actions that transform states to successor states. Current states and goal states are represented by sets of symbolic literals and relations among them. Thus, a plan is defined as an action sequence that changes the prevailing situation successively until a state is reached that meets the goal conditions. Although this traditional approach to planning has been investigated since the early seventies, it is still the basic principle of current AI planners.

Several definitions of the term *plan* can be found in the literature. In the following discussions on planning, a definition for plan will be used as follows.

> A *plan* is a scheme, program, or method worked out beforehand for the accomplishment of an objective[4]

The following paragraphs categorize different types of planners and review representative implementations.

Domain specific planners

This type of planners are made or tuned for a specific type of application. They will not work well – if at all – in any other domain. Many successful real-world planning systems work this way. The remarkable success in games that require complex planning like chess or bridge (Levy and Newborn, 1991; Smith et al., 1998) show the strength of such planners. Other every-day applications of domain-specific planners can be found in industry, e.g. in sheet metal bending machines where a software plans the sequence of bends (Gupta, 1999).

The lack of flexibility in their application necessitates laborious adaption or reimplementation for every new domain. The robot control algorithms described in Section 3.1.3 may be seen as domain specific planners as well. Path planning, for example, plans a sequence of motion that is supposed to achieve the objective of reaching a goal in an optimal way.

[4]http://www.dictionary.com, accessed on 25 April 2009

Multi-purpose service robots will be capable of achieving several different tasks. Thus, the applied planning algorithms have to account for these different characteristics. Delivery services, automatic surveillance and clearance show completely different demands for planners. These examples show that domain specific planners can hardly be applied to the deliberative component in service robots.

General purpose planners

In principle, a general purpose planner works in any planning domain. It uses no domain-specific knowledge except for the definitions of the basic operators. Thus, it cannot benefit from particular strategies for planning in certain domains.

In practice, it is not feasible to develop planners that really work in *every* possible domain. Several assumptions are made to restrict domains to classical planning domains: the system has to have finitely many states, actions and events; it has to be fully observable, deterministic and static, and plans have to be sequential with implicit time durations.

These assumptions restrict the problem to path-searching in a graph, where nodes are states of the world and edges are actions. (Fikes and Nilsson, 1971) introduced the Stanford Research Institute problem solver (STRIPS) that is widely referred to as the first classical automated planning system. At that time no heuristics could be found to guide the search through the space of possible actions. Thus STRIPS quickly becomes intractable if applied to complex problems. More recent planners utilize heuristic search and thus provide significantly better performance.

(Bonet and Geffner, 2001) introduced the heuristic search planner HSP2.0. HSP planning is based on the idea that planning problems are mapped into search problems in a suitable space, which are solved using a heuristic function extracted automatically from the problem encoding. A unique feature of HSP2.0 is its ability to plan either progressively or regressively from initial state to goal state. An implementation of HSP planning with simplified heuristics was introduced by (Hoffmann, 2001). This planner performs better for simple tasks but gets lost in local minima if applied to more complex domains.

Other general purpose planning approaches are Graphplan (Kambhampati et al., 1997), SATPLAN (Kautz et al., 2004), BLACKBOX (Kautz and Selman, 1998) or the temporal planner SAPA (Do and Kambhampati, 2003).

Configurable planners

The flexibility of domain-independent planners is at the expense of efficiency. (Bylander, 1991) showed that even with reasonable restrictions, general purpose planning is at least PSPACE-hard. This makes general purpose planning a less attractive choice in real-world domains. Configurable planners overcome these problems by providing a domain-independent planning engine while accepting input that includes information about how to solve problems in the domain.

The knowledge how to solve problems can be coded in Hierarchical Task Networks (HTN) that provide information on how to decompose abstract tasks in sets of more simple tasks until atomic actions are reached. (Nau et al., 2003) proposed the HTN planner SHOP2 in which a domain is specified by describing the robot behaviors in terms of template tasks together with methods for recursively decomposing template tasks down to primitive tasks.

This type of planners are widely used in robots that are equipped with autonomous planning capabilities. A customized planner that combines principles from HTN planning with temporal aspects of SAPA was developed by (Beaudry et al., 2005) for the autonomous robot Spartacus. For other examples of autonomous robots equipped with planning capabilities see Section 3.6. A java-based implementation of the SHOP2 planner (Ilghami and Nau, 2003) is used for the robot system implemented in this work. Reasons for this choice are given in Section 6.2.

3.2.2. Plan-based robot control

A robot control system must cope with data in many different grades of granularity, from sensor readings to user-supplied mission data. It must yield purposeful action on different time scales, from fast reflexes (Section 3.1) to optimal long-term task organisation (Section 3.2.1). Accordingly, a robot control program needs a special structure and organization that integrate these incoherent pieces into coherent overall action (McDermott, 1992). Plan-based robot control systems amalgamate plan generation on an explicit symbol level, plan execution working in close sensor-motor coupling, and execution monitoring. A typical approach to integrate reactive and deliberative components are hybrid systems.

Hybrid systems combine different layers for reactive and for deliberative control components. Typically, a middle layer (often called sequencing layer) mediates between the reactive and the deliberative components, resulting in a three-layered architecture. The concept of a hybrid deliberative and reactive architecture is originally attributed to (Arkin, 1987) and was implemented in AuRA (AUtonomous Robot Architecture). The reactive component of AuRA is based on motor schemata that define behavioral processes. The deliberative component consists of a mission planner, spacial planner and the plan sequencer. AuRA is modular and flexible and has been applied to numerous robot systems (Konolige et al., 1993; Chung et al., 1998; Low et al., 2002, etc.).

An advantage of a layered architecture is that the deliberative layer is relieved from the pressure to produce real-time responses to changes in the environment. Only major changes that cause the reactive execution to fail initiate deliberative replanning (Ghallab et al., 2006).

The plan-based control architecture DD&P is presented in (Schönherr and Hertzberg, 2002). This architecture shows the classical separation of a deliberative and a behavior-based part. Interestingly, the hierarchy among the layers reflects only degrees of abstraction of what they deal with. In the control flow of the overall system both layers are equitable. In the same way that the deliberative component drives the reactive execution, the reactive execution can influence the deliberative component. Although this idea is obvious (what happens now can change my future plans and my plans determine what happens now) it has not been stated explicitly before.

Scalability and flexibility are the primary goals of the behaviour-based robot research architecture (BERRA) (Lindstrom, 2000). The hardware abstraction is divided into controllers and resources: controllers represent actuators and command handling, resources represent sensors and sensor fusion modules. A task execution layer mediates among hardware abstraction and deliberative components. An actual implementation of a deliberative component is not included in the architecture. Since BERRA's clear component-based structure promotes extension and testing of planning components, this architecture is attributed to plan-based robot control systems.

An approach to integrated planning and control without a layered structure is proposed in (Beetz, 2001). Structured reactive controllers (SRC) are control programs that are specifically designed to be transparent and modular so that automatic planning processes can reason about and revise them. SRCs are hierarchically organized as they are responsible for high-level plans as well as for low-level behaviors. SRCs consist of several behavioral modules and plans how to use these modules on different abstraction levels. In principle, all SRCs can reason and revise other SRCs to generate robot behavior. Thus, SRCs provide a framework for integrated planning and control rather than being a layered architecture.

Apart from (Beetz, 2001), the only approaches to seamless integration of reactive and deliberative components into, say, a black box that reasons and executes behaviors are artificial neural network driven robots. These, however, do still not reach the level of abstraction that is needed for real service robot applications.

3.2.3. Cognitive architectures

The main purpose of cognitive architectures is enhanced understanding of human cognition. Researchers in cognitive psychology hope to support their theories by emulating phenomena that have been observed in humans. Cognitive architectures posit fixed sets of computational mechanisms that putatively underlie a wide range of human cognition. Since they initially did not address the purpose of being implemented on embodied systems, they focus solely on the cognitive processes and neglect necessities that arise from their application to real robot systems. Thus, this line of research goes in parallel with robot planning and control architectures. Since cognitive researchers have recently observed that cognition cannot be separated from embodiment, these architectures have been applied to robots, too.

The question of whether architectures are attributive to cognitive architectures or to plan-based robot control systems as described in the previous section cannot be answered conclusively. The distinction which is made here is based on the initial purpose of the architectures. The robot architecture BERRA, for example, was designed to be flexible and scalable and to enable robots with enhanced autonomous capabilities. For this reason, BERRA was described in the previous section on plan-based robot control. The SOAR cognitive architecture (Laird and Congdon, 2006), in contrast, was originally intended to model human cognition and later on applied to robots.

The SOAR architecture provides a platform to investigate the representation of knowledge and inference mechanisms based on that representation. It is probably the most important cognitive architecture and shares many properties with other architectures e.g. ACT-R (Anderson, 1993), ICARUS (Langley et al., 1991), EPIC (Kieras and Meyer, 1997) and PRODIGY (Carbonell et al., 1991). These architectures have collectively been applied to a broad set of phenomena in cognitive psychology.

Newer approaches weaken the purely cognitive character of such architectures and include modules for perception and action generation. This important step towards embodied cognitive robots bears the risk of losing the independence of robot hardware. The SOAR architecture, for example, was extended with plugins to incorporate robot controllers (Benjamin et al., 2004). A formal model of concurrent sensory-motor activity as well as an algebraic theory of task decomposition and reformulation was integrated to a new framework called ADAPT. This framework was implemented on a Pioneer P2 robot (Benjamin et al., 2006).

The SAPHIRA architecture (Konolige et al., 1997) was originally designed for the robot Flakey (Konolige et al., 1993). Now, it is used for ActiveMedia[5] platforms. The architecture coordinates the planning of operations, motor controls and sensing operations. Physical sensors are handled directly by SAPHIRA and it is not designed for hardware independence. The central aspects of SAPHIRA are the ability to interact with other agents and to take advice about the environment.

A Situated Artificial Communicator (SAC) is proposed in (Zhang and Knoll, 2003). The purpose of the SAC is to receive assembly instructions in unconstrained natural language and to execute them in a physical robot environment which consists of two stationary manipulators equipped with pneumatic jay grippers, force/torque sensors, and hand cameras. From the instructor's point of view, the SAC should resemble a human communicator as closely as possible. The instructor is therefore modeled as an extension of the robot's perceptual system. Information about human gestures and gaze is incorporated into world knowledge which is obtained by various static and articulated cameras (Zhang et al., 1999). In addition to the disambiguation of situated and possibly incomplete human instructions, a layered learning approach for operation sequences for two arms in the context of cooperative tasks was proposed.

The architecture EMIB focuses on the selection of behavior-producing modules based on emotion and motivation (Michaud, 2002). It has been applied to a simulated robot

[5] http://www.activrobots.com, accessed on 25 April 2009

without emotion, a mobile robot that has to learn in a non-stationary environment from observation, and to a mobile robot interacting with humans using light signals.

3.2.4. Robot software frameworks

Many complex robot problems can be solved if they are investigated in isolation. Further developments can in principle be based on these results, however, every group that builds complex robot systems faces the same problem: the system integration. To make the investigation of complex new problems possible, the experimental platform has to provide a set of less complex behaviors as well as several dedicated software modules to control different hardware devices. Some of the software modules are coupled in tight perception-action cycles and demand for near real-time execution. Software modules need to be executed in parallel and have to communicate mutually. Robot software frameworks attempt to establish flexible communication protocols for the exchange of sensor data and control commands as well as to provide a middleware for robot algorithms and hardware controllers.

Problems that have been occurring in multi-robot environments (Ferch, 2001) nowadays occur in single robot systems with multiple actuators. In multi-robot environments, a communication protocol has to be established between independent machines whereas the communication in multi-actuator platforms takes place within a single operating system. This may simplify the sharing of information, but problems will occur when real time capabilities need to be assured. Collision avoidance, trajectory generation and interaction planning have to be considered within one system and share computational resources.

Robot software frameworks attempt to simplify the task of putting different soft- and hardware modules together. In doing so, their main purpose is applicability in several robot setups. They are often specifically designed to enable integration of deliberation, planning and behavior of robots. The fact that only a few software frameworks are applied to many different robot systems shows the difficulty in designing frameworks that are really portable to different robot environments. The following paragraphs present software frameworks that have been applied successfully to robot systems.

The Roblet-Technology

The Roblet-Technology provides a framework that minimizes the effort for developers to write distributed systems. This architecture was introduced by (Westhoff et al., 2006) and fits the requirements of robot programming nicely. It makes the writing of distributed control or monitoring programs possible which are referred to as Roblets that are sent to Roblet-Servers. The servers can run on different computers within a local area network. Thus, complex algorithms can be executed easily on different machines which provide the necessary computational capacities. The framework itself does not support hardware devices at all. Instead, it simplifies the encapsulation of native hardware-dependent libraries and makes them accessible via interfaces for abstract robot functionality. Interfaces provided by Roblet-Servers can be seen as a hardware abstraction layer since the underlying implementation and thus the currently used hardware does not have to be known.

The integrated software system for the mobile service robot TASER that is presented in this work is implemented using the Roblet-Technology (Baier et al., 2006). Currently, there are over ten Roblet-Servers up and running.

Player

Player is a device server that provides socket-based access to a wide variety of robot sensors and actuators (Gerkey et al., 2003). It is similar to the Roblet-Technology in that multiple client programs can communicate with the devices concurrently. It differs from Roblets in that Player exchanges messages with the clients while Roblets are code fragments that execute on the target machine. Because Player's external interface is simply a TCP socket, client programs can be written in any programming language that provides socket support. A drawback of this architecture is that it is strictly client-server based. All clients have to connect to one single Player server. Player comes with a set of drivers for various devices, in contrast to the Roblet-Technology that focuses purely on the architectural structure. Player thus can be regarded as a robot library in addition to its architectural character.

Component framework for autonomous robots

A component framework for autonomous robots is proposed in (Orebaeck, 2004). This attempt to provide a general framework for the control of robot hardware adopts a component-based software engineering approach (CBSE) that encapsulates the details of each component. Components can constitute hardware encapsulation, behavior generation or deliberative intention. The communication among components is realized with CORBA.

Although the framework tries to be as generic as possible in terms of hardware abstraction, the author himself admits that parts of the system are "too much hard-coded"(Orebaeck, 2004, Section 10.3) and that it has to be modified in the case that new types of sensors are needed.

The Task Description Language

The *task description language* (TDL), introduced by (Simmons, 1992), is a syntactic extension of the C++ programming language. In a twofold compilation process a preprocessor first generates pure C++ code which includes a task management library. The description language is not tied to any robot hardware since the intermediate C++

code can be compiled on any operating system (Simmons and Apfelbaum, 1998). TDL is a three-tiered robot control architecture that naturally detaches a symbolic and an analogous layer:

- the behavior layer (real-time control) interacts with the physical world, controlling actuators and collecting sensor data.
- the planning layer specifies, at an abstract level, how to achieve goals and how to deal with goal interactions.
- the executive layer mediates between the symbolic level of the planner and the discrete level of the behaviors.

This approach preserves all flexibility of a native programming language while a syntactical guidance for programmers demands a clear definition of robot tasks. A tool for automatic verification of task description models is available (Simmons et al., 2000). This tool verifies formal models derived from TDL code for logical inconsistencies. TDL was extended with a scheduling component and applied to a reconfigurable planetary robot system (Wang et al., 2005).

3.3. Grounding of action and perception

The *symbol grounding problem* (SGP) is concerned with the question of how symbols get their meaning. (Harnad, 1990) adopts a theoretical view and claims that all kinds of intrinsically meaningless symbolic representations (e.g. situations, actions, relations, temporal constraints etc.) have to be related to their non-symbolic denotata in the real world. Although behavior-based robots as described in Section 3.1.2 produce astonishing behavior completely without symbolic processing, it is nowadays widely accepted that symbolic reasoning capabilities are needed to render complex autonomous robot behavior possible. Harnad himself proposed a hybrid connectionist/symbolic approach to the SGP, which consisted of bottom-up grounding of symbols in iconic and categorical representations.

Language acquisition as a special case of the SGP is discussed in the next section. It concerns the grounding of spoken (or written) words. Sub-problems of the SGP with particular relevance for robotics are the ability to relate symbols to robot actions (Section 3.3.2) and to relate symbols to external objects (Section 3.3.3). The latter is also called symbol anchoring.

3.3.1. Language acquisition

Language acquisition in robotics has been focusing most prominently on vocabulary for external objects. Furthermore, the choice of objects is usually restricted to those objects that are visually observable. The talking head experiment (Steels, 1999) was designed to demonstrate the feasibility of bootstrapping a communication system among agents situated in a shared environment. Two years later, (Steels and Kaplan, 2001) show that social interaction helps focusing on what needs to be learned. The authors used the notion of a language game to teach a robot words with human mediation. A dialog approach for learning semantic categories and word acquisition with visual perception for grounding objects is proposed in (Holzapfel et al., 2008). Contrary to Harnad's claim that a robot has to autonomously establish the semiotic map that grounds symbols in the world, (Roy, 2005) propose a framework for grounding language in the world that is partly based on explicitly modeled sensory motor schemas.

All known approaches to autonomous language acquisition are very limited concerning the size of acquired vocabulary. (Steels, 2007) claims that the SGP is solved, still several authors agree on the need to scale up possible vocabulary size (Cangelosi, 2006).

3.3.2. Grounded robot actions

The problem of grounding symbols that denote robot actions appears trivial since the external effects of robot controllers are known in most cases – at least in probabilistic terms. However, the strong abstraction from worldly details at symbol level makes a translation into physical behavior non-unique. The effects of robot control programs are context dependent to a certain degree, which is not reflected in the symbolic operator descriptions. For example, the action of reaching for an object with a manipulator appears to be well defined. Inverse kinematics for a redundant manipulator may provide several different motion trajectories to reach the goal. Thus, a symbolic description of a reach-action without additional specification of certain arm configurations is not uniquely grounded in its physical effect. Grounding symbols for actions is an area of active research.

In the DD&P robot control architecture, (Schönherr and Hertzberg, 2002) propose a loose coupling of symbolic identifiers to robot actions:

> Executing an [symbolically represented] operator means stimulating more or less strongly the behaviors working in favor of the operator, and muting those working against its purpose.

This gives some degree of freedom to the underlying behavior-based system to react to unforeseen situations e.g. obstacles. However, in the case of precisely specified actions, such as pressing a doorknob, the concurrent execution of different behaviors will degenerate to a single one.

A neural model, based on mirror neurons, that is able to learn the meanings of three action labels through imitation is described in (Wermter et al., 2005). The group of (Steels and Spranger, 2008) show how a group of humanoid robots acquire a model of their bodies, such that they agree on a set of symbols to describe body movements. A combination of imitation and language games led to these results.

(Saffiotti and LeBlanc, 2000) proposed grounding of robot gaze control for a team of robots in the RoboCup '99 competition. Is was shown that information on gaze movements can be represented along with the information on the perceptual appearance of the object to be grounded.

An affordance-based view on grounding of planning operators is proposed in (Lörken and Hertzberg, 2008). It is argued that if the robot perception process does include perception of affordances, then grounding of actions can be reduced to grounding of affordances and thus rely on the affordance-behavior coupling to guide the physical behavior.

Every robot system that integrates a symbolic planning component has to relate symbolic identifiers for actions to the routines that drive the actuators to produce the actions. As shown in the literature, no standard method has been established yet.

3.3.3. Symbol anchoring

Most robot actions manipulate or refer to objects in the robot's environment. Thus, a robot needs to be able to refer to objects outside itself. However, variables can be assigned to values in programs, but variables cannot be assigned to external objects. To quote (McDermott, 1992),

> Variables can at best be bound to descriptions of objects, that is, to information sufficient to manipulate and reacquire them.

Furthermore, sensor data are inherently incomplete, inaccurate, ambiguous and even faulty, which makes the problem of relating internal symbols to external entities non-trivial (Müller, 2008).

The sub-problem of the SGP to ground physical objects is widely referred to as *symbol anchoring* (SA). The first domain independent definition of the anchoring problem was given in (Saffiotti, 1994), while the first attempt at a computational theory of anchoring was reported in (Coradeschi and Saffiotti, 2000).

Symbol anchoring is concerned with the problem of how to create and maintain in time the connection between symbols and sensor data that refer to the same physical objects. In (Coradeschi and Saffiotti, 2003) a systematic approach is presented that in principle is capable of linking perceptual data to symbols of arbitrary complexity. One assumption in this architecture is that low-level perception routines are capable of splitting up the sensory stream into sets of percepts that are assumed to originate from single physical objects. This assumption makes clear that no consideration of size, partonomic relations and object complexity has been made. There is no way to decide whether a visual clue is the percept originated by e.g. a doorknob or just a part of the percept for the whole door. At this point, either top-down information has to be introduced to guide the perceptual focus (see Section 3.4.1) or a general contract of the complexity of considered objects has to be made (Weser and Zhang, 2007).

While it is reasonable to enrich symbolic representations of objects with physical properties e.g. color, shape etc., this is not sufficient to ground the symbol in the world. This description might be sufficient for a purely cognitive agent to combine symbols based on physical properties to yet more complex symbols, e.g. combining stripes and a horse leads to a zebra. This combination is, however, of purely syntactic nature since the physical properties (stripes, horse-shape) have no meaning to the agent. How an agent can perceive a physical object depends on the agent's sensoric capabilities in the same way as on the properties of the object itself. This leads to the idea that the physical properties used to ground a symbol have to be either specifically designed for certain kinds of agents or they have to include all potentially useful information i.e. a complete physical model. The latter is obviously not possible. Thus, in practice symbols are enriched with just enough information to be perceived by the targeted robot platform.

3.4. Robot perception

Autonomous robots perception is a major prerequisite for interacting with and operating in an unstructured environment without the explicit control of a human operator. Perception allows a robot system to recognize the state of the world and to continuously update its own model of the environment.

Symbol anchoring, as proposed by (Coradeschi and Saffiotti, 2003), makes some strong demands on the capabilities of the perceptual system, e.g. it has to be able to identify subsets of the whole sensory stream that originates from single objects. This, and other arguments, show that this framework is hardly applicable to complex real-world domains. Top-down knowledge has to be provided to the perceptual system to guide the focus of attention.

To increase the performance of the perceptual system, the sensors can be directed actively on presupposed target objects to obtain more relevant sensor data. This process is called *proactive robot perception* and is discussed in the next section. Another commonly used approach to improve robot perception is the integration of several sensor modalities with complementary characteristics as described in Section 3.4.2.

3.4.1. Proactive robot perception

Proactive perception is the natural approach to make use of top-down knowledge in perceptual processes: the agent deliberatively focuses its sensors and algorithms on the assumed target objects. Perception becomes an active process that has to be initiated on purpose rather than an ambient service that is constantly active.

The next best view problem and active robot localization can be seen as prominent instances of proactive perception. In automatic 3D model reconstruction, a camera is moved around an object to capture images that provide complementary information. (Li et al., 2005) propose a method that automatically determines the next best view from the partial object model and the last two successive viewpoints. Active robot localization refers to the idea that a mobile robot changes its position in the service of localization. In the case of unresolved uncertainties in localization, the robot can perform a motion that will provide a maximum of information about its position (Beetz et al., 1999; Thrun et al., 2001). An active vision approach to track landmarks for mobile robot navigation is proposed in (Davison, 1998). It has been shown that information gained from serial fixation on a succession of features can be used for strategic navigation.

(Wasson et al., 1998) propose active gaze control for a stereo camera system. By analysing only small subsets of the visual field, more complex processing algorithms can be applied. The perception results are integrated in a comprehensive representation by means of a visual memory. A similar approach is proposed in (Saffiotti and LeBlanc, 2000). The symbol anchoring framework by (Coradeschi and Saffiotti, 2003) is extended to include information about the current perceptual needs of the controller into the anchors. This enables perceptual processes to focus actively on important aspects of the world.

Compared to perceptual processes without physical motion, proactive perception is rather slow. This drawback can be eased if perception routines of different quality are combined. (Weser and Zhang, 2007) propose to use week assumptions arising from simple but fast measurements to narrow the search space for more time-consuming active perception routines. This allows for anchoring of objects using e.g. haptic exploration in an interative approach. This idea was developed as part of this thesis and will be described in more detail in Section 7.2.

3.4.2. Multi-sensor robot perception

The synergistic use of multiple sensors promises increased performance in robot perception. Different sensors cope with different aspects of the world and complement one another if combined properly.

(Luo and Kay, 1995) classify the combination of different sensor modalities into *multi-sensor integration*, which is the synergistic use of information provided by multiple sensors to assist in the accomplishment of a task, and *multi-sensor fusion*, which refers to an actual combination of different sources of sensory information into one representational format.

Multi-sensor integration

Multi-sensor integration takes place in almost every robot action. Since most actions are guided or monitored by sensors, there are at least two modalities involved: the actuator and the sensor. If manipulative tasks are to be achieved by a mobile robot, there are even more modalities involved. Opening a door (Kim et al., 2004) or pushing a wheel chair (Rubrecht et al., 2008) requires at least a robot manipulator, the mobile platform and sensors to measure the position of the target object as well as the success or failure of the action.

Multi-sensor fusion

In the context of symbol anchoring there is always a joint representational format to be used, that is, the data structure of the object to be anchored in sensor signals. Fusion of data that originate in one object requires the involved sensors to provide a geometrically overlapping workspace. Frequently used sensors for fusion applications are sonar or laser rangefinders and different camera systems. A common problem to be solved is the mutual calibration of sensors in space (Zhang, 2004; Mei and Rives, 2006; Kobilarov and Sukhatme, 2006, etc.). Carnegie Mellon University provides a laser-camera calibration toolbox, a Matlab®-based graphical user interface to an easy and portable technique for external calibration of a camera to a laser rangefinder[6].

Laser range data and camera images are often fused due to their complementary properties while being easily directed at a common workspace. (Weser et al., 2006a,b) fuses these sensors to track multiple persons in an indoor environment. While laser range data provide better accuracy in the position of the persons, the camera images can be used to distinguish different persons. A technique to fuse visual and depth information acquired by a camera and laser range scanner is proposed in (Scheibe and Scheele, 2004). The purpose of this work is visualization and surveying, however, the authors admit that this system works only in fairly controlled environments.

3.5. Robot learning

Learning is one of the fundamental abilities that a system must possess in order to be considered intelligent. It may be generally defined as the process of improving behavior based on experiences. Similar to planning capabilities, learning enables a system to form behavior that was not foreseen explicitly by the programmer. The following sections provide a brief overview of different types of robot learning. The distinction is based on the type of information that is learned and its effect on the robot's action in the world (Connell and Mahadevan, 1993).

3.5.1. Parameter learning

This type of learning optimizes numerical functions for calibration of sensors in an existing structure. Parameter learning may be considered as the weakest form of learning since a lot of information needs to be built-in while the amount of learned information is quite low.

Depending on the application, several specific methods are used to learn the system's parameters. Camera calibration, for example, is a domain that can be satisfactorily modeled with mathematical functions that take the system parameters, such as focal

[6]http://www.cs.cmu.edu/~ranjith/lcct.html, accessed on 25 April 2009

length, viewing angle, distortion and others, into account. The parameters can be learned if known properties of the world are compared to obtained images (Tsai, 1986).

If the learning domain is not known in such detail, general function approximators are used to learn parameters of the system. Commonly used methods are neural network based approaches, cf. (Haykin, 1998), or fuzzy learners, cf. (Russo and Jain, 2000).

3.5.2. Learning about the world

This category of learning is the most widely used in robotics. Contrary to parameter learning, in this category some kind of representation about the world is constructed.

Trajectory learning and robot mapping are introduced in Section 3.1.3 as optimized task-specific algorithms for robot control. These applications are prominent examples of learning about the world. Another example is the learning of object appearances. The most commonly used sensor modality for object recognition is vision. However, the appearance of objects cannot be represented sufficiently for recognizing them in a wide range of perspectives, lighting conditions, etc. Thus these representations are shaped by learning methods. (Pope and Lowe, 1996), for example, proposed a method to learn scale-invariant feature vectors (Lowe, 1999) of objects in a probabilistic way.

Whenever knowledge about the environment is generalized from direct perception, this process can be attributed to this learning category. Perception of human trajectories is generalized to trajectory patterns, perception of empty space is generalized to maps, and several images of certain objects lead to a reusable representation of the object's visual appearance.

3.5.3. Task level learning

Learning to solve new tasks requires constructing goal-directed sequences of primitive actions. As will be described in the next section, this complies with the central goal of robot planning. Task level learning thus includes learning of heuristics or domain knowledge that guides planning processes.

Reinforcement learning (Sutton and Barto, 1998) has been used to learn behaviors in the sense of learning policies for goal-achieving sequences of actions. However, the results of reinforcement learning are flat policies without any hierarchical abstractions of state-action pairs.

Another line of research is instance-based learning. In contrast to learning methods that construct behavior strategies by means of a general target function when training examples are provided, instance-based learning methods store the training examples if they provide sufficient information to solve future problems (Mitchell, 1997).

Although instance-based learning is a widely studied topic in AI, computational performance problems in long-term experiments remain an open issue (Francis and Ram, 1993). A common explanation for performance problems is the trade-off between utility and cost of learned structures. While learning may simplify problem solving on the one hand, it certainly increases the time that is needed to explore the acquired knowledge. (Kennedy and Trafton, 2007) report computational performance problems in two important cognitive architectures (SOAR and ACT-R, see Section 3.2.3) when the amount of learned knowledge is increased. This, and the difficulty of lower-level issues, makes learning of new behaviors for robots a difficult task.

3.5.4. Autonomous mental development

While task level learning concerns learning of complex behaviors by means of sequences of basic robot actions, autonomous mental development (AMD) includes learning of the basic actions itself. It comprises not only the optimization of existing structures like parameter learning, but also the evolution of behavioral structures from scratch given nothing but a general developmental program (Weng et al., 2001).

(Konidaris, 2008) proposes to learn robot skills autonomously by using the options framework. This extension of the mathematical framework of reinforcement learning and Markov decision processes includes closed-loop policies for taking actions over periods of time (Sutton et al., 1999). Different types of Markov decision processes are evaluated with respect to their potential use as basis for AMD in (Weng, 2004). This evaluation concludes with the insight that the realization of higher cognitive capabilities based on such models has not been demonstrated yet.

An evolutionary approach that is supposably the most closest to real AMD is proposed in (Waibel, 2007). It was demonstrated that complex robot behavior can be learned from scratch by selection and mutation of successful robot instances. These experiments have been carried out in simulation and with real robots.

Although AMD is perhaps the hardest type of learning, it has not received much attention yet. Except for evolutionary approaches, the challenge of AMD remains untouched until now.

3.6. Service robot experiments, variability and scalability

Today robotic applications are moving more and more to everyday environments like private households. For example, there already are commercial robots for vacuum-cleaning (Forlizzi and DiSalvo, 2006). However, such robots can only perform very limited tasks, usually restricted mostly to safe navigation without destroying anything (Shieh et al., 2004; Fung et al., 2003). Research on complex multimodal service robots in natural environments is still rare. Most successful experiments are carried out in restricted domains with particular robot activities as research subject. This is hardly enough to identify the cognitive capabilities needed for perceiving, interpreting, analysing, and executing robot behaviour autonomously in unrestricted natural environments.

The problem of competently accomplishing everyday manipulation activities, such as setting the table and preparing meals, is addressed in (Müller, 2008) as a plan-based control problem. A library of general and flexible plans for a household robot is proposed and verified in comprehensive experiments with a simulated robot in a simulation environment with realistically simulated action and sensing mechanisms. The encountered scenario is clearly beyond what current robot systems can achieve today. Although many virtual robots model the kinematics and dynamics of the world to search for goal-directed behaviour (LaValle, 2006), most of them ignore the robot's sensory systems and assume that the state of the world is known with certainty (Kemp et al., 2007).

Fortunately, researchers are focusing more and more on real-world experiments rather than on simulations. The AAAI Mobile Robot Challenge, for example, requires the robots to participate physically in the competition. (Simmons et al., 2003) showed that due to the clearly stated goals of the challenge, the possible courses of actions could be modelled accurately for an intentional planning system. The authors, however, admit several areas for improvement, such as human-robot interaction, visual perception and fast and accurate monitoring of the environment. The robot Spartacus (Michaud et al., 2006) integrates planning and scheduling, sound source localization, tracking and separation, message reading, speech recognition and generation, and autonomous navigation capabilities on-board a custom-made interactive robot for the competition in 2005.

The University of Massachusetts Amherst developed a mobile manipulator platform explicitly designed for variability. A software suite, which is based on a multi-objective task-level control framework, is used to integrate a variety of motion capabilities, including task-based force or position control of the end-effector, collision-free global motion for the entire mobile manipulator, and mapping and navigation for the mobile base (Katz et al., 2006). However, different robot tasks are investigated in isolation and the experimental environment is fairly controlled. A complete robot system for outdoor cleaning services is presented in (Nishida et al., 2006). The robot consists of two 5DOF arms with a 1DOF gripper each. Experiments in which the robot autonomously collects rubbish have been reported. Most of the mobile manipulation experiments which are presented in the literature focus on delivery tasks (Shieh et al., 2004; Fung et al., 2003) or cleaning tasks (Forlizzi and DiSalvo, 2006; Nishida et al., 2006). Another common point is that experiments are typically carried out in short time frames, with the remarkable exception of (Thrun et al., 1999), which do not allow conclusions to be drawn on their scalability to realistically long-term scenarios.

A rather unusual demonstration scenario for cognitive technical systems is introduced in (Beetz et al., 2008). The focus is on different activities in the assistive kitchen domain such as table setting, cooking and household chores. In addition to sensors and actuators that are mounted on the robot, the assistive kitchen domain includes the environment in perception and action processes. However, it is one of the rare attempts to provide an exhaustive experimentation environment that includes all aspects of robot perception and autonomous mobile manipulation.

3.7. State of the art summary and thesis scope

Research on autonomous service robots provides several challenges beyond robot applications in stationary environments, such as mobile navigation, manipulation under uncertainty and the need for planning and problem solving capabilities. Another difference to conventional industrial robots is that developments aim at the versatility of applications rather than increased performance in a certain class of applications.

To tackle the demand for versatility, robots are equipped with AI-based planning mechanisms. Hybrid systems have been widely accepted and adopted as the standard for the development of robotic architectures. They combine aspects from traditional AI, such as symbolic planning, reasoning and abstract knowledge representations, with reactive abilities to perform robustly under uncertainty and in dynamic environments. Hybrid systems are most commonly realized as layered architectures. A prerequisite for such architectures is that primitive operators of the planning layer have to be implemented in robot control programs that achieve physically what was planned symbolically beforehand.

Although not always stated explicitly, most of the successful robot platforms are based on a set of control programs that serve as an interface between the physical execution of actions and deliberative planning. These control programs differ significantly in the degree of abstraction at which they are implemented. In the case of complex control programs, the planning component is rather simple (Hanebeck et al., 1997). If control programs become more primitive, the planning component becomes an immanent part of the system (Thrun et al., 1999; Simmons et al., 2003; Asfour et al., 2006). Only few publications report investigations on robot actions in the role of an interface between the deliberative and reactive layer per se. (Kim et al., 2005) classified actions into a taxonomy based on the kinds of sensors and algorithms used during execution. The question of an appropriate granularity of actions is not addressed explicitly in the literature. A detailed discussion on this question is an important contribution of the present thesis. Chapter 4 provides rationals for basic robot actions at different granularities for the use in layered deliberative and reactive robot architectures.

The literature describes highly elaborated hardware controllers for robot devices and robust algorithms for various robot tasks. Still, there is currently no robot operating system available that can be used for a wide range of robot platforms. Due to diverse robot hardware and tight coupling of developed algorithms with prevailing experimental setups, developers have to implement large parts of basic hardware control, primitive robot skills and higher cognitive processes from scratch each time a new platform is set up. This is what has been done in the TAMS workgroup as well. It is time to put the pieces together into a coherent system that allows further investigations of high-level robot behavior. For this purpose, all available robot devices from the experimental platform TASER will be encapsulated in Roblet-Servers. Based on this unified access to robot devices, a library of basic robot control programs will be defined and implemented, which is described in Chapter 5. A symbolic description of the control programs will be defined that serves as basic operators for symbolic AI-based planning.

Many traditional AI planning systems have been proven theoretically and they have been verified and tested. Specifically, HTN planners have been applied in a wide range of applications. Still, many researchers build customized planners for robot systems. In Chapter 6 of this thesis it will be demonstrated how a ready-made HTN planner can be applied to the service robot domain.

One of the most challenging problems in autonomous service robotics is reliable robot perception. While robot manipulators can be controlled with high-precision to exact positions, in adverse situations the position of objects under manipulation can hardly be perceived at all. Recent advancements in robot perception are the integration of multiple sensor modalities with complementary properties and active exploration. The latter is only possible if sensors and actuators are cooperatively controlled, which requires elaborated and complex experimental platforms. Thus it has been described less frequently.

In this thesis, several examples for active exploration will be given. A combination of sensors and actuators on the control level will be described that enables the experimental platform TASER to perceive objects reliable, e.g. by touching their surface. Supported by symbolic planning, different actuators are used to focus several sensors to objects of interest. This allows for integration of laser range data, camera images and force measurements to perceive complex objects reliably. This goal-directed integration of multiple sensors and actuators on the symbol level requires a well-performing robot platform and provides a testbed to evaluate the overal performance of the implemented system. The performance of the implemented robot system will be demonstrated in Chapter 7 with such experiments on plan-based proactive perception.

4 Atomic Robot Skills for Plan-Based Robot Control

We can't solfe problems
by using the same kind of thinking
we used when we created them
Albert Einstein, 1879-1955

One of the striking conclusions drawn from literature is that an autonomous service robot has to have both close sensor-control loops and abstract planning that foresees the effects of the intended course of actions. Attempts to combine planning and control in one integrated homogeneous system are still of theoretical nature (Beetz, 2001), focus on specific applications or clearly do not reach the level of abstraction that is needed for applied service robots (Low et al., 2002). Some scientists claim that an architectural distinction between planning and control will not lead to truly intelligent and autonomous systems. Although their approaches may be cognitively more plausible or more promising for research on advanced embodied intelligence, still, this work favors a layered architecture for the following reasons:

1. Robotics is not only a tool for AI research, it also focuses on practical and robust solutions that will lead to products suitable for a wide range of applications. A layered architecture effectively supports debugging, maintenance and further development and thus promises a short time to market with technologies that are both state of the art as well as sophisticated and robust.
2. The separation of the two layers is not strict. The layers are tightly coupled via object models that provide symbolic descriptions as well as rich numerical representations to be used in perception and action processes. Control programs for physical action modify the symbolic representations and in turn symbolic plan descriptions modify the execution flow of control programs. Furthermore, neither of the two layers will have exclusive control of the overall system: an action can cause the planner to generate new solutions while the planner causes actions to be executed.

It is inherent to every layered plan-based robot architecture that basic operators which are used by the planning layer have to be related to control programs that drive robot actuators. This leads inevitably to the question of an appropriate abstraction level for robot skills that serve as an interface between deliberative planning and reactive robot control. Simple skills can be used flexibly in several tasks while being easy to implement. At the same time they increase complexity in the planning component, which acts as a bottleneck of the overall system anyhow. Complex skills, in contrast, can alleviate the planning layer while flexibility and reusability is decreased.

Although the question of balancing the complexity is inherent to every plan-based robot architecture, it has not been addressed individually in the literature. A conclusive answer for all kinds of robot applications cannot be given due to varying objectives and overall complexities of different target robot platforms. This chapter aims at providing arguments that have to be considered while designing atomic robot skills. Parts of this chapter have also been published in (Weser and Zhang, 2009).

The terms *atomic robot skill*, *primitive operator*, *robot action* and *control program* will be used throughout this work according to the following definitions:

Primitive operator A primitive operator constitutes the basic building block for deliberative planning systems. It specifies a desired change in the robot's circumstances[1] in abstract terms and may be parameterized if applied to concrete situations in the planning process. Primitive operators are represented in symbolic terms and do not account for details that result from physical execution. They will be defined formally in Section 6.1.1.

Control program A control program is the part of the robot's software that directly controls the hardware, i.e. sensors and actuators. It generates executable sequences of motor commands based on sensory input and internal believes about the environment. This makes a control program that particular piece of software that is invoked to actually change or perceive the world physically. Due to direct access to sensors and actuators, a control program can hardly be implemented in a platform-indepent way.

Robot action A robot action is the instantiation of a control program to a certain situation. It describes the physical movement which is caused by the execution of control programs under certain external and internal conditions. An action may also proceed without physical movements if perception or information services are intended and coded in the control program.

Atomic robot skill An atomic robot skill is the generic term for pairs of primitive operators and control programs. It describes robot actions in both symbolic representation and numerical implementation.

Atomicity of a robot skill does not refer to the execution of a control program, it rather refers to the operator's use in a planning component. Admittedly, a robot planner can influence the execution flow of control programs by parameterization, still it cannot decompose the operators into more basic building blocks.

[1]Circumstances include the robot's internal state, e.g. knowledge about the world. Thus a primitive operator may also specify perception routines.

The rest of this chapter is organized as follows. A general discussion of the role of robot skills in a hybrid deliberative and reactive robot control architecture is given in Section 4.1. Section 4.2 discusses the problem of an optimal granularity level for atomic robot skills. This section will present lists of arguments that favour skills of low complexity or abstract complex skills respectively. The last section of this chapter bridges from the abstract discussion on robot skills to a more implementation-oriented view. It proposes an internal representation for robot skills, external objects and relations among these with respect to the implemented robot software systems.

4.1. The role of atomic robot skills

Atomic robot skills have at least two roles. They are syntactic objects that form the basic building blocks for symbolic planning processes and plan representations (operators) and they are executable prescriptions (control programs) of robot behavior. This section describes properties that a skill must exhibit in order to be used in this duality. (Lyons and Arbib, 1989) state a number of features for robot skills that focus on actions and behavior generation.

4.1.1. Robot skills as executable robot control programs

A strong abstraction from worldly detail is required and unavoidable for effective planning on the symbolic level. For the part of acting concretely according to the current symbolic plan step, this abstraction makes a translation into physical behavior non-unique (Schönherr and Hertzberg, 2002). The executing control program must autonomously anticipate details for action execution based on the present situation that is perceived via sensors. This is what makes control programs complex pieces of software that demand certain properties. Important properties are described in the following subsections.

Real-time constraints and safety considerations

All robot motions involve the risk of critically harming the robot or its environment. Safety precautions should be implemented directly in the robot control programs in order to avoid the possibility to bypass them by other processes. Immediate reaction to unforeseen events, such as navigating around persons while following a planned trajectory in a crowded hallway, has to dominate the plan to be executed. Another example precaution is the abort of physical movements if certain force values exceed their limits. In other words, reaction overwrites plan step execution if unacceptably dangerous situations are encountered.

For the effective operation of precautions, all necessary calculations for robot behavior and perception should be carried out in near real-time. Real-time operation is of no importance if the considered action is not concerned with motion generation but with perception[2] of static objects.

Concurrent processes

Robot actions may be composed of two or more non-complex objectives that are to be satisfied simultaneously. Each goal may require a dedicated sensor-actuator loop that constantly produces commands to drive the actuator. If two or more loops operate the same actuator, the resulting commands have to be integrated into a single crisp one that can be executed by the actuator. The navigation scenario given above states an example in which two processes, i.e. follow path and avoid collisions, competitively drive the same actuator, namely the mobile platform. The integration of concurrent control loops for single robot actuators takes place at skill level. Concurrent execution of different skills is discussed in Section 4.1.2.

In addition to the concurrent control of single robot actuators in different control loops, several actuators may be controlled concurrently. For example, several joint angles have to be controlled simultaneously to achieve reasonable manipulator movements, and drive wheels can only achieve reasonable mobile motion if they operate cooperatively. These examples show that concurrency is inherent to most if not all robot control programs that are concerned with physical motion.

Self-monitoring and verifiability

(Kim et al., 2005) define verifiability of the aspired physical effect as an obligatory property of robot actions. If the completion of an action cannot be verified by the robot's sensors, there is no way to provide reliable information to a planner.

Sensing as well as acting is fundamentally error prone, thus verifiability of the outcome of an action alone is not sufficient in real-world applications. Failures potentially occur at any time and may cause exceptions that require replanning based on changed circumstances. Robot control programs must continually monitor their own execution and provide mechanisms for failure detection, analysis and possibly recovery strategies at any time during execution (Ghallab et al., 2006). In the case of failure, the control program must provide exception propagation to the overlying planning layer.

4.1.2. Robot skills as basic operators for symbolic planning

In addition to the role as a behavior producing program, as described in the previous section, a robot skill serves as the basic building block for deliberative planning. If

[2]While perception and action are often seen as antipodes, the symbolic planning layer treats deliberative invocation of a (possibly time consuming) sensor processing method just as an atomic operation that intends to change the internal belief about the world.

only the use in symbolic planning systems is considered, it would be sufficient to specify actions in a symbolic description language, such as the Planning Domain Definition Language (PDDL) (Ghallab et al., 1998). However, planning and execution cannot be seen as distinct processes. In contrast to applications of AI planning, plan-based robot control aims at generating robot behavior rather than producing just a plan (McDermott, 1992). This leads to certain properties that differentiate between atomic skills for plan-based robot control and atomic operators in AI planning domains. Important properties are described in the following subsections.

Concurrent execution and non-interference

Concurrent control of actuators on the execution level is discussed in Section 4.1.1. In some cases, concurrent execution on the task level may be useful as well. An example application is a skilled robot that starts reaching for an object on a table before the mobile approach to the table has been completed (Müller and Beetz, 2007).

If skills need to be executed simultaneously, they must not interfere with each other to preserve deterministic execution and thus to avoid uncertainty in planning. Non-interference refers to the idea that the outcome of parallel processes is not mutually affected. In the example given above, this requirement is satisfied because the involved actuators, i.e. the mobile platform and the manipulator, are independent.

However, if the relative position of the manipulator's target is computed in the beginning of the reach-action, this interferes with mobile motion. In this case, the target position for manipulation cannot precisely be determined a priori due to uncertainties in the localization of the mobile platform. Thus the two actions interfere and have to be excluded mutually. This can be done by blocking actuator devices that are not directly used on the control level or in symbolic terms in the primitive operators. However, most AI planners produce plans by means of sequences of atomic actions. Concurrent execution of skills is rarely intended.

Recoverability and consistency

After the execution of an action, the internal belief of the world should be consistent in two ways. First, the symbolic world description must not contain conflictive data. For example, a door can be either locked or open, but not both at the same time. Second, the symbolic world description should be consistent with the external physical world.

In distributed systems and data base management applications, a requirement for atomic operations is recoverability (Boudol, 1989). In this context, recoverability is the condition that an operator should either complete or it should leave the state of the system unchanged. For the symbolic representation of a skill in the form of a primitive operator, this property is given by definition.

In the physical implementation of an atomic robot skill this is hard to achieve. Uncompleted actions usually require opposed actions to restore the foregoing situation. For this reason, recoverability is not a suitable requirement for atomic skills at execution level. It is rather that an action needs the ability to report its state of execution to the symbolic layer at any time. In this way, the symbolic world representation remains up-to-date in all cases of possible failures in action execution.

Grounded action and perception

The correctness of internal world representations cannot be guaranteed due to deficiencies in sensing and perception processes. Proposed solutions to the problem of grounding symbols for actions and objects in the external world have already been discussed in Section 3.3. At this point, it should be emphasised that such problems have to be effectively solved in plan-based robot control. In applied service robotics this is not only an area of research per se, it is rather a difficulty that has to be solved practically. In contrast to AI-based planning processes with no relation to robotics, this problem should be considered in the design process of basic operators for plan-based robot control.

4.2. Complexity levels of atomic robot skills

How can robot skills be of different complexity on the one hand while being atomic on the other? The answer is simple: skills are never atomic from the perspective of execution. Every action is composed of several functions, methods and control structures. From that point of view, atomicity is reached only by single assembly instructions or isolated motor commands. However, in a two-layered system robot skills naturally implement atomicity from the perspective of symbolic planning. Robot operators are – by definition – the basic building blocks for planning. In other words, a robot skill is regarded as atomic at an abstract level, even if it has to be implemented as a complex process at a more concrete level (Boudol, 1989). Thus, the question is not whether an action is atomic, but on what level of complexity the intersection of symbolic planning and reactive control, i.e. physical execution should be.

Robot behavior is generated entirely by software. Some problems are better suited to symbolic reasoning while others prefer continuous processing. Still, in principle every problem can be solved on both layers. This leads to the design choice of where to solve particular problems. While the complexity of the overall system is not affected as illustrated in Figure 4.1, this choice influences how much computation should be done at the control level and how much should be done at the planning level.

Computation on the execution level can be more efficient due to task-specific implementation (e.g. sensor data processing, coordinate transformation, trajectory planning), in turn this may decrease the chance to be utilized by the planner in different situations. (Beaudry et al., 2005) state that the decision about which problems are to be handled using automated planning and which ones are to be managed by robot control programs are

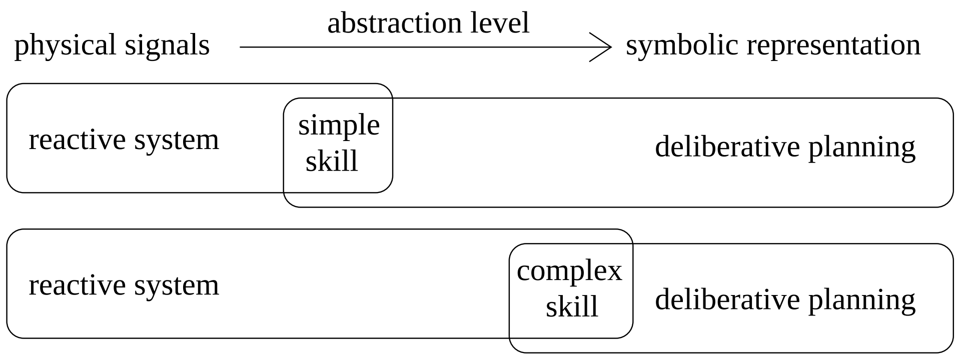


Figure 4.1.: The complexity of the overall system is independent of the abstraction level of robot skills. The simpler the skills are, the more computation is required in the deliberative planning component. The more complex a particular skill is, the more computation has to be done at reactive execution.

> ultimately a matter of design choice, depending on the capabilities of the planning system being used.

To quote (Arkin, 1998, p.207),

> the nature of the boundary between deliberation and reactive execution is not well understood at this time, leading to somewhat arbitrary architectural decisions.

An approach towards a better understanding of atomic skills is given by (Kim et al., 2005). They classified atomic robot skills into a taxonomy based on the modalities being used. Although important criteria for atomic robot actions are mentioned (e.g. verifiability, non-interpretability), they do not provide criteria for an appropriate abstraction level for actions. The authors simply write

> Each task is broken down into a set of atomic actions executed by the robot without future interpretation. They are meaningful chunks of robot algorithms that use sensor information to monitor the progress of actions [. . .]

A definition of actions using the term *meaningful* is inexplicit. Since meaning is given to algorithms by programmers and designers, this formulation supports the idea of actions being subject to design choices.

Although the question of balancing the complexity is inherent to every plan-based robot architecture, it has not been addressed individually in the literature. In this section, several criteria that serve as a general guideline for the design and implementation of atomic robot skills will be provided. These criteria are used later on to properly define skills for the service robot TASER. The implementations of these skills are described in Section 5.

4.2.1. Specific planning algorithms vs. general purpose planning

The symbolic layer specifies a general purpose planner that in principle can solve all classes of problems. However, it should be used only for problems that cannot be solved easily by specific algorithms. General purpose planning is not optimized to one domain, neither does it use algorithms that are specifically designed for certain classes of problems. Thus, a general purpose planner is rather slow.

If dedicated algorithms exist for certain classes of problems, these should be used for the sake of efficiency. The appropriate place for the implementation of such algorithms are robot control programs. Since robot control programs do not necessarily control hardware, they can also provide information services. Inverse kinematics and geometric path planning are examples of problems that should not be solved at planning level.

It is also cognitively plausible to contrast task-specific planners from general symbolic reasoning. Symbolic reasoning is comparable to conscious thinking in natural systems while a task-specific program – that just performs actions – can be compared with the unconscious execution of automated skills. If a problem, e.g. using a doorknob, has been learned extensively, it will be done without consciousness. New problems, in contrast, require active thinking if no appropriate skill can be applied. In other words: what *can* be done without reasoning, will be done without it. Similar in artificial systems: if geometric path planning and kinematic algorithms exist, there is no reason to plan solutions on the symbolic level using a general purpose planner.

4.2.2. Unambiguity of action execution

Atomic actions should be unambiguously executable and without the need for further interpretation by the executing robot control program. The prevailing context of execution may alter *how* the action is executed, but not *what* the effect of that action will be.

Usually, continuous motions are performed to achieve an action. Thus, different execution variants may lead to different physical side-effects. The action should be designed in a way that these side-effects will be of minor importance and that symbolic representations of these effects will not be needed. For example, the exact motion trajectory that is performed to reach a position with a robot arm is of no importance for goal-directed reasoning in symbolic terms. Assuming an active collision avoidance in the execution layer, there will be no need to specify a certain trajectory. Exceptions may be operations in which e.g. the upright orientation of an object under manipulation has to be assured. For such cases, it must be possible for the planner to influence the execution of actions. This can be done via parameterization of control programs.

The abstraction from details of the precise robot motion to rather coarse symbols is unavoidable and needed for efficient planning. This is exactly what makes goal-directed reasoning possible. However, this abstraction should be conservative enough to guarantee the unique interpretation of the skill in the executive layer.

4.2.3. Multi-modal integration in atomic robot skills

As described in Section 4.1.1, most robot skills that generate robot actions involve several actuators to achieve reasonable behavior. Furthermore, most robot control programs are implemented as sensor-actuator loops. Thus, robot skills naturally implement the integration of multiple modalities across sensing and actuation.

Due to the strong abstraction from concrete representations of sensor data to symbolic object descriptions, cross-modal interactions in sensor data cannot be detected at the symbol layer. If such interactions are of importance, this has to be reflected in decisions on the granularity level of atomic robot skills. Object tracking in different sensor modalities, for example, requires low-level sensor data integration within the atomic robot skill.

The above-mentioned considerations on multi-modal integration at the execution level leave no or only few options regarding the abstraction level to the developer. If multi-modal sensor data are to be integrated, this requires complex actions that capture and process these data. In cases where low-level integration is not needed, a more primitive definition of robot skills is to be preferred that uses as few modalities as possible. The integration of multiple modalities into complex robot behavior then takes place at symbol level. Using this strategy, the system can benefit from the planner's ability to adapt flexibly to sensor failure by generating plans that do not depend on the faulty sensor.

4.2.4. Reuse of atomic skills vs. optimized implementation in robot actions

Robotic research can be expected to result in de facto standards for software modules, hardware platforms, and algorithms. Face detectors, low-level vision algorithms and machine learning algorithms, for example, are already available as standard libraries. Since complex robot tasks, e.g. mobile manipulation in human environments, are system level problems, sharing components will be especially important for scientific progress, such that researchers can build on one another's contributions, compare approaches, and generate repeatable results (Kemp et al., 2007).

Atomic robot skills which are defined at a high abstraction level offer lots of possibilities for optimized implementation and efficiency. An abstract `grasp_object` skill, for example, can start reaching for the object before its exact position is recognized. The drawback of complex robot skills is that they can only be used in relatively few situations. Reaching out for an object as part of the grasp skill cannot be used in other tasks, such as pushing an object aside. Parts of the control structures and sensor-actuator loops that achieve this behavior have to be implemented repeatedly in several control programs. The more fine-grained an atomic action is, the higher the possibility for it to be used in different tasks.

There is no unique solution to the trade-off between optimized reuse of small chunks of robot control programs on the one hand, and optimized efficiency in large parts of robot control on the other. However, with a growing number of robot skills, it is often reasonable to refactor the skill library by means of decomposing complex skills into smaller ones, to allow efficient maintenance and further development.

In addition to the reuse of atomic skills in different robot tasks, it is also desirable to allow reuse among several robot platforms. While reuse of control programs is only possible in relatively few situations, sharing of primitive operators should be possible if definitions are well-conceived.

4.2.5. Summary of rationals for different abstraction levels

A conclusive answer to the question of which granularity level is to be preferred cannot be given. There are too many other factors that influence the design of a robot system, such as the capabilities of the planner, the intended overall abstraction level of the tasks to be achieved, or the possibility to use dedicated hardware to swap computational requirements. Nevertheless, arguments that constitute a guideline for designing atomic skills for hybrid deliberative reactive robot architectures can be given. The following two paragraphs list rationals that favor skills of low complexity or abstract complex skills respectively.

A. Rationals for simple robot skills

1. **Reuse:** a robot skill should be applicable in as many scenarios as possible. The simpler the skills are, the higher the chance of using them in different tasks.
2. **Generalizability:** a generic skill description on the symbol level is to be preferred to allow reuse of operators in the planning process.
3. **Unambiguity:** a skill must be unambiguous in terms of its effect and it must be executable without further interpretation. How the effect will be achieved may depend on the current external conditions.
4. **Verifiability:** the completion of an action and the state of execution should be verifiable by robot sensors at any time. The control program has to report its progress to the symbolic layer to ensure the consistency of the symbolic world representation with reality. This requirement depends on the sensory capabilities of the robot and holds for successful execution as well as for all kinds of possible failures.
5. **Planning flexibility:** in case of too complex robot skills, the planner will lose its flexibility to react to unforeseen situations and tasks. If only a couple of complex skills are stringed together, there is no use for advanced symbolic planning.

B. Rationals for complex robot skills

1. **Efficiency:** implementation of robot skills can be highly optimized. The larger the chunks of work to be done in a single skill, the larger the possibility for optimization.

2. **Hardware independence:** hardware independence can only be achieved to a certain degree. However, it becomes more difficult if skills are of low abstraction. Grasping an object, for example, can be done with different grippers by executing specific motion sequences. If an executable sequence of low-level motion commands is provided by the planning layer, it may not be applicable to other grippers. The more basic a skill is, the closer it is coupled with the robot hardware. Thus it becomes difficult to apply the skill to other robot platforms.

3. **Task-specific algorithms:** a general purpose problem solver should only be used for problems that cannot be solved efficiently by task-specific algorithms. Compared to task specific algorithms, general purpose planning is rather slow. Thus, if dedicated algorithms exist for problems, these should be used in robot controllers to alleviate the planning layer. Inverse kinematics and geometric path planning are examples of problems that should not be solved at the planning level.

The relatively few arguments for complex robot skills give the impression that a general tendency to simple robot skills is desirable. In fact, if one assumes a perfect planner that has the capability to code the robot behaviors on the fly for every robot task in every situation, native hardware commands could serve as atomic robot skills. Obviously, such a planner is not available. The capabilities of the deliberative planning component are supposably the bottleneck of the overall system, thus the rationals concerning efficiency (B.1) and task-specific algorithms (B.3) are of major importance.

Note that these arguments are guidelines rather than hard constraints. Actually it is impossible to fully satisfy all of them, e.g. verifiability 4.2.5, A.5 excludes independence of robot hardware 4.2.5, B.2. These arguments will be used throughout the next chapter to define and implement basic robot skills for the service robot TASER.

4.3. Representation of robot skills and objects

As defined in the beginning of this chapter, a robot skill consists of two parts: first, a symbolic description of the skill in the form of a primitive operator that can be used in deliberative planners, and second, a control program that integrates concrete knowledge from a world model with current sensor readings and generates executable control sequences for robot actuators. The following two sections describe how the two representational views are realized in this work.

```
(room office)
(table martins_table)  (in martins_table office)
(cup cup42)  (on cup42 martins_table)
(empty cup42)  (color cup42 green)
```

Listing 4.1: Symbolic description of exemplary objects and facts about the world. The first word in brackets denotes the name of a property or relation.

4.3.1. Symbolic representation

Representation languages have been investigated by AI researchers to describe problems as well as world knowledge. In the context of planning, they provide possibilities for the comparison and exchange of different algorithms and planning systems. Representation languages meet the demands for representing actions as well as objects for symbolic reasoning components that generate plans without consideration of details of the real world.

To support the exchange of the planning layer in robotic applications, a standardized representation language should be used. In this project, a representation similar to the problem domain definition language (PDDL) is used. PDDL (Ghallab et al., 1998) is a description language that was established for organizing the international planning contest at the annual international planning conference AIPS/ECP (now ICAPS). PDDL has made planning systems comparable in a simple and practical way, and has become a quasi standard for problem descriptions.

The description of a robot skill is functionally composed of

- a list of preconditions that have to be satisfied before the execution and
- the effect that the execution will have on the world, which is specified as a list of facts that are no longer valid and a list of facts that become true due to the action.

Robot skills may be executed in relation to objects which also have to be represented both symbolically and numerically. The symbolic description of external objects consists of one or more *logical atoms* that define the object's identity, properties, and relations to other objects.

A detailed description of the syntax being used in this work can be found in (Ilghami, 2006) and will therefore not be described here. Listing 4.1 shows an example of the symbolic definition of objects, relations and properties. An exemplary definition of a robot skill is shown in Listing 4.2. These examples should be sufficient to provide a basic understanding of how skills and objects are represented symbolically.

```
1 (:operator (!approach ?object)
2   ( (use mobile) (clear) (in robot ?room) (in ?object ?room) )
3   ( )
4   ( (near robot ?object) )
5 )
```

Listing 4.2: Symbolic description of a basic operator: The unique skill name starts with an exclamation mark by convention. Terms with preceding question mark are symbolic variables. The list in line 2 describes requirements that have to be fulfilled to permit execution of the skill. Line 3 denotes facts that will be deleted from the world description; it is empty in this example. The list in line 4 specifies facts that will be added to the world description if the skill is executed.

4.3.2. Interfacing the world

An operator describes robot behavior at an abstract level in symbolic terms, and it has to be related to the external world. The problem of how to ground internal symbols in the external world is discussed in Section 3.3. In the present work, a pragmatic approach to relate symbols to the sensory appearance of external objects and operators to the behavior producing control programs is used as follows.

Symbolic descriptions and their concrete numerical counterparts share a common data structure. Figure 4.2 schematically shows the object-oriented approach that is used. Each robot skill is encapsulated in a data structure which provides access to both the symbolic description of the primitive operator and to the control program that will execute the skill physically. Therefore, the interface `AbstractSkill` is defined that provides two methods:

`getOperator()` provides the symbolic description in the form of primitive operators that will be used for symbolic planning.

`execute(AbstractObject...)` executes the action physically by invoking the control program. Calls to native robot libraries and sensor processing algorithms are encapsulated in this method.

As shown in Figure 4.2, the concrete implementations of robot skills (i.e. implementations of the interface `AbstractSkill`) may be enriched with methods that read or modify numerical properties of the skills. If actions use or manipulate external objects, these are passed to the `execute` method as parameter.

External objects implement the `AbstractObject` interface which defines only one method:

`getSymbolicDescription()` provides the symbolic description of the object, its properties, and its relations in the world represented as a list of literals that can be interpreted by the symbolic planner.

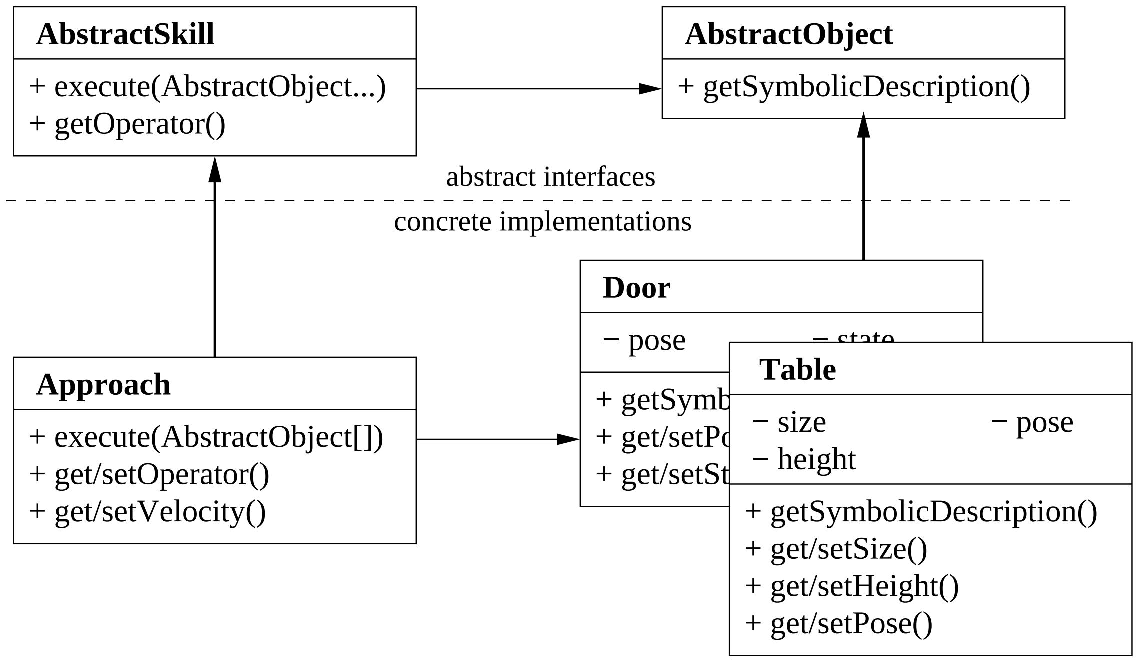


Figure 4.2.: Symbolic and numerical representation of skills and objects is done via shared data structures. Concrete implementations of objects include additional numerical data that are needed and modified by the robot control programs. The term *abstract* denotes the object-oriented conceptualization of an interface.

The concrete implementation of a physical object may provide several properties that can be measured by robot sensors or manipulated by robot actuators. These properties are represented in numerical terms. They are used by control programs if robot skills are executed with the particular object.

In this work, investigations with focus on perception are mainly restricted to static objects. Static objects, i.e. objects that do not move autonomously, do not require elaborate anchoring techniques. If anchoring or ambient perception processes, e.g. robot self-localization or people tracking, are needed, they need to be implemented as isolated processes in addition to the proposed robot architecture. The autonomous acquisition of symbol-object and symbol-action relations is not part of the investigations described here.

5 A Library of Multi-modal Skills for a Mobile Manipulator

Logic will get you from A to B.
Imagination will take you everywhere,
Albert Einstein, 1879-1955

The notion of atomic robot skills is defined and discussed in the previous chapter. The discussion is based on the assumption that a layered architecture can be used for plan-based robot control. Alternative architectures without a layered structure are reviewed in Chapter 3. Fundamental to all layered architectures is a library of atomic robot skills. Regardless of the type of the planner that is used, the atomic elements of plans have to be physically grounded. Perception has to process sensor data and actions have to be executed with robot actuators.

Assuming that today's robot learning methods are not sufficiently elaborated for learning robot skills from scratch, at least an initial set of skills has to be handcrafted. Since no library is available for robot actions that can be used with a variety of robot platforms, developers have to implement basic hardware controllers, basic robot skills and complex control processes each time a new platform is set up.

This chapter describes a concrete practical implementation of a library of robot skills which will be used by the multi-modal service robot TASER. Many of the presented robot skills have been refined through experiments, and numerous refactorings have been necessary due to extensions of the robot's hardware and software architecture. This process led to general recommendations for the design of atomic robot skills for plan-based robot control which are described in Section 4.2.5. Whenever the definition and implementation of a robot skill conforms to rationals from Section 4.2.5, this will be pointed out in the skill description. The control structure that executes the atomic robot skills in a goal-directed manner will be described in the Chapter 6. Possible exceptions and the handling of these will be discussed.

In addition to the practical focus of this chapter, the question of how atomic robot actions for plan-based robot control can gain from the combination of different modalities will be answered. While it is well accepted that robot skills with focus on actions require sensors to guide, monitor and validate their behavior, it is usually assumed that perception proceeds without physical motion of the robot. Exceptional applications in which active processes are used in the service of perception are reviewed in Section 3.4.1. As already proposed in (Weser et al., 2008), proactive perception in general can greatly improve robot perception. Supporting this idea, seven out of ten implemented robot skills with the purpose of perception require robot manipulators to set the perceptual focus. It will be shown that proactive perception greatly simplifies perception algorithms and allows for the application of algorithms which would otherwise be intractable.

Section 5.1 describes objects that can be perceived and manipulated by the implemented robot skills. The description of perception processes, realized as robot skills, is described in Section 5.2 and the manipulation of objects is described in Section 5.3. In the same section, other skills with the purpose of acting physically are described, too. Since action is involved in most perceptual skills and perception is in turn required by all actuator skills, the distinction between perception and actuation skills might seem inappropriate. However, the underlying criterion for this classification is not whether the robot acts but what the purpose of the skill is. In perception skills, movements are serve the purpose of better perceiving the world, while perception in executive skills adopts mainly the role of monitoring and guiding robot motions. Both sections on robot skills provide definitions of primitive operators that can be used in symbolic planning processes.

5.1. External objects

A number of objects have been identified that can be detected and manipulated by current mobile manipulators. The choice of objects is based on the perceptual capabilities of the particular robot being used on the one hand and on the requirements that arise from the test challenges given in the introductory chapter on the other. The internal representation of objects consists of symbolic descriptions for relations to other objects as well as of physical properties that can be detected by the sensors of the used robot platform. Since this work focuses on the integration of sensors, actuators, robot control and robot planning, the individual algorithms for perception of particular objects in single sensor modalities is only of minor interest. The combination of several perception algorithms focusing on a common interest and possibly supported by robot actions, however, provides a reliable method for object detection and examination and has not been explored in such detail to date.

Table 5.1 lists the objects that have been defined to allow the demonstration of plan-based robot control within the scenarios given in Chapter 1. The table lists descriptions of symbolically represented relations as well as physical properties which are attached to allow detection and recognition of the objects.

object	symbolic properties	physical properties
room	• is the robot in there? • connected doors	• geometric dimensions
door	• open/closed • locked/unlocked • connected rooms	• geometry • position (world coordinates) • opening angle
cup	• location (table, hand) • color (red, blue, etc.)	• relative position to robot • size (grasp relevant parameter) • visual appearance (template)
bucket	• location (room, robot hand)	• position (world coordinates) • size (grasp relevant parameter)
table	• location (room) • empty surface	• position (world coordinates) • geometry

Table 5.1.: Implemented objects: relations are represented symbolically and physical properties are represented numerically to be used by TASER's perception and manipulation skills.

5.2. Multi-modal perceptual skills

Table 5.2 lists implemented robot skills with the focus on perception. The selection is based on what is needed to realize the test scenarios given in Chapter 1. A fairly high abstraction level was chosen to allow proactive perception at skill level. At the same time, the skills are simple enough to ensure flexible use in the planning layer to be used in as many tasks as possible. A closer look at Table 5.2 shows that almost all objects that are listed in Table 5.1 recur. The only exception is the object `room`. This is not needed, since the symbol layer keeps track of the current room of the robot and the self-localization in the physical layer is always active. Locomotion skills, as will be described in Section 5.3, check for inconsistencies in the symbolic representation of the robot's location. An exception from "usual" perception skills is the `calibrate camera` skill since it does not provide information on external objects. However, it is another example of how robot behavior can gain from integration of perception and action if combined properly. The skill calibrates image coordinates to the external world by combining the PTU, the robot arm, force sensors on the robot hand and knowledge about the table which is under examination.

Only one perception skill, i.e. `find door`, integrates multiple sensors on skill level. Multi-sensor integration will mostly take place on symbol level, which will be shown in Chapter 6. Nevertheless all skills are inherently multi-modal across sensor and actuator modalities. The current belief about the world determines the attentional focus on which sensors will be focused actively to obtain the most valuable sensor data. The information on actuation, sensor data and previous knowledge is then integrated and stored in

	laser	camera	force	mobile	arm	hand	ptu
find door	×	×					
localize door	×			×			
check door latch			×	×	×	×	
localize cup		×					×
find table	×						
confirm table		×		×			×
touch table			×	×	×	×	
localize bucket	×			×			
calibrate camera		×	×		×	×	×

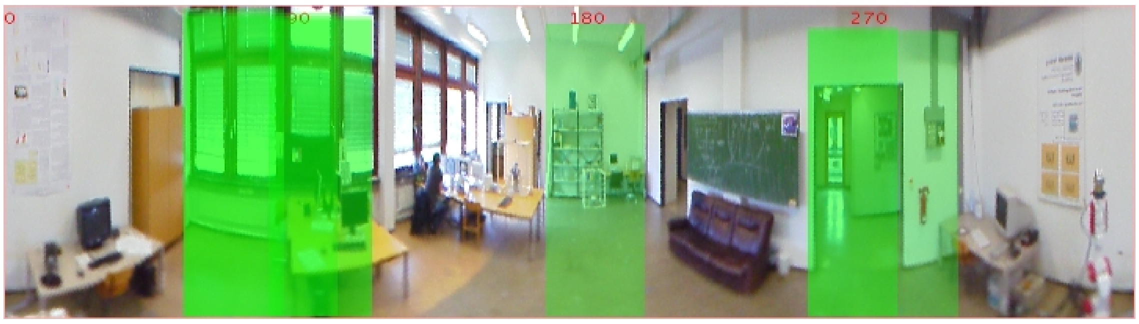
Table 5.2.: Robot skills with focus on perception: most of them require robot actuators to focus sensors on interesting parts of the world.

numerical representations that are attached to objects as described in Table 5.1. The symbolic representation of the world is modified directly by robot control programs that execute the skills. This constitutes a close coupling of the symbolic and physical layer.

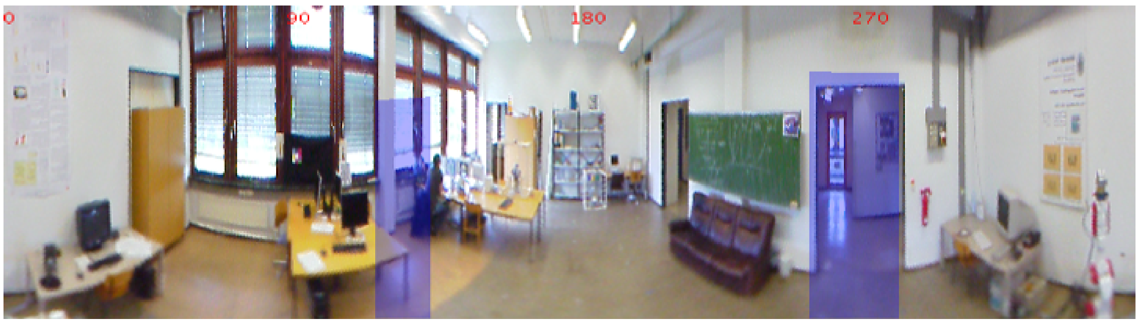
5.2.1. Find door

The `find_door` skill is the only perception skill that performs low-level multi-sensor fusion. The initial recognition of doors requires information provided by non-local sensors, i.e. sensors that are able to measure across long distances. Assumptions on the visual nature of the door can be reduced to a minimum if laser range data are combined with image data: a door has to have vertical edges on both sides. This can be detected in omni-directional images in real-time. Figure 5.1(a) shows a typical scene with possible door candidates perceived using this assumption. Due to the generality of the assumption, the door candidates clearly overestimate real occurrences of doors. The assumptions on the appearance of doors in laser range scans can also be reduced to a minimum due to the complementary properties of both modalities. The simple assumption that a door appears as a gap in a wall in laser range data is sufficient to complement the results from the image algorithm. Figure 5.1(b) shows the door candidates resulting from laser range data. The combined multi-modal result (Figure 5.1(c)) is the intersection of the sets of door candidates resulting from both uni-modal perceptions. Both unimodal perception algorithms are applicable regardless of the state of the door, e.g. open, closed, in between.

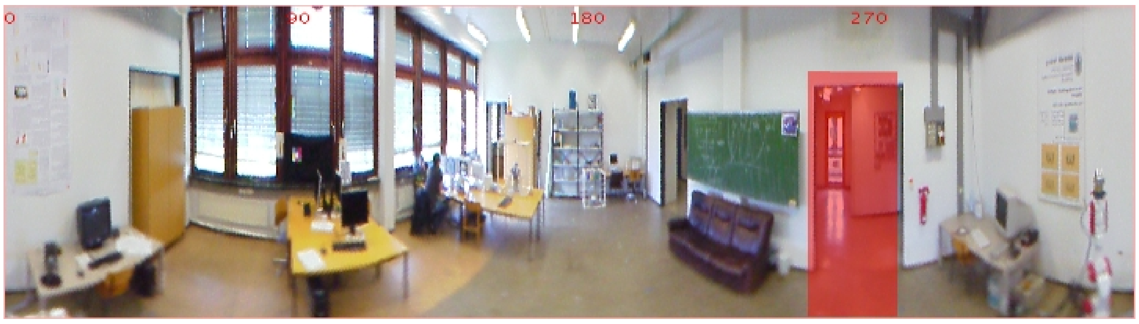
The primitive operator for this skill is shown in Listing 5.1. The only preconditions are the availability of the used sensors. Since the outcome of the perception is not clear at planning time, the operator has no effect on the symbolic world representation. If the solution of a problem depends on this perception, the planner will replan after the execution of this perception. The interaction of planning, perception and replanning



(a) door candidates detected in omni-directional images



(b) door candidates detected in laser range data, the results are transformed to image coordinates for better visualization



(c) integrated multi-modal result

Figure 5.1.: Multi-sensor integration for door detection on the skill level.

```
(:operator (!find_door)          ; name of the operator
  ( (use laser) (use omni) )     ; preconditions
  ( )  ( )                       ; effect in symbolic terms
)
```

Listing 5.1: The operator definition for `find_door` does not change the state of the world in symbolic terms. Only the control program that physically executes the skill will change the state of the world. If a problem is dependent on the results of this perception routine, replanning will be required after the execution of the skill.

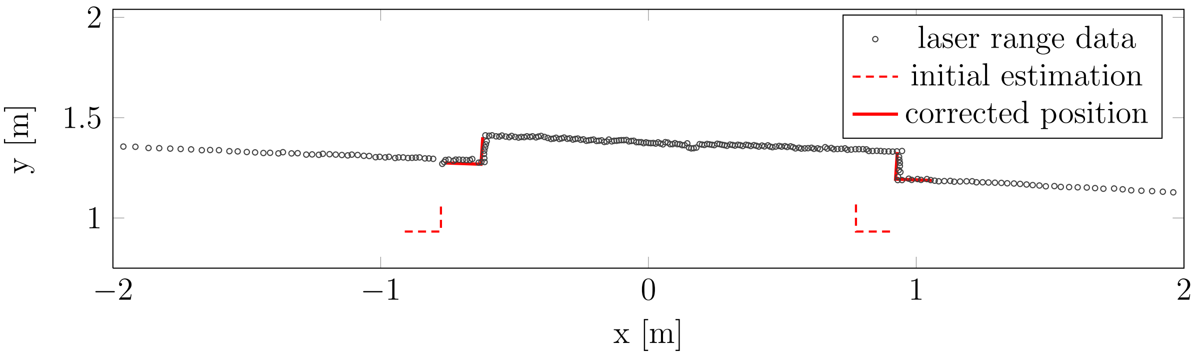


Figure 5.2.: Perceiving the exact door position: the initial estimation (dashed red) is matched to laser range data (gray circles). The shape of the door is assumed to be known.

will be described in detail in Chapter 6.

5.2.2. Localize door

The `localize` door skill determines the exact position of the door relative to the robot. The skill requires a previous estimation of the door position and a model of the shape of the door. An iterative algorithm that is similar to the expectation maximation algorithm by (Dempster et al., 1977) has been developed to match an arbitrary shape – in this case the door shape – to a set of data points in Cartesian coordinates. The algorithm is described in detail in Appendix A. Figure 5.2 shows the initial assumption of the location of a door and the matched result.

The purpose of this skill is to provide the position of the doorknob for opening the door. Since the doors in the experimental environment all show a standardized format, the 3D position of the doorknob can be calculated directly from the 2D position of the door. However, other techniques to perceive the doorknob directly with the robot's sensors, e.g. stereo vision, can easily be integrated due to the layered architecture and a clear modular

```
; may add (nosuccess localize ?sensor ?object)
; and cause replanning
(:operator (!localize ?sensor ?object)
  ( (use ?sensor) (not (nosuccess localize ?sensor ?object)) )
  ( )
  ( (poseknown ?object) )
)
```

Listing 5.2: The operator definition to localize objects adds the literal `(poseknown ...)` to the world state by default. If the perception fails, an exception will cause the planner to search for an alternative plan.

structure of the software environment (see Section 2.3). The self-localization of the robot is not reliable enough to store the exact door position in Cartesian world coordinates. Every time the robot changes its own position, previously perceived positions of external objects are made invalid. Thus the robot needs to approach the door before the exact door position can be perceived. A separate skill is implemented for approaching objects to allow reuse on the planning level and to avoid code duplication on the control level (4.2.5, A.1).

The primitive operator for the `localize` skill is shown in Listing 5.2. It is defined in a generic way to allow flexible use in planning processes (4.2.5, A.2). The operator assumes that the object will be localized successfully and thus adds the `(poseknown ?object)` literal to the world state. In the case of unsuccessful perception, a `(nosuccess ...)` marker is added to the world state and an exception will cause the planner to search for an alternative plan. This interaction of planning and execution constitutes a general pattern for perception with predictable results and will be detailed in Section 6.3.4.

5.2.3. Check door latch

This skill is used before the robot will open a door, to decide whether the robot needs to use the doorknob before it pushes the door open. To check whether the door latch is in the notch, the robot has to stand near the door. The robot can then determine the state of the door by pushing it and monitoring the force values at the robot hand. The door is assumed to be unlatched if the force stays below a threshold for a distance that is larger than the flexibility which is inherent to the door and the robot system. Figure 5.3 shows typical force patterns for closed and open doors. The oscillation of the force values while the robot pushes a door (most noticeable in the blue plot) results from the swing of the door in response to the absorbed kinetic energy. The amplitude of the force values decreases since the kinetic energy of the door is increased with every pushing contact.

If the literal `(latched ?door)` does not exist in the symbolic world description, the

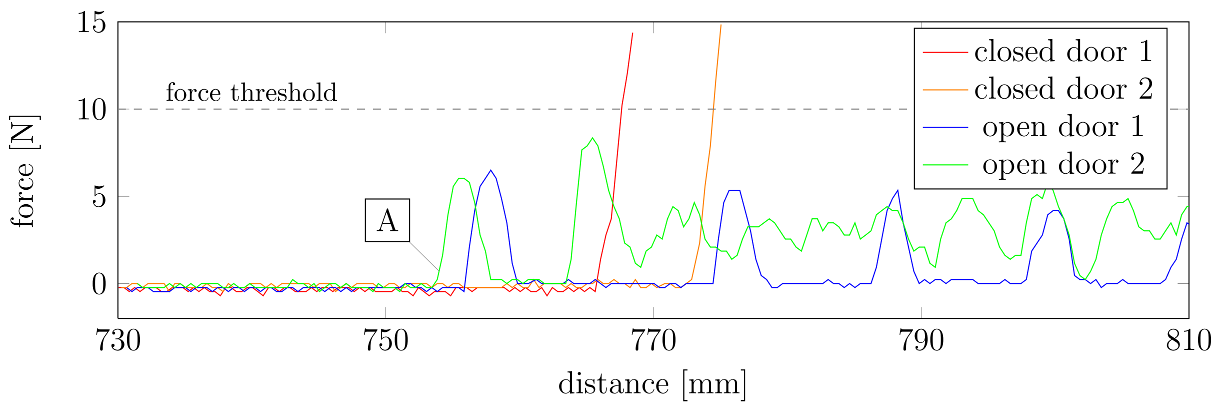


Figure 5.3.: Four trials to push a door open: the force values indicate success or failure of the action. If the force value quickly increases beyond a threshold (10 N are used in the experiments), the robot stops the motion and assumes that the door is closed. If the force value stays below the threshold for several centimeters after the first door contact (e.g. A for the green plot), the door is assumed to be open.

```
; may add (latched ?door) and cause replanning
( :operator (!check_latch ?door)
   ( (use arm) (use hand) (use force) (near robot ?door) )
   ( ) ( )
)
```

Listing 5.3: The symbolic description of `(check_latch ?door)` assumes that the door is unlatched. If on the contrary it is latched, an exception will invoke the planner to replan.

planner assumes that the symbol which is assigned to `?door` is unlatched. This is also the default assumption for the primitive operator which describes this skill. Thus the operator (Listing 5.3) has no effect. If the door is latched, the planner will replan based on the new situation.

5.2.4. Localize cup

This skill is a prerequisite for grasping of objects. The symbol `cup` is synonymous with graspable objects that usually lie on tables and can be detected visually. The skill is not intended to initially find an object. Instead it requires previous assumptions on the location of the object, i.e. on which table it is, and estimates the exact position relative to the robot. In this, this skill is similar to the skill that localizes doors (Section 5.2.2).

For the execution of this skill it is assumed that the robot stands near the table on which

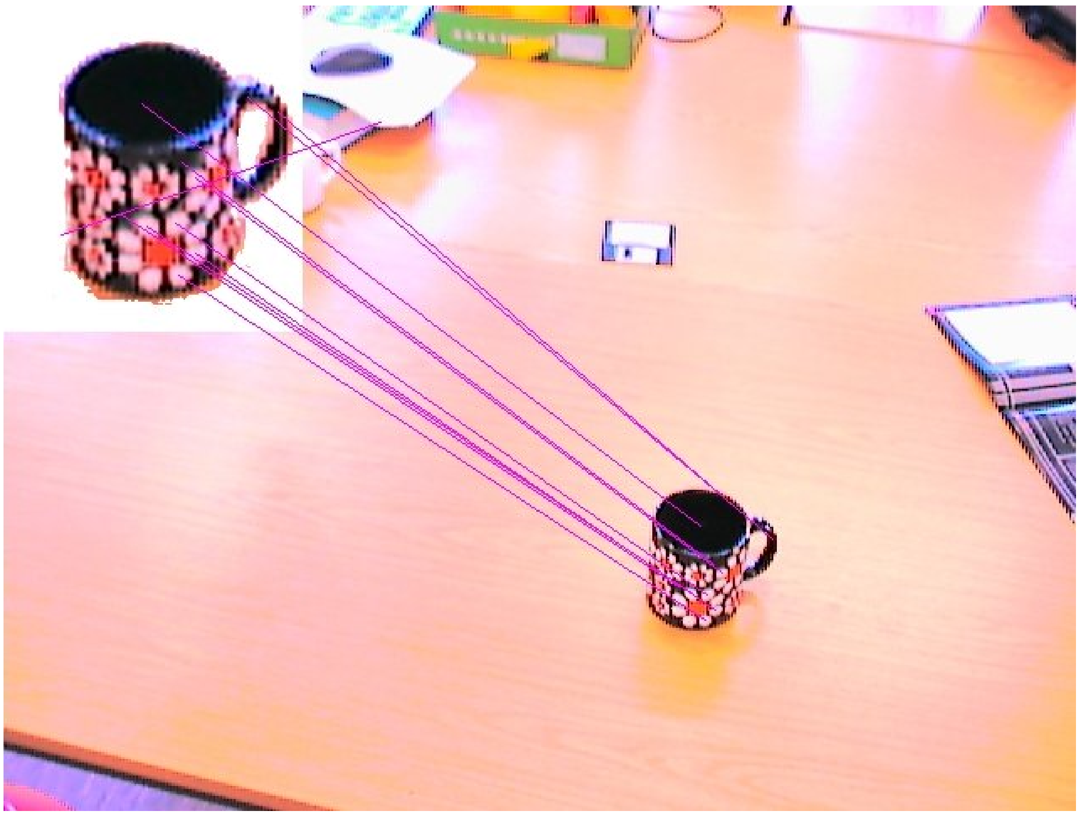


Figure 5.4.: Objects on the table are detected using SIFT features obtained from object templates.

the object should be located. By means of the pan-tilt unit, the camera is focused on the table scene to obtain the best possible viewing direction. Currently, the intended object can be detected in image coordinates using SIFT features (Lowe, 1999, 2004) from object templates. The template is part of the object representation as described in Section 5.1. Figure 5.4 shows a detected cup with the corresponding features in the template and in the obtained image. Autonomous detection of unknown objects is currently not possible. A human instructor will be asked to draw a box around the object of interest in a graphical user interface, if needed. The position of the object in image coordinates is transformed into robot centered Cartesian coordinates using the transformation matrix which is described later in Section 5.2.9.

The symbolic description of this skill is identical with the one shown in Listing 5.2. At planning level, this allows for reuse of the operator (4.2.5, A.2). Unambiguity at execution level (4.2.5, A.3) is given since the operator will be instantiated with an object and a sensor. The type of the object (a literal `(door ?object)` or `(cup ?object)` etc. must be available) and the sensor to be used indicate the type of perception that will be performed.

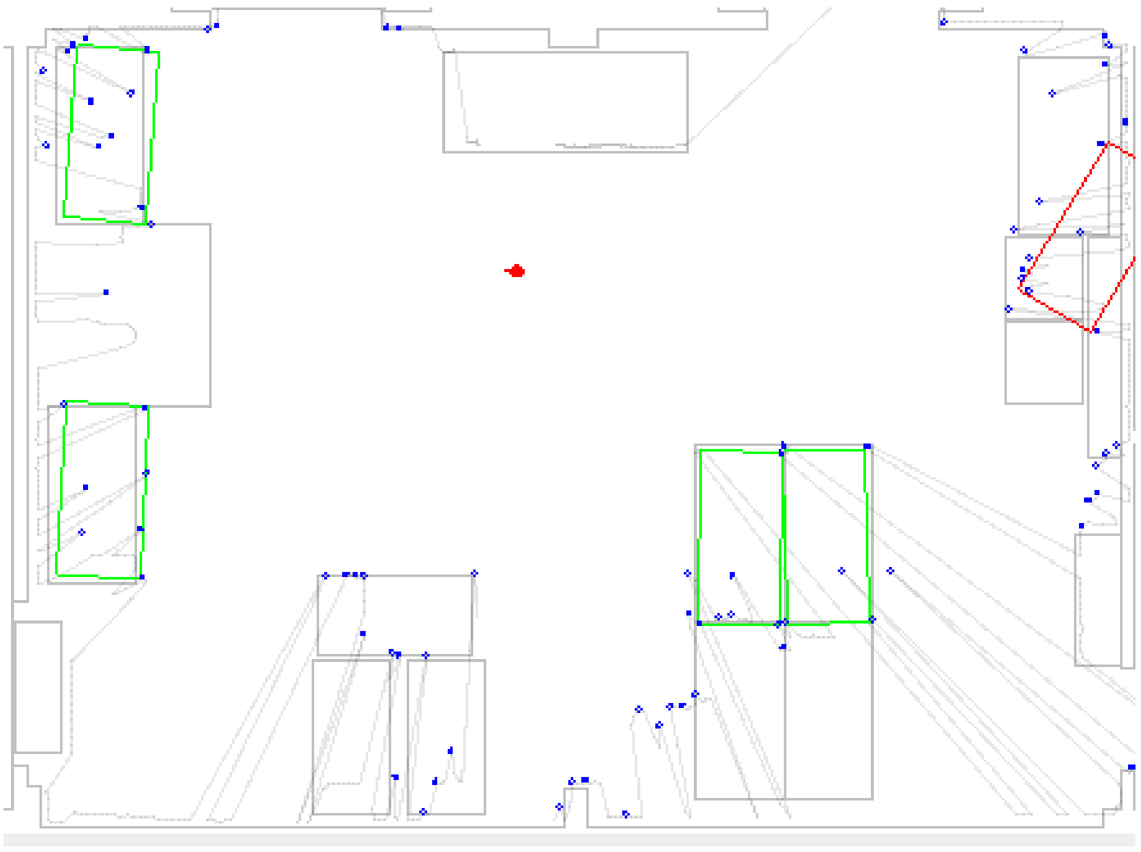


Figure 5.5.: Example of four correct (green) and one false (red) detected table. The blue dots denote possible candidates for table legs. This is the result of the unimodal feature detector using laser range scans.

5.2.5. Find table

Finding a table in one sensor modality alone is cumbersome and error-prone. Algorithms using different sensor modalities are integrated on the symbol level to transfer initial vague assumptions into reliable internal representations of real tables. Thus, the initial detection of tables, which is realized in the skill described here, does not have to be reliable. The detected table candidates will be confirmed or dismissed using the `confirm_table` skill which is described in the following section.

The initial detection of table candidates takes place in laser range data. A valid pattern for a table in laser range data consists of at least three table leg candidates in an orthogonal arrangement with a certain ratio of width and length. The size of the table is assumed to be known a priori and is represented along with the symbolic table description as described in Section 5.1. A table leg candidate appears as local minimum in the sequence of laser range measurements. The number of detected tables varies depending on the tolerance of the difference of the real table size and the size of the detected table candidate. As shown in Figure 5.5, the tolerance is chosen so that the algorithm overestimates the occurrences of real tables, which may result in false table candidates. Nonetheless due to clutter, occlusion and insufficient sensor resolution, it cannot be guaranteed that all real tables will be identified.

The symbolic representation of the `find_table` skill is similar to the one used for the `find_door` skill (see Section 5.2.1), except for the name and that the laser range scanner is the only required sensor modality. Robot tasks whose solution depends on autonomously detected table candidates require replanning after execution of this perception skill.

5.2.6. Confirm table

Visual detection of complex objects, e.g. tables in this case, in a real cluttered environment is a challenging task and requires large data sets to learn appropriate visual features. Visual algorithms can be greatly simplified if previous knowledge will be exploited.

The proposed approach here is to focus the camera actively on a part of an assumed table candidate. Using this strategy, the search space for the image processing algorithms can be reduced. A previously known geometry model of considered tables provides strong assumptions on the visual appearance in the obtained image. In the current implementation, it is assumed that the robot stands at a well-defined position relative to the table. The camera is then focused actively to a corner of the estimated table. In this way, the detection of a noticeable near-vertical edge in the center of the image is sufficient to tell correct tables from false alarms. Figure 5.6 shows three correct tables and one false alarm.

The primitive operator to confirm a table, shown in Listing 5.4, is defined in a generic way, similar to the `localize` operator (Listing 5.2). It can be used for different objects and sensors on the symbol level (4.2.5, A.2) and can uniquely be assigned to certain control programs due to instantiation of the symbolic variables.

5.2.7. Touch table

If a definite guarantee of correct detection is required, the robot can touch the table top to confirm a surface using haptics as the third sensor modality. While the visual confirmation was able to dismiss wrong table candidates reliably in isolated experiments, this skill provides an excellent testbed to demonstrate proactive perception in integrated experiments. Furthermore, this skill provides information on the height of the inspected table. Chapter 7 describes the integrated experiments that make use of all perception skills focusing on tables.

The skill described here does not demand a separate operator definition. The `confirm` operator which is described in the previous section will be used with force sensors as sensor modality. Execution of this skill requires the robot to stand nearby the table, which is guaranteed at the planning layer as will be described in Chapter 6.

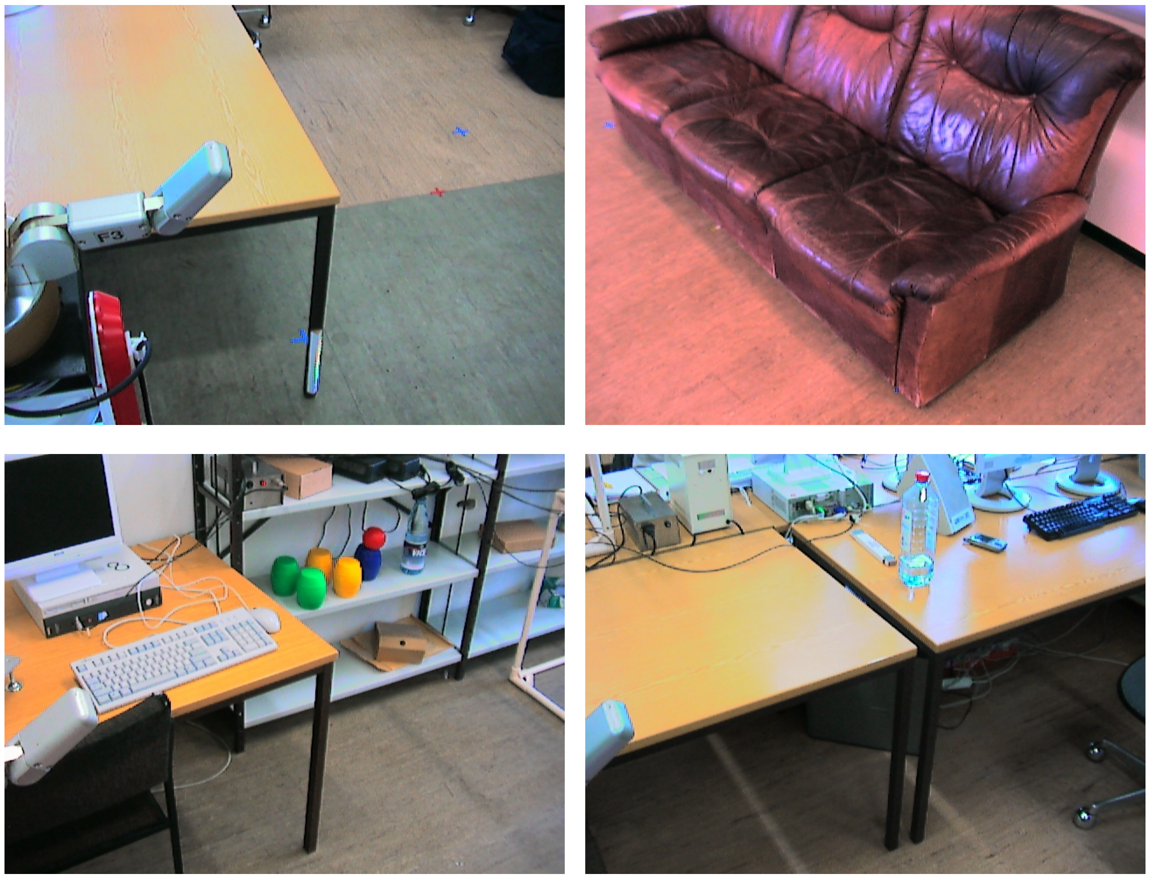


Figure 5.6.: Visual verification of supposed table candidates: the upper right table candidate has been identified as false alarm.

```
; may add (nosuccess confirm ?sensor ?object ?actuator)
; and cause replanning
(:operator (!confirm ?sensor ?object ?actuator)
  ( (use ?sensor) (use ?actuator)
    (not (nosuccess confirm ?sensor ?object ?actuator))
  )
  ( )
  ( (confirmed ?object) )
)
```

Listing 5.4: The operator definition to confirm objects adds the literal `(confirmed ...)` to the world state by default. If the perception fails, an exception will cause the planner to generate an alternative plan.

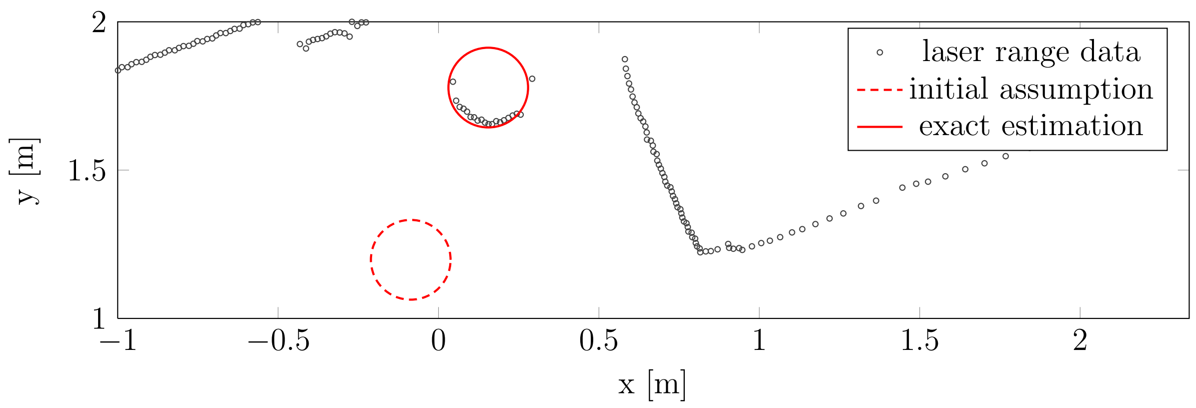


Figure 5.7.: Initial assumption of the bucket's pose (dashed red) and estimated exact position (solid red) after applying the find bucket skill.

5.2.8. Localize bucket

Similar to the localization of a door or an object on a table, this skill requires previous assumptions on the location of the bucket. On the symbol level, it uses the same representation which is the operator shown in Listing 5.2.

The exact position of the bucket is determined using laser range data and the shape-matching algorithm which initially was developed to find the position of a door in laser range scans. See Appendix A for details of the algorithm. Figure 5.7 shows an initial assumption of the bucket's location and the exact estimation of the position after applying the algorithm.

5.2.9. Calibrate camera to table top

If the robot is supposed to grasp an object from the table top, it has to determine the 3D position of the target object. There are several ways to do so, e.g. stereo image analysis. However, if the height of the table and the camera parameters are known, it is possible to compute the 3D coordinates directly from the object's position in image coordinates.

In the presented work, a 3x3 affine transformation matrix is used to transform image coordinates into a horizontal plane in robot-centered coordinates at the same height as the table top. Although precise camera parameters are not taken into account, this is sufficient to compute 3D coordinates of objects placed on the surface with less then 3 cm error. This is sufficient to reliably perform grasps of cups, bottles etc.

A transformation matrix is used according to the following equation.

$$\begin{pmatrix} x_i & y_i & 1 \end{pmatrix} \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x'_w & y'_w & z'_w \end{pmatrix} \tag{5.1}$$

Image coordinates x_i and y_i are assumed to be known. The world coordinates x_w and y_w are computed with $x_w = x'_w/z'_w$ and $y_w = y'_w/z'_w$ respectively.

To determine the correct matrix M, it is sufficient to find 3 pairs of corresponding coordinates in both frames. Each pair of coordinates $(\ x_i \quad y_i\ )$ and $(\ x_w \quad y_w\ )$ provides the following two equations.

$$
\begin{aligned}
x' &= \frac{x_i m_{11} + y_i m_{21} + 0}{x_i m_{13} + y_i m_{23} + 1} \\
y' &= \frac{x_i m_{12} + y_i m_{22} + 0}{x_i m_{13} + y_i m_{23} + 1}
\end{aligned}
\tag{5.2}
$$

Thus, three pairs of corresponding points lead to six equations which are sufficient for finding a unique solution for the six unknown variables of the transformation matrix.

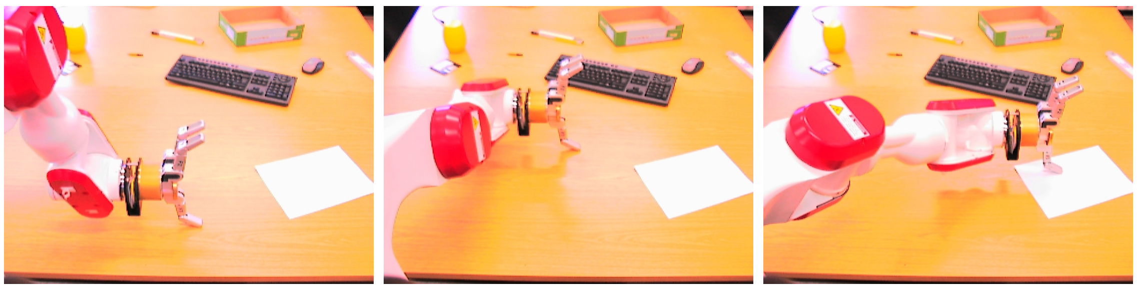
The corresponding points for calibration can be found if the robot touches the surface of the table by using force-controlled motion. The touching point in world coordinates can easily be computed via direct kinematics. The image coordinates of the fingertip can be determined using the template-based algorithm which has been described in Section 5.2.4 due to the well-known visual appearance of the robot hand. Figure 5.8(a) shows the robot touching the table top. An exemplary table scene captured with the robot's camera and transformed to robot-centered world coordinates is shown in Figure 5.8(b). A transformation matrix has to be determined for each combination of table height and position of the camera, i.e. the joint angles of the pan-tilt unit (PTU) on which the camera is mounted on.

5.3. Multi-modal executive skills

As already stated in the previous section, the distinction of skills into executive and perceptual is not exclusive. While the main purpose of executive skills is the generation of physical movements, all of them use sensors to guide the actions. Executive robot skills are thus inherently multi-modal across sensing and perception. Moreover, most of them also use several actuators. Table 5.3 lists implemented skills and the sensor and actuator modalities that are involved in execution. This set of executive skills provide all necessary means to implement the target scenarios given in the introduction. A key factor for the choice of the abstraction level of executive skills is the desire for the integration of task-specific algorithms (4.2.5, B.3). Whenever an elaborated software library is available for certain robot applications, it will be used in the control program. The following paragraphs describe executive skills that demonstrate multi-modal integration at the sensor-actor level.

5.3.1. Approach and leave object with the mobile platform

Most plans for manipulation tasks start with the approach to an object of interest. To do so, the robot analyzes the object and its environment to find a suitable approach


(a) The robot touches the table top to self-calibrate its camera.

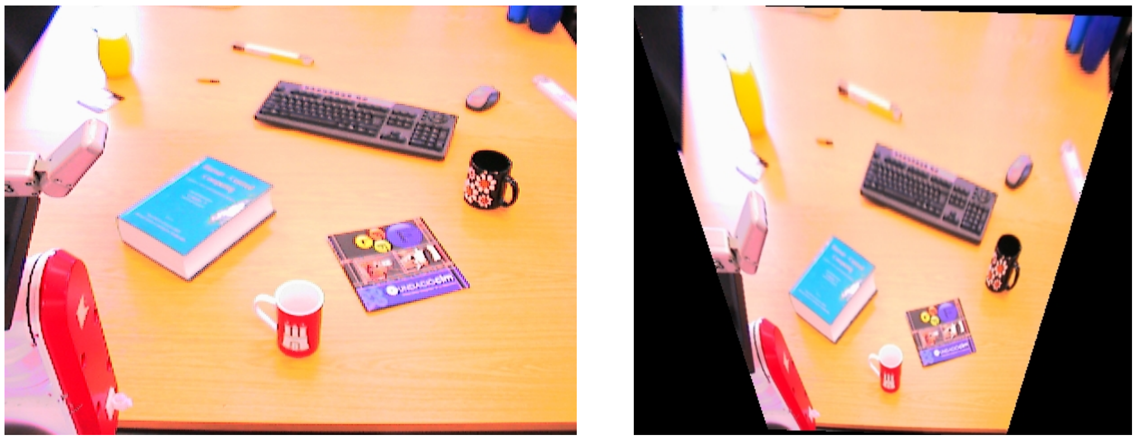

(b) Table scene in image coordinates and transferred to robot-centered cartesian coordinates.

Figure 5.8.: After determining the transformation matrix (Equation 5.1) by touching the table top, an arbitrary table scene can be transferred to robot-centered world coordinates.

	laser	camera	force	mobile	arm	hand	ptu
approach object/planned locomotion	×			×			
leave object	×			×			
reach for object			×		×	×	
move to transport position			×		×	×	
grasp object			×		×	×	×
open doorknob			×		×	×	
cross door	×			×			

Table 5.3.: Robot skills with focus on action execution. All of them require sensor modalities to monitor or guide the action execution.

trajectory. The approach trajectory is intended to prepare the last meter to the object to avoid unwanted mobile behavior that results from the geometric path planner's ignorance of the object's properties and other circumstances. A table, for example, can be approached best in an orthogonal direction to one of its long sides while the direction of the approach to a bucket is of no importance. Other circumstances that restrict the robot's mobility near objects may be a large object carried by the robot which extends the robot's overall scale.

To reach the starting point of the approach trajectory, the robot uses the service of a geometric path planner. The planner requires an edge-map of the environment and assumes that the position of the robot is known. It is composed of three steps:

1. extend edges from the map according to the size of the robot. Thus, the robot can be approximated as a single point.
2. generate visibility graph of the resulting obstacles
3. perform A* search on the visibility graph from the robot's current position to the goal position

Figure 5.9 shows a planned trajectory in a building of the University of Hamburg.

If the robot should leave an object that has been approached before, it moves backwards for a certain distance. This is necessary since turning the mobile platform requires larger safety distances to objects than straight mobile motion. If the area behind the robot is blocked, a human operator is needed. However, this hardly ever happens since the robot has approached the object on a similar trajectory.

Symbolic definitions for the two skills are shown in Listing 5.5. The preconditions for an approach skill state that the object must be in the same room. Changing rooms is treated as an individual skill due to narrow doors that may require particular caution. Another precondition is that the robot is not allowed to approach an object if it is currently near another object. In this case, the robot has to leave the other object first to avoid possible collisions while turning away from it. Otherwise the collision detection of the geometric path planner used by the `apprach` skill would stop the mobile motion and cause replanning.

5.3.2. Reach for an object with the robot arm

Similar to the skill described previously, the reach skill is a prerequisite for most manipulation tasks. Before the robot can manipulate an object (e.g. grasp cup, open doorknob, push light switch), the manipulator has to reach for it.

This robot skill requires the exact position of the target object relative to the robot to be known. The robot arm trajectory is computed by a specific purpose planner taken from the Robot Control C Library (RCCL) by (Lloyd and Hayward, 1992). Unless no other requirements enforce Cartesian motion, the trajectory from the current position to the goal position is interpolated in joint space. Before the generated trajectory is executed,

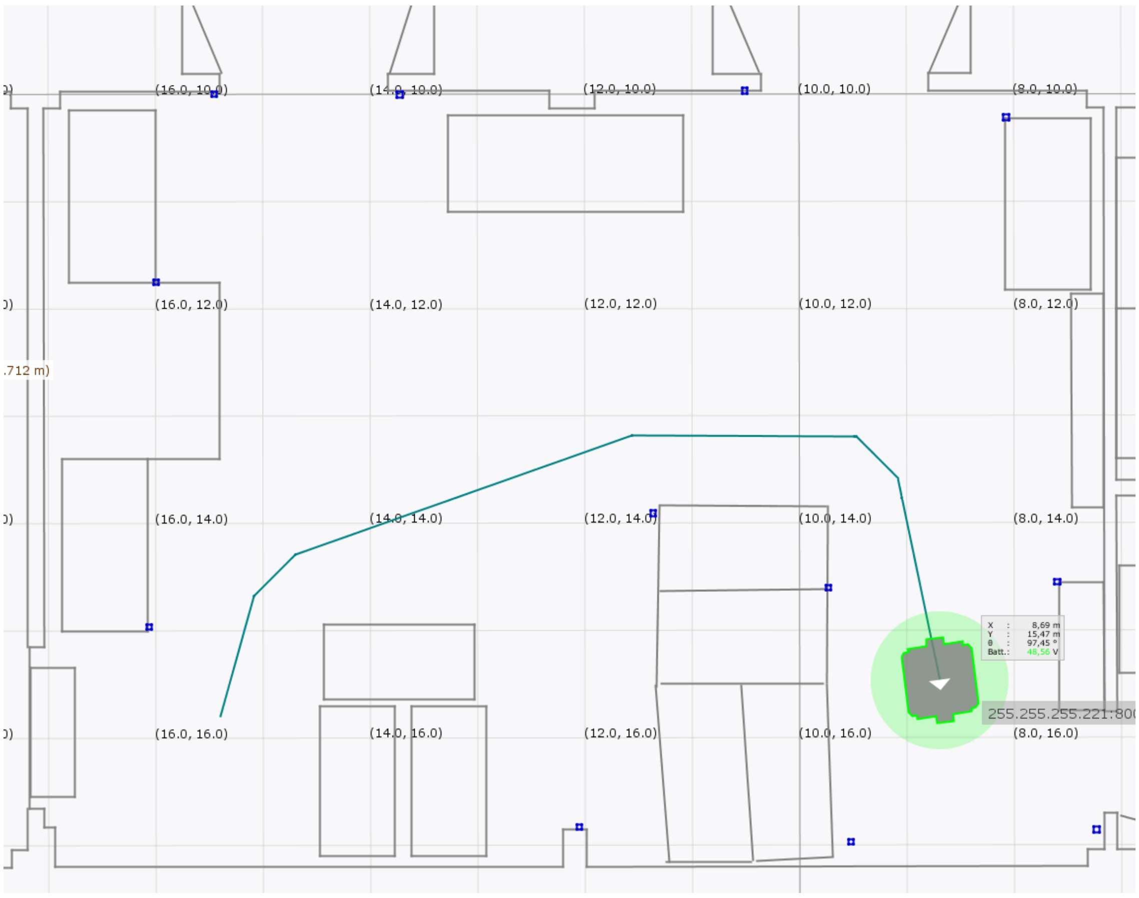


Figure 5.9.: Paths for the mobile platform are generated by a geometric path planner based on A* graph search. The planner has been implemented in this project and is used at skill level.

```
(:operator (!approach ?object)
        ( (use mobile) (use laser)
          (not (nearto ?other)) (not (same ?other ?object))
          (approachable ?object) (in ?object ?room)
          (in robot ?room) )
        ()
        ( (near robot ?object) )
```

```

(:operator (!leave ?object)
        ( (use mobile) (use laser)
          (approachable ?object) (near robot ?object) )
        ( (near robot ?object) )
```

Listing 5.5: Symbolic definition of the approach skill and the leave skill. Note the two preconditions for approaching an object: the robot has to be clear of other objects and it has to be in the same room as the target.

it will be checked for singularities in continuous motion and for collisions with the robot's body or the target object. If such problems are encountered, additional waypoints that are known to be retractable are included. For safety reasons, force values from the robot hand are monitored during the execution of the movement. If an unexpected force is applied (actually there is no force expected within a reach action), the motion will stop immediately.

Listing 5.6 shows the operator for this skill. It needs two alternative definitions depending on whether the object is on the floor or on another surface, e.g. a table. In the latter case, the precondition specifies that the robot has to be near the surface object instead of being near the object to be reached for.

5.3.3. Grasp object

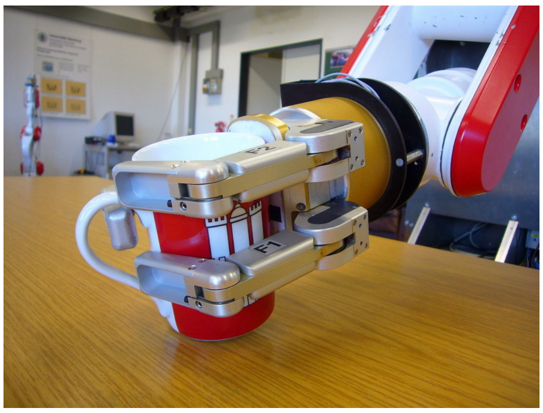
In this work, a simple force-controlled power grasp is implemented in which the robot closes its hand until a predefined force threshold is reached. The position of the fingers after the grasp yields information about the success of the action. If the hand is completely closed after the grasp, it can be assumed that the object has slipped away. Figure 5.10 shows the robot grasping different objects.

More elaborate grasping methods have been developed by researchers that specifically investigate this topic. The work of (Baier-Löwenstein, 2008), for example, presents a method based on reinforcement learning which is able to learn grasps for arbitrary objects with an arbitrary number of contact points. Since the project which is presented

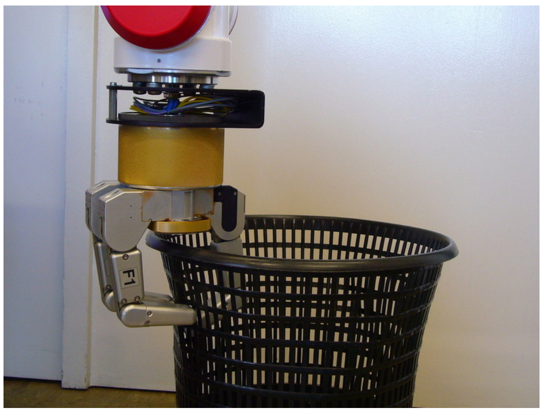
```
(:operator (!reach ?object)
   ( (use arm) (use force)
     (on ?object floor) (near robot ?object)
     (park manipulator) )
   ( (park manipulator) )
   ( (near manipulator ?object) )
)

(:operator (!reach ?object)
   ( (use arm) (use force)
     (on ?object ?surface) (near robot ?surface)
     (park manipulator) )
   ( (park manipulator) )
   ( (near manipulator ?object) )
)
```

Listing 5.6: Symbolic definition of reach operator. Two definitions according to different locations of the object to be reached for.



(a) Robot grasps a cup



(b) Robot grasps a bucket

Figure 5.10.: Robot grasping different objects.

```
(:operator (!grasp ?object)
   ( (use hand) (use force)
     (on ?object floor) (near robot ?object)
     (near manipulator ?object) )
   ( (near robot ?object) (near manipulator ?object) )
   ( (manipulator have ?object) )
)

(:operator (!grasp ?object)
   ( (use hand) (use force)
     (on ?object ?surface) (near robot ?surface)
     (near manipulator ?object) )
   ( (on ?object ?surface)  (near manipulator ?object) )
   ( (manipulator have ?object) )
)
```

Listing 5.7: Symbolic definition of grasp operator. The two definitions differ in preconditions and effect due to different initial locations of the object.

here focuses on system integration rather than on further development of one particular robot skill, such elaborated approaches are not followed up. The integration of the grasping library that resulted from the work of (Baier-Löwenstein, 2008) is discussed in Section 8.4.

Listing 5.7 shows the symbolic operator for the grasp object skill. Similar to the reach operator, two alternative definitions have do be implemented according to the location of the object.

5.3.4. Move manipulator in transport position

A position of the robot arm and hand that is safe for mobile motions, i.e. that minimizes the additional measures of the robot resulting from supernatant objects, is called transport position. Such positions are important if the robot moves along narrow passages. Individual transport positions will be used if the robot carries different objects in its hand. For example, a cup should be held upright. Figure 5.11 shows transport positions of the arm without a cup, a bucket and without an object.

The trajectory from the grasping position to the transport position is provided by the RCCL library. This is similar to the reach task but with the difference that Cartesian interpolation is used by default. Listing 5.8 shows the symbolic definition. The particular object is not required for symbolic planning, however, it will be passed to the execution layer to consider object-specific properties. If the robot has no object in its hand, the symbolic variable will be instantiated with the default value "empty".

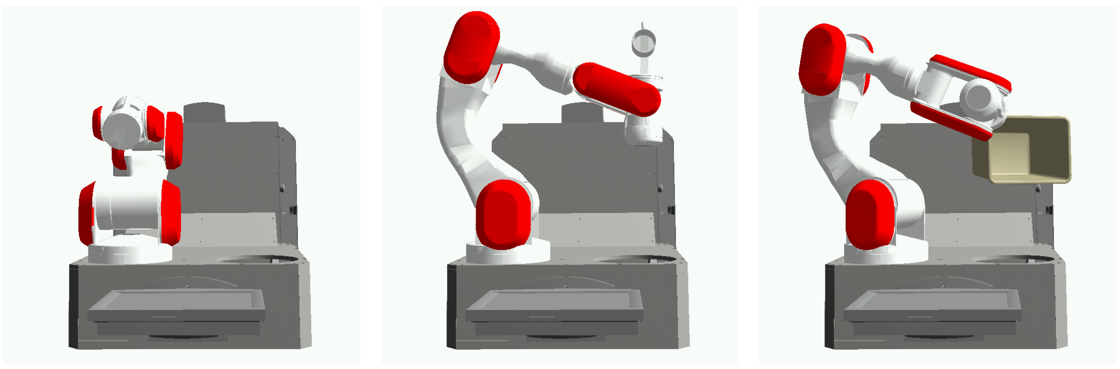


Figure 5.11.: Top-down view of a simulated robot: transport positions for different objects increase the overall measures of the robot.

```
(:operator (!park manipulator ?object)
        ( (not (manipulator park) (use arm) )
        ( )
        ( (manipulator park) )
)
```

Listing 5.8: The atomic operator for the safe-park skill.

5.3.5. Cross door to another room

The geometric path planner for the mobile platform is capable of planning valid paths through narrow passages such as doors. However, the planner approximates the shape of the robot as a circle which is not always appropriate. If the robot carries different objects, the enclosing diameter which is used by the path planner has to be increased for safety reasons. Figure 5.11 shows the robot with different transport positions of the arm. Specifically, positions which increase the robot's size mostly in longitudinal direction may lead to the inability of the planner to generate paths for narrow passages. Since the planner approximates the robot as a circle, it would not find solutions that could actually be executed due to smaller lateral measures of the robot. To avoid the robot's possible inability of passing through doors, this behavior is performed by a dedicated robot skill. The symbolic description is shown in Listing 5.9.

The physical execution of the action computes two waypoints for the mobile platform, one in front of the door and one behind it. By approaching the latter in a straight line from the first point, it is assured that the robot keeps the maximum distance to the door frame. Rotation of the mobile platform is not permitted within the door frame. Furthermore, a collision avoidance mechanism is active which would recognize if the door is not opened completely or if the door is too narrow for the measures of the robot.

```
(:operator (!cross_door ?door)
   ( (use mobile) (use laser)
     (door ?door) (near robot ?door) (in robot ?room)
     (connect ?door ?room ?other) (not (closed ?door)) )
   ( (near robot ?door) (in robot ?room) )
   ( (in robot ?other) )
)
```

Listing 5.9: Operator for crossing a door.

5.3.6. Open doorknob

This manipulation action assumes that the exact position of the door is known and that the robot hand has already reached for a position above the handle.

The action proceeds as follows.

1. The assumed distance to the door will be assured by touching the door.
2. The hand moves back above the handle to push it down. From the point where a finger first contacts the handle, it is sufficient to push 5 more centimeters to unlock it.
3. To ensure that the lock does not snap back, the hand will be moved in the direction of the door.
4. To verify whether the action was successful, the robot pushes the door again. The action was successful if the door shows less resistance.

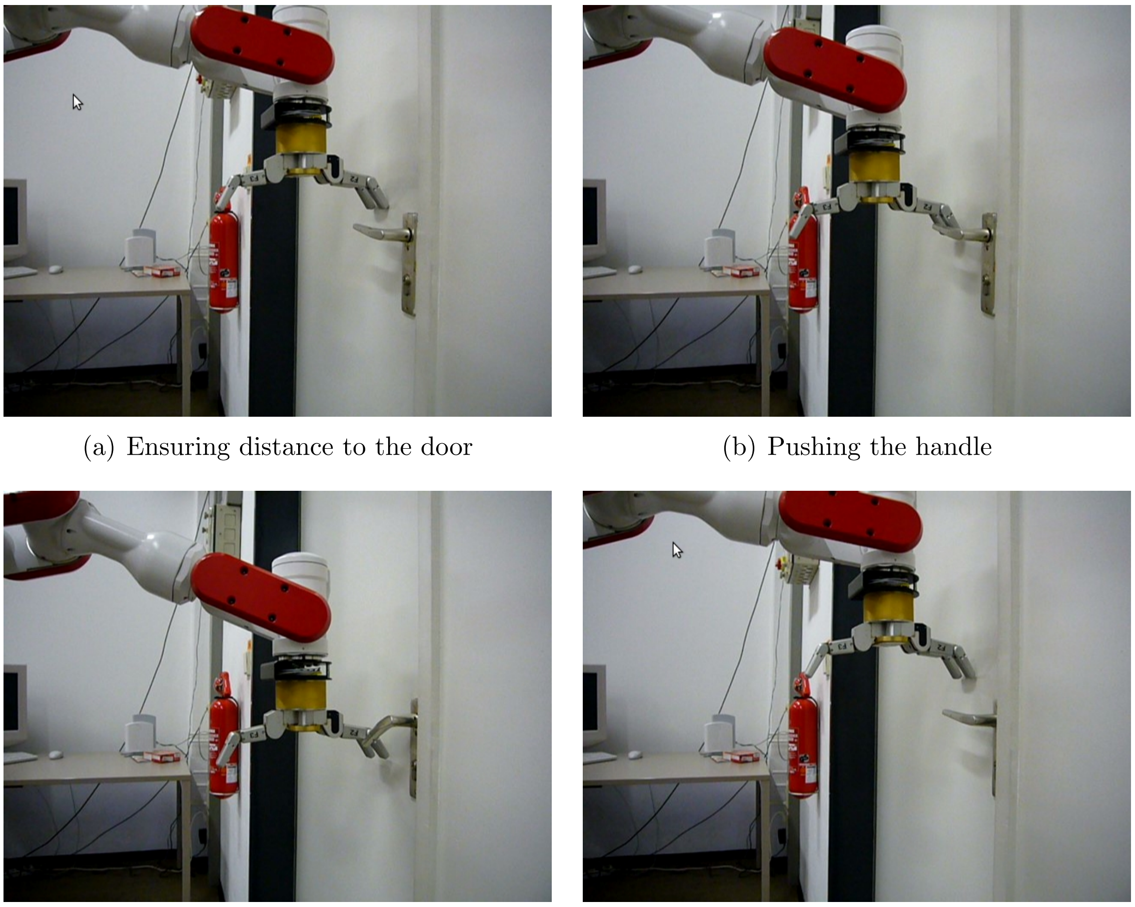
Figure 5.12 shows four stages of the robot opening a door. This action is by far not atomic in its execution; nevertheless to the symbolic layer, this action appears to be atomic. The symbolic operator is shown in Listing 5.10.

5.4. Summary and discussion

The following section summarizes described implementations and findings from this chapter. The discussion in Section 5.4.2 mainly concerns the compliance of the implemented skills with the arguments on appropriate abstraction levels of atomic robot skills given in Section 4.2.5.

5.4.1. Summary of contributions

In this chapter, the definition and implementation of an initial set of robot skills for plan-based robot control have been described. The implementation of skills consists



(a) Ensuring distance to the door

(b) Pushing the handle

(c) Ensuring that the lock does not snap back

(d) Verify whether the action was successful

Figure 5.12.: Four stages of the robot opening a door.

```
(:operator (!open ?door)
   ( (use arm) (use hand) (use force)
     (closed ?door) (near manipulator ?door)
         (near robot ?door) (poseknown ?door)
   ( (near manipulator ?door) (closed ?door) )
   ( () )
)
```

Listing 5.10: The open door skill is by far not atomic in its execution, still it appears atomic at the planning layer. This listing shows the symbolic definition. This operator does not introduce new facts about the world since a door is assumed to be open if no (closed door) atom is present any more.

of both the primitive operator for symbolic planning processes and the robot control program to be executed physically with the robot platform.

The symbolic representations of skills in the form of primitive operators will reveal their value mostly in the following chapter on robot planning. They will be compiled to a hierarchy of abstract robot tasks which will serve as a domain for AI-based planning.

Most implemented robot control programs have been tested in an unmodified office environment. Results from action execution in initial small scale experiments have been presented. It has been shown that the integration of multiple sensors and actuators on the skill level is essential for reliable complex robot behavior. Both the generation of actions and the perception of the environment can benefit from multi-modal combination to increase robustness and accuracy. While the find door skill is the only skill that utilizes multi-sensor *fusion* according to the definition of (Luo and Kay, 1995), it could be shown that the *integration* of active processes increases the reliability and applicability of robot perception routines. Unreliable estimations of tables, for example, can be verified using simple image algorithms if the camera is actively focused to a certain part of a table candidate. Without the support of actions and the utilization of prior knowledge, an algorithm for table detection in images would be much more complex, if possible at all. Similarly, cups on tables require the camera to be focused on the table scene. The use of force sensors for perception is generally only possible if combined with motion of the robot arm and hand.

To sum up, proactive perception allows for the application of algorithms which would otherwise not be possible or intractable. Reliable perception of tables, cups, door states and others could not be realized without actively focusing the robot's attention to certain parts of the world.

5.4.2. Discussion

The initial implementation of a library of robot skills aims at demonstrating the feasibility of a layered architecture for plan-based robot control using AI-based planning techniques. Since this project focuses on the integration of robot skills into one comprehensive robot system, it does not compete with narrow research topics such as grasping, localization, multi-sensor integration, object recognition, etc. Other research that specifically focuses on single robot behaviors may be more elaborated than the algorithms used in this work. However, due to a modular structure, state-of-the-art algorithms and robot control techniques can be easily integrated. References to publications on methods and results of this research are given in Chapter 3. This related work is indispensable for providing practical and theoretical foundations for the implementation of the respective robot skills.

The quality of the defined robot skills with respect to the guidelines given in Section 4.2.5 is hard to demonstrate. The compliance of some of the arguments cannot be discussed until the robot skills have been executed according to a plan which was generated by a symbolic planner. However, efficient task-specific algorithms (4.2.5, B.3) are extensively used in most of the executive skills. Manipulation actions utilize RCCL as specific purpose planner to solve inverse and direct kinematics and to provide arm trajectories. A geometric path planner for mobility as well as ambient processes that provide a self-localization service at any time alleviate the symbolic layer. These tasks would clearly go beyond the scope of any general purpose planner that has been developed so far. Verifiability of executive skills (4.2.5, A.5) is given by comparing the expected robot location, force values, joint angles and finger positions with the actual ones. Concerns regarding the concurrent execution of control programs have not been considered since most AI planners produce plans by means of sequences of atomic actions.

6 Planning for Extended Autonomy and Complex Robot Behavior

Planning is just a way of avoiding
figuring out what to do next.
Rodney Brooks, 1987

The previous chapter described the definition and implementation of atomic robot skills. The symbolic representation of these skills has been explicitly defined with respect to its use as basic building blocks for deliberative planning. In this chapter, the planning process is described that generates executable sequences of atomic skills which constitute solutions for abstract robot tasks given by human instructors.

The literature review in Chapter 3 led to the conclusion that general purpose planners without any additional domain knowledge guiding the search process tend to be intractable if applied to realistic domains. Hierarchical task nets (HTN) have proven to be an efficient tool that significantly improves planning performance while providing a convenient way to introduce expert knowledge about planning domains. Domain knowledge in the form of abstraction hierarchies of robot actions constitutes a robot memory at an abstract level that supports deliberative planning processes. In this chapter, it will be investigated whether AI-based HTN planning can be used without major modifications in the domain of embodied robot systems.

A major difficulty in integrating AI-based planning into robot control architectures is the need to overcome the closed-world assumption (CWA) which is underlying every symbolic planning process. The CWA implies that everything that is not explicitly stated in the world description is assumed to be false. This prevents traditional AI planners from directly generating plans which depend on objects or facts that have not been perceived at planning time.

The decision to use HTN planning for this project leads to the question of how abstraction hierarchies for robot skills should be designed and implemented such that planning and plan execution methods can achieve substantial improvement in robot behavior for everyday manipulation tasks in human environments. Two methods to overcome the discrepancy of the CWA on the one hand, and the incompleteness of world knowledge that is inherent to real robot domains on the other, will be proposed in this chapter. They constitute patterns for the interplay of planning and control and will be applied whenever perception is explicitly needed on the planning level. They contribute to answering the research question stated above.

The following section provides a formal description of notions for HTN planning. Specifically, basic operators which have already been introduced informally in Chapter 4 and hierarchical compositions of complex robot tasks will be defined. Section 6.2 briefly describes the open source planning tool JSHOP2, which is used in this project as deliberative component. The integration of this planning tool into the control architecture of the robot TASER will be described in Section 6.3. This section shows how HTN planning can be used with embodied robot architectures. Section 6.4 describes an HTN definition for the service robot scenario used in this work. The chapter concludes with a summary and discussion of the described implementations and findings.

6.1. HTN Planning with ordered task decomposition

Hierarchical Task Network (HTN) planning is like classical planning in that each state of the world is represented by a set of atoms, i.e. facts about the world, and each atomic operation corresponds to a deterministic state transition. It differs from classical planners in what they plan for and how they plan for it (Ghallab et al., 2004a). HTN planning does not plan to reach a certain state which is represented as a Boolean formula on a set of logical atoms, but rather to perform a set of tasks. Planning proceeds by decomposing tasks recursively into smaller and smaller subtasks until atomic operators are achieved.

The basic ideas of HTN planning originate from (Tate, 1977). HTN planning has been more frequently used in planning applications than any other planning techniques. Examples include logistics (Biundo and Schattenberg, 2001), manufacturing process planning (Smith et al., 1997), production-line scheduling (Wilkins, 1988), robotics (Morriset and Ghallab, 2002) and others. One reason for this is that HTN methods provide a convenient way to code problem solving strategies in domain descriptions. Expert knowledge can be represented in hierarchical structures that are fairly readable for humans.

Compared to classical planners[1], the primary advantages of HTN planners are their sophisticated reasoning and knowledge representation capabilities. Likewise, their primary disadvantage is the need to provide not only a set of operators but also domain-specific planning and decomposition strategies coded in the hierarchical task net.

Basic operators have already been introduced in the previous chapter to describe atomic robot skills symbolically. The syntax of the planner which is used here, i.e. JSHOP2, is used to represent these operators. In the following section, basic operators will be

[1]Classical planning refers to planning for restricted state-transition systems (Russell and Norvig, 2003), also known as STRIPS planning

defined more formally along with the notion of HTN methods. HTN methods are the construct for building hierarchies of robot tasks. This is described in Section 6.1.3.

The formal definitions of operators, methods, domains and problems given in the following sections are taken from (Ghallab et al., 2004b). The descriptions given here differ slightly from general definitions for HTN planning as they are adapted to the syntactic possibilities of the particular planner being used.

6.1.1. Definition of primitive operators and methods

To define a primitive *operator*, which basically describes a transition from one state to another, the term *state* has to be defined first. The term *method* is then defined as a conditioned sequence of tasks. A *task* is the conjunction of methods and operators. If a task is an operator, then it is called a *primitive* task; otherwise, it is called *complex* or *nonprimitive*.

state: A state s is a set of ground atoms of a first-order language L. An atom p holds in s iff $p \in s$. A set of literals g (i.e. atoms and negated atoms, possibly parameterized) is satisfied by s iff a substitution σ for variables in g exists such that every positive literal from $\sigma(g)$ is element of s and none of the negated literals from $\sigma(g)$ is element of s.

operator: An operator is a quadruple $o = (n(o), p(o), d(o), a(o))$ were

$n(o)$ is a unique name of the operator. It is of the form $n(x_1 \dots x_k)$ with an arbitrary name n and a list of all variables n_i that may appear in $p(o)$, $d(o)$ or $a(o)$.

$p(o)$ are preconditions for the operator represented as a set of literals. If this set is satisfied in s, the operator can be applied.

$d(o)$ and $a(o)$ describe the effect of the operator o on a state s. They are sets of ground atoms that will be deleted ($d(o)$) and added ($a(o)$) from the initial state s.

If an operator is applied to a state s, all variables in $d(o)$ and $a(o)$ are substituted by the same substitution σ that satisfies the preconditions. The state s_{t+1} after applying of the operator becomes the set-theoretical expression $s_{t+1} \leftarrow s_t \cup \sigma(a(o)) \setminus \sigma(d(o))$.

method: A method is a triple $m = (n(m), p(m), e(m))$ were

$n(m)$ is a unique name of the method. It has the same form as the name of an operator.

$p(m)$ are preconditions for the method. They are equally defined as the preconditions for operators.

$e(m)$ is the effect of the method. It is a set of tasks that defines the hierarchical decomposition of the method. All variables of the resulting subtasks will be substituted by the same substitution σ that satisfies the preconditions if the method is applied. The effect of a method is specified as (partially) ordered or unordered set – depending on the capabilities of the planner. In this work only ordered sequences of tasks will be used.

A HTN method as defined above may appear in three different forms: it may be composed of tasks that are all primitive, it may contain complex tasks or it may be composed of complex tasks only[2]. A method is called *primitive* if it is composed of primitive tasks only, otherwise it is called *nonprimitive*.

Since operators and methods are accessed by their names which have the same syntax they can be embraced by the term *task*.

6.1.2. Planning domains and problems

In the following the terms *planning domain*, *planning problem* and *plan* are defined.

planning domain A domain $D = (O, M)$ consists of the set O of basic operators that are available to the system and a set M of composed methods based on the operators.

planning problem A planning problem is the triple $P = (s_0, T, D)$ where s_0 is the initial state, T is a list of tasks to be solved and D is the domain in which the tasks $t_i \in T$ should be achieved.

In the planning process, the first nonprimitive task will be selected from the task list of the problem, and it will be substituted by its effect. Only substitutions with preconditions that are fulfilled in the problem's world state are taken into account. The decomposition process is recursively applied until no nonprimitive task remains. The resulting primitive operators $\pi = \{o_1, \dots o_n\}$ constitute a valid *plan* if they can be applied sequentially to the initial world state.

6.1.3. Composed method hierarchy

The definition of operator, method and task as given in the previous section can be represented visually as a directed graph. Primitive tasks are always leaves while the root node and intermediate nodes are complex tasks. Figure 6.1 shows a simple example of such a hierarchical task net (HTN). The color of boxes will be used in the following to illustrate the type of the task: gray denotes that a task is complex (i.e. a method) and blue denotes primitive tasks (i.e. operators). The rightward-pointing arrow represents the ordering of the subtasks. Since only ordered sets of tasks will be used as method decomposition within this work, the arrow will be omitted in all following figures.

HTN methods as defined in the previous section are potentially recursive since the effect of a method may comprise other methods or the method itself. As described in Section 6.1.1, in the application of a HTN to a certain problem all variables will be assigned

[2]The effect of a method can also be an empty set of tasks, however, this is a special case that is of no practical relevance.

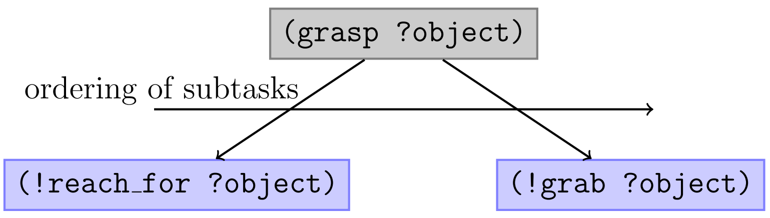


Figure 6.1.: A simple hierarchical task net example. The `(grasp ?object)` task is decomposed into primitive `reach` and `grab` operators.

```
(defproblem example taser (
  (use arm) (use hand)
  (table t) (near robot t)
  (cup cup7) (on cup7 t) )
  ( (grasp cup7) )
)
```

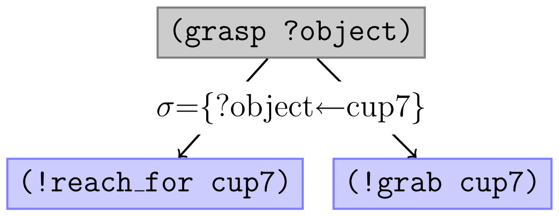


Figure 6.2.: The problem definition (left) represents the current state and a list of abstract tasks that are to be solved. The list contains only one task: `(grasp cup7)`. The HTN example from Figure 6.1 is applied to the problem by assigning the identifier `cup7` to the variable `?object`.

unique literals from the world state. Figure 6.2 shows the `(grasp ?object)` again, this time applied to an example problem which is shown next to the decomposition tree. The ordered sequence of instantiated operators from this method decomposition provides a plan for the given problem.

The notion of task decomposition provides a powerful tool to describe expert knowledge of certain domains in a fairly natural way. Furthermore, the hierarchical nature of HTNs provides simple graphical representations that are easy to be interpreted. However, an HTN planner may generate plans that contain thousands of nodes which are difficult for humans to understand. In such cases it is useful to visualize the structure of the decomposition tree at various levels of abstraction as provided in the planners O-PLAN (Tate et al., 1994) and SIPE-2 (Wilkins, 1990).

6.2. The HTN planning system JSHOP2

SHOP2 (Simple Hierarchical Ordered Planner 2) is a domain-independent configurable planning system based on Hierarchical Task Network (HTN) planning. In the 2002 International Planning Competition, SHOP2 received one of the top four awards which was also one of the two awards for distinguished performance. SHOP2 has been developed at the University of Maryland and has been made open source since June 14, 2002[3]. It is still under active development (last release on September 2, 2008) and provides implementations in both Lisp and Java.

JSHOP2 is the java implementation of SHOP2. While it is functionally identical to the Lisp implementation, it differs from other configurable planners in how domain descriptions are processed. A unique feature of this planner is a two-fold compilation and planning process (Ilghami and Nau, 2003). While other planners interpret domain descriptions directly, JSHOP2 compiles domain descriptions to programs specifically generated for that planning domain. The program can then be run to solve problems in that particular domain. This process allows implementation-level optimizations that are otherwise not possible and have not been explored in previous research on AI planning (Ilghami, 2006). To a certain degree, it is possible to perform domain optimizations that are usually only subject to domain-specific planners.

Another reason for the comparably good performance of JSHOP2 is the restriction that it is based on ordered task decomposition. This is a modification to general HTN planning in that it plans for tasks in the same order that they will later be executed. This restriction has been made with the purpose of decreased planning time (Ilghami, 2006). A rather unusual feature that votes for JSHOP2, is the possibility to include native code calls in planning domains (Ilghami and Murdock, 2005). This can be used to influence decisions on the planning level based on sensor readings.

Although JSHOP2 incorporates many features from PDDL, such as quantifiers and conditional effects, it cannot naturally handle temporal planning domains. Temporal PDDL operators have to be translated into SHOP2 operators that maintain bookkeeping information for (multiple) timelines within the current state. However, temporal aspects are not explicitly represented in the domain of the service robot TASER.

In addition to the exceptional good performance, the seamless integration into TASER's software environment due to the same programming language (TASER's software environment is mostly written in Java, see Section 2.3) is an important reason for the choice of this particular planner. Other advantages of JSHOP2 that led to this decision are its availability as open source and an expected support of a large user community.

6.3. Integrating HTN planning and robot control

The part of robot control that is superior to both the planning layer and the executive layer could be called intention, emotion, mind or just superordinate control structure. Although some people claim to build emotional systems for robots (Michaud, 2002; Ratanaswasd et al., 2005), current robot systems at the best emulate simple emotions. It is obvious that intentions are introduced explicitly by the developer. Furthermore, in the present work, a robot is considered as an advanced tool for the autonomous execution

[3] `http://sourceforge.net/projects/shop`, accessed on 25 April 2009

of service tasks rather than an artificial creature which simulates natural intelligence and emotion.

If the term robot is used in the following as if it were an intelligent individual that does something on purpose, this refers to the outer control structure of the overall robot system. The proposed strategy to define behavior of such a, say, meta controller in this project is the following: while the robot waits for new instructions it keeps still. The only active internal processes in such an idle mode are passive perception routines, such as localization.

Every time the robot receives a new instruction, it is stored in a task queue which is sequentially executed. Note that the robot does not receive goals stated as desired properties of the world. The instructions are formulated as abstract tasks to be achieved by the robot, and thus it is not a disadvantage to execute these sequentially. Changing the order of the remaining tasks in the task queue would possibly result in unwanted effects. For example, if two tasks in the queue are to empty the cup and to fill coffee into the cup, autonomous sorting of the tasks according to any given criteria is unwanted.

As long as the task queue is not empty, the robot generates a description of the planning domain and a problem description based on the current state of the world, the available robot devices, and the given instruction from the task queue. Based on this, the planner generates a plan which is executed sequentially by invoking the control programs of the execution layer according to the listed operators. Figure 6.3 shows the idealized control flow of the complete system. The process marked with (A) generates the description of the planning domain and planning problem, as will be described in the next section. Process (B) denotes the activation of the planning process, which results in a solution of the problem. This plan is then sequentially executed (C).

Obviously, the control flow in Figure 6.3 is the ideal case. Several problems may occur, e.g. the planning process may not come up with a valid plan or the execution of primitive tasks may fail. The following subsection describes the process of generating a domain and problem definition for the planning layer (mark (A) in the figure). Section 6.3.2 details the execution of plans and discusses exceptions that may occur in planning and action execution. Sections 6.3.3 and 6.3.4 describe patterns of the interaction of planning and execution that have been proven beneficial for the service robot scenario. It is mainly these patterns that contribute to answering the research question on the hierarchies of robot actions stated in the introduction of this chapter.

6.3.1. Autonomous definition of domain and problem descriptions

The robot autonomously generates the planning domain and the planning problem based on the current circumstances for each planning process.

The domain file describes two things: primitive robot operators which are directly dependent on the available robot devices and their hierarchical abstractions in the form of HTN methods. Thus it depends not on the task to be solved nor on the current world

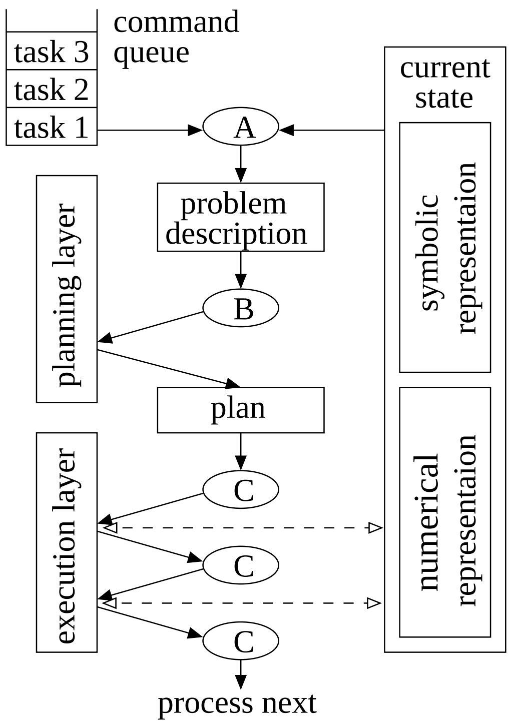
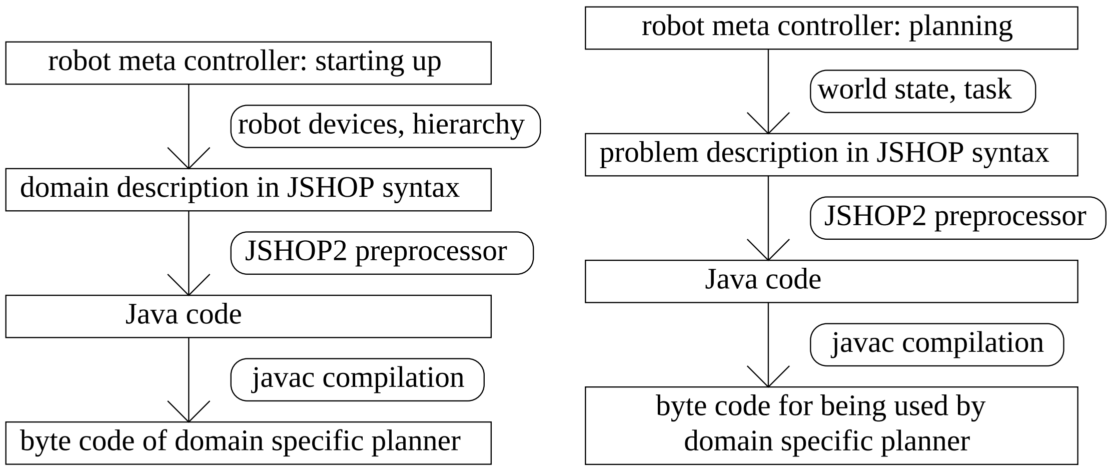


Figure 6.3.: Control flow of the overall robot architecture. If tasks are to be executed, the description of the planning problem will be generated (A). Then, the planner will be invoked to create a solution to this problem (B). This plan will then be executed sequentially (C).

state at the time it is generated. If the hardware setup of the robot changes (e.g. due to malfunction of several sensors), this changes the capabilities of the robot which are represented in the domain description. If, for example, the robot should pass a closed door and the arm is not available, manipulation operators cannot be used in the planning process. The robot planner must generate a detour instead of opening a door. Thus, an appropriate way to quickly change the planner's behavior with respect to the prevailing robot setup is needed.

To avoid repetitive generation of domain-dependent parts of the planner with the precompiler of JSHOP2, primitive operators state their required robot devices as preconditions. Malfunctions of robot devices will be reflected in the problem definition, such that the affected operators cannot be applied anymore. Using this approach, the domain description needs to be compiled only once if the robot system is started.

In the compilation process, the domain description will be generated based on the complete robot setup including devices that are temporarily not available. The generated domain description in JSHOP2 syntax is then precompiled to Java code for domain-specific components of the planner, which are finally processed by the Java compiler. The compilation process to generate the domain-dependent parts of the planner is illustrated in Figure 6.4(a).


robot meta controller: starting up
robot devices, hierarchy
domain description in JSHOP syntax
JSHOP2 preprocessor
Java code
javac compilation
byte code of domain specific planner
robot meta controller: planning
world state, task
problem description in JSHOP syntax
JSHOP2 preprocessor
Java code
javac compilation
byte code for being used by domain specific planner

(a) Compilation of domain specific planner (b) Compilation of planning problem

Figure 6.4.: Threefold compilation process to generate the planning domain and the problem for the planner JSHOP2 using internal representation of operators, the current world state and abstract tasks.

In contrast to the planning domain, the planning problem changes with virtually every new situation and with every new task given by a human instructor. Thus it has to be regenerated for every single planning process. The compilation process is similar to the planning domain, but with the difference that the current task and situation are taken into account in the first step. It is illustrated in Figure 6.4(b).

The complete compilation processes is autonomously triggered by the robot's meta controller every time the robot receives new tasks or the circumstances are changed. The circumstances include information about the external environment as well as the availability of the robot's hardware devices. Availability of the robot devices is guaranteed through regular requests via Roblets to the respective Roblet-Servers. In case of malfunction of a certain sensor, the sensor will not be included in the description of the current state. Thus operators that require this particular device in the preconditions can not be applied.

6.3.2. Sequential plan execution and exception handling

The ideal case is that a valid plan for the given task can be found and that all primitive operators can be executed physically with the respective robot control programs. In such a case, the general control flow is straightforward: all actions of the plan are executed sequentially without any interruption. However, different exceptions may arise that prevent the idealized control flow, which is already illustrated in Figure 6.3. The following paragraphs categorize possible exceptions and describe how the system behaves in such situations.

Invalid internal world representation The symbolic representation of the world may be inconsistent with the external world or the external world has changed in the meantime. Examples are doors that have been closed recently or objects that have been taken away.
exception handling: the control program being executed when the exception occurred will change the world representation autonomously in both symbolic and numeric representations. Replanning will be initiated based on the improved world state.

Unavailability of robot devices Malfunction of the robot hardware prevents the execution of an action.
exception handling: The executed control program will remove the symbol from the world state that marked the particular device as available, and causes replanning. The planner will then try to find an alternative plan in which the corrupted device will not be used.

Planning exception The planner cannot find a solution to a problem.
exception handling: For many situations, a change of the circumstances of the robot is sufficient to enable the planner to find a solution. In such cases the planner may provide a preliminary plan that first changes the robot's position and then tries to find a solution for the problem again. However, this approach has to be specified in the planning domain. Otherwise the robot requires help from a human instructor.

Execution exception An action may not succeed due to insufficiencies of the control or the hardware of the robot. This exception may have several reasons, for example the lack of satisfactory perception routines, an upcoming collision which cannot be avoided autonomously at the execution layer, or a singularity in the robot arm trajectory.
exception handling: In such cases, the action will mark its own symbolic representation as unsuccessful and abort to cause replanning. In this way, the planner tries to find an alternative plan. In order to remove the mark after execution of the task, an operator will be added to the task queue that has no other effect than cleaning the symbolic world description of such bookkeeping information.

The execution exception states the fallback position for all kinds of exceptions that cannot be attributed to any of the first three cases. As additional safety precaution, the meta controller checks every two successive problem definitions for similarity to prevent loops in the planning process.

6.3.3. A pattern for perception routines: deliberative replanning

The closed world assumption prevents AI-based symbolic planners from finding plans that include unknown objects – it is assumed that objects which are not present in the domain representation are not existent at all. This signifies a major problem for plan-based robot control since the external world cannot be known or perceived entirely.

Goal-directed sequences of robot actions often include perception routines that provide information at execution time which would have been needed at planning time. For example, instructions for the robot can depend on external entities that are not known to the robot at planning time. A robot may be instructed to grasp an unknown object from a table, or it may bring all cups to an unknown table in the kitchen. Since the unknown objects are not part of the symbolic world state, an unmodified HTN planner cannot provide solutions to such tasks.

In the introduction of this chapter, the research question of whether AI-based planning tools can be used without major modifications in the domain of embodied robot systems was posed. The generation of solutions for tasks that depend on objects which are not known at planning time is an important requirement for answering this question positively.

The proposed approach to deal with indefinite tasks is to introduce a primitive operator that causes the robot to replan deliberatively. Thus the planning layer can be triggered by the execution layer if this operator is executed. Using this operator, an indefinite task can be solved by generating a (possibly complex) plan to acquire the needed knowledge and generating a solution for the task taking the improved world knowledge into account. The example task to grasp an unknown object from a table will be used to explain the procedure.

In the planning domain, two possible decompositions of a `(pick_from ?table)` task will be defined as shown in Listing 6.1. If an object on the table is represented in the world state, the abstract task will be substituted by an ordinary plan to approach the table, localize the object, and pick it up. If the object is not known, the task decomposition will let the robot approach the table, search for an object and generate a plan based on the acquired knowledge. The `find_cup` operator changes the state of the world according to the results of its own execution. If the object of interest is added to the symbolic state of the world, the planner will find a valid plan in the second attempt. The bookkeeping literal `(nosuccess find_cup)` will be added if no object could be found and prevents infinite looping of planning and execution of the perception.

In this simple example, this solution appears trivial. However, more complex plans may be generated to acquire the needed knowledge.

This pattern of perception and active replanning is applied in tasks that depend on unknown objects at planning level. It provides a solution to the problem of the planner's inability to handle unknown objects and turns around the usual dependency of the planning and execution layer. Instead of being triggered by the deliberative layer, the execution layer invokes deliberative planning actively.

```
(:method (pick_from ?table)
  ; with known object on table
  ( (use arm) (use ptu) (use camera) (use mobile)
    (table ?table) (cup ?object) (on ?object ?table)
  )
  ( (approach ?table) (!localize camera ?cup)
    (!reach_for ?cup) (!grab ?cup)
  )

  ; without knowledge of object
  ( (use arm) (use ptu) (use camera) (use mobile)
    (table ?table) (not (nosuccess find_cup))
  )
  ( (approach ?table) (!find_cup) (!replan) )
)
```

Listing 6.1: The `(pick_from ?table)` method provides two different decompositions depending on the knowledge represented in symbolic terms.

6.3.4. A pattern for predictable perception routines: using exceptions

The interplay between perception and replanning requires certain bookkeeping mechanisms. For example, the task to grasp any object from a table may be defined as a search for an object on one table and – if unsuccessfully – recursively search for an object on another table. In this case, the planner needs to know which tables have been checked already. To avoid repetitive searching on the same table, bookkeeping information is included into the world state. This bookkeeping information can be seen as a symbolic short-term memory. The primitive operators `(!!add...)` and `(!!rm...)` are introduced to add or remove symbols to the world state. Since this operators are not executed physically, they will not be taken into account for the calculation of the costs of a plan.

The overhead caused by bookkeeping mechanisms on symbol level may be dispensable if the outcome of the perception is predictable to a certain degree. In this case, the expected outcome of the perception will be added to the symbolic representation and the operator for initial perception will be substituted by an operator for confirmation of the assumed result. Replanning will be caused only if necessary in the unlikely case that the expected result could not be confirmed.

The `(confirm ?sensor ?object ?actuator)` operator was introduced in Section 5.2.6[4]. This operator can be used to confirm objects that have been artificially added to the

[4]This operator used in the Listings 6.2 without the `?actuator` parameter adds the default value `null` as the actuator and calls the operator from Section 5.4 on p. 68.

```
(:method (pick_from ?table ?object)
  ; with known object on table
  ( (use arm) (use ptu) (use camera) (use mobile)
    (table ?table) (cup ?object) (on ?object ?table)
  )
  ( (approach ?table) (!localize camera ?object)
    (!reach_for ?object) (!grab ?object)
  )

  ; without knowledge of object
  ( (use arm) (use ptu) (use camera) (use mobile)
    (table ?table) (not (nosuccess confirm camera ?object))
  )
  ( (approach ?table)
    (!!add (cup ?object)) (!!add (on ?object ?table)
    (!confirm camera ?object) (pick_from ?table ?object)
  )
)
```

Listing 6.2: Method to grasp object from table. In this revised definition, the object will be added to the world state and confirmed by sensors.

world state by the planner. Plans are then generated with the default assumption that the confirmation will proceed successfully. In this way, the planner avoids dispensable replanning. The precondition that precludes the `(nosuccess ...)` marker prevents unwanted loops in planning and execution.

This proposed pattern for the design of abstraction hierarchies for robot tasks taking perception with predictable result into account is used whenever prior assumptions can be deduced from the instruction. While the pattern described in the previous section utilizes deliberative replanning to permit robot plans that are dependent on perceived objects, this pattern needs to replan only if the default assumption on symbol level turns out to be wrong. In this way, the robot is able to generate plans that include perception while avoiding frequent replanning.

6.4. Planning domain for the service robot scenario

This section describes the definition of the planning domain for the service robot scenario which is used in this thesis to demonstrate and validate the proposed approach. The developed hierarchy of composed complex methods constitutes a symbolic memory for the robot which describes on an abstract level how to solfe complex tasks. The following subsections resume the scenarios given in the introductory chapter to illustrate the defined abstraction hierarchy of basic robot actions as described in Chapter 5.

6.4.1. Navigation in challenging environments

Navigation is a required ability for both other scenarios given in the introductory chapter. As already described in Section 5.3.1, the robot uses a geometric path planner at execution level to navigate safely around known objects. A reactive collision avoidance mechanism is also active that protects the robot from running into unknown obstacles. The deliberative component comes into play if the robot has to navigate across different rooms. A dedicated robot skill is used to change rooms safely by respecting the narrow passages while crossing a door. The symbolic planner serves as a topologic path planner that produces sequences of approach and crossing of doors. Listing 6.3 shows the definition of a navigation problem with the abstract task `(move_to ?room)` which provides four possible decompositions depending on the situation.

The problem becomes more interesting if one or more doors are closed. The problem description then remains the same except for an additional `(closed door1)` literal in the initial state. To permit usage of the manipulator, the literals `(use arm)` and `(use hand)` have to be added as well. Otherwise the robot could not open the door and the planner would not find a solution to the problem.

The task becomes even more challenging if the door is unexpectedly closed. Figure 6.5(a) shows a plan for the navigation problem given in Listing 6.3 which will fail in the case of an unexpectedly closed door. The `(!cross_door ?door)` operator will fail and add a `(closed ?door)` literal to the current state. An alternative plan based on the changed situation is shown in Figure 6.5(b).

The robot's possibility to navigate autonomously and safely across different rooms is only possible due to a combination of geometric path planning, reactive collision avoidance, manipulation skills and the ability to integrate these behaviors properly.

6.4.2. Pick-up and delivery services

Pick-up and delivery services are popular testbeds for mobile manipulation services. They require almost every ability that enables a robot to perform arbitrary tasks from a systems perspective. Admittedly, grasping of objects may be simplified to a rough grab, which is comparatively simple in contrast to other manipulations, such as handling of tools or using any kind of handles and switches. Still the combination of mobility, localization, perception and manipulation is an outstanding challenge inherent to any mobile manipulation task. Thus pick-up and delivery services are specially well-suited to demonstrate the level of sophistication of a robot system.

As described in the chapter on robot actions, the grasp of an object consists of several primitive robot operators; specifically they are mobile approach, perception of the (as much as possible) exact location of the target, reaching out for the object, performing

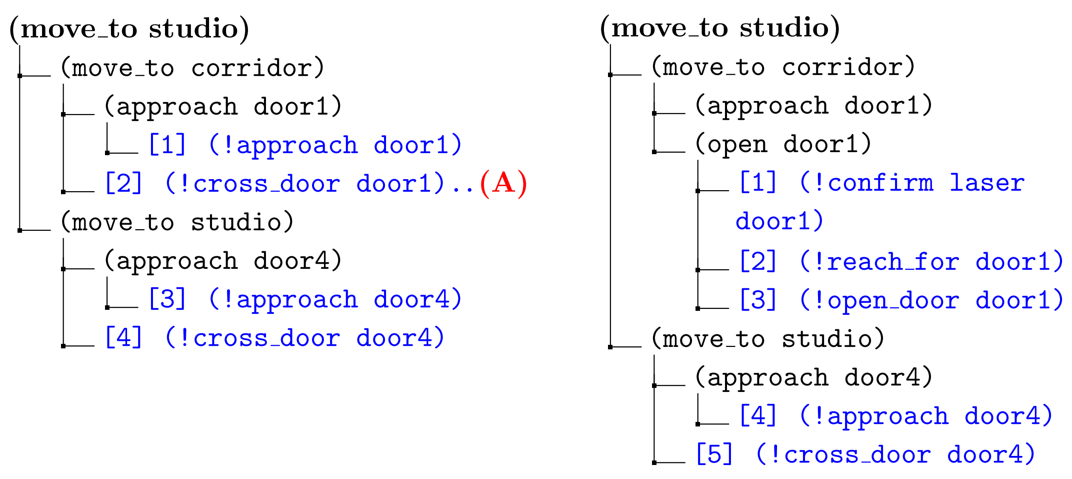
```
(defproblem problem taser
( (use mobile) (use laser)
  (room lab) (room studio) (room corridor)
  (door door1) (connect door1 lab corridor)
  (door door4) (connect door4 studio corridor)
  (in robot lab) )
( (move_to studio) ) )
```

```
(:method (move_to ?room)
  ; already there
  ( (in robot ?room) )
  ( )

  ; connected rooms, open door
  ( (in robot ?other) (connects ?door ?room ?other)
    (not (closed ?door))
  )
  ( (approach ?door) (!cross_door ?door) )

  ; connected rooms, closed door
  ( (in robot ?other) (connects ?door ?room ?other)
    (closed ?door) (manipulator park)
  )
  ( (approach ?door) (open ?door) ) ; the robot crosses the
                                     ; door when opening it
  ; go to the corridor first
  ( (in robot ?other) )
  ( (move_to corridor) (move_to ?room) )
)
```

Listing 6.3: The problem definition for a simple navigation task includes the required robot devices, rooms and their relations, as well as the robot's initial location. The method definition provides four possible decompositions depending on the current situation.



(a) Hierarchical plan for a navigation problem

(b) Alternative plan for the same navigation after recognizing that the door is closed

Figure 6.5.: The initial plan on the left could not be achieved due to an unexpected closed door. The `cross_door` operator (A) failed and caused replanning taking the new situation into account. For better visualization, the illustration of HTN trees is changed to a vertical arrangement in this and the following figures. Primitive operators are marked blue and start – by convention – with an exclamation mark.

the grasp (which actually is a rough grab in the current implementation), and moving the object into a transport position.

A possible location change in between the pick-up and place-down task is performed using mechanisms as described in the previous section. The place-down task consists of several primitive actions – similar to the pick-up task. A complete hierarchy of tasks which states a possible solution to pick-up a cup from one table and place it to another table in the same room is given in Figure 6.6.

If the robot should deliver an object to another room with a closed door in between, the planner recognizes the need to open the door before the robot picks up the object. This is necessary since the robot is equipped with only one manipulator. Thus door opening is impossible while the robot holds an object.

6.4.3. Inventory taking

Perceiving different objects in large-scale scenarios is still a challenging problem in robotics. This problem is avoided here by causing the robot to search explicitly for requested objects only. Abstract tasks are defined that let the robot search for certain classes of objects, and these abstract tasks are in turn combined into more general inventorying tasks.

```
(delivery_service cup1 table2)
  (fetch cup1)
    (approach table1)
      [1] (!approach table1)
    (pick_up cup1 table1)
      [2] (!focus ptu cup1)
      [3] (!confirm camera cup1)
      (grasp cup1)
        [4] (!reach_for cup1)
        [5] (!grab cup1)
      [6] (!reach_for transport)
  (deliver cup1 table2)
    (move_to lab)
    (approach table2)
      [7] (!leave table1)
      (approach table2)
        [8] (!approach table2)
    (place_down cup1 table2)
      [9] (!focus ptu table2)
      [10] (!find_surface)
      [11] (!reach_for surface)
      [12] (!release cup1)
      [13] (!reach_for park)
```

Figure 6.6.: Hierarchical representation of a plan for a simple pick-up and delivery service. The initial instruction is marked in bold. The sequence of primitive operators marked in blue state the executable solution to this task.

The abstract `(find_all_tables)` task will be described in the following as an example for an inventorying task. The task is decomposed recursively to search in all rooms for table candidates using laser range scans. The bookkeeping literal `(checked ?room)` is used to keep track of visited rooms and those that have to be visited. In every room, the primitive task `(!find_locally ?object ?sensor)` is instantiated with `table` and `laser` to perform the actual work. If all rooms are visited, the perceived tables are confirmed using image algorithms and the pan-tilt unit to focus the camera. The confirmation of a single table in isolation is also described in Section 5.2.6. The planning process that generates a sequence of actions to confirms all table candidates requires a representation of the table candidates at planning time. Since this is not available before the table candidates have been detected, the planner has to be reinvoked. The initial plan for the `(find_all_tables)` task is thus to perceive possible table candidates first, and then invokes replanning that takes the detected candidates into account. This approach to replanning deliberatively is also described in Section 6.3.3 as pattern for perception of objects. Listing 6.4 shows the recursively defined decomposition of the described

```
(:method (find_all_tables)
  ; this room
  ( (use laser) (in robot ?room) (not (checked ?room)) )
  ( (!find_locally table laser) (!!add (checked ?room))
    (find_all_tables)
  )

  ; other room
  ( (use laser) (room ?room) (not (checked ?room)) )
  ( (move_to ?room) (!find_locally table laser)
    (!!add (checked ?room)) (find_all_tables)
  )

  ; all rooms checked
  ( (not (confirm next)) )
  ( (!!add (confirm next)) (!replan) )

  ; confirm table candidates
  ( (confirm next) )
  ( (!!rm (confirm next)) (confirm_all_tables) (clean) )
)
```

Listing 6.4: Recursive definition of the `(find_all_tables)` method.

method. Experimental results that validate the proposed approach for inventory taking are given in Section 7.2.

6.5. Summary and discussion

In the beginning of this chapter, HTN planning has been introduced formally. The open source planning system JSHOP2 was described as it is used as the deliberative component of the robot system TASER. The question of whether an unmodified AI planner can be used for planning courses of actions for real-world embodied robots, can be answered positively. It needs to be shown in experiments whether the proposed approach is applicable for embodied robots in real world scenarios. Experimental results with the TAMS service robot are provided in the next Chapter.

To answer the initial research question of how abstraction hierarchies for robot skills should be designed and implemented such that planning and plan execution methods can achieve substantial improvement in robot behavior, two major points turned out to be specially useful.

First, planning and execution layer should mutual influence each other rather that driving the execution layer top-down by the planning layer. This allows flexible handling of exceptions that may arise e.g. from incorrect internal representations of external phenomena. Furthermore, a tight integration of AI-based planning and reactive skill execution into a coherent architecture avoids problems of the closed world assumption. This allows for planning under unknown conditions, e.g. planning for manipulation of objects which are unknown at planning time. Two patterns to handle this problem are proposed in Section 6.3.3 and 6.3.4. These patterns lead to valid plans and plan-execution cycles as described in Section 6.4.

And second, multi-modal aspects of robot sensors and actuators should be modeled at symbol level, too. If the requirements for primitive operators include availability of the used modalities, the domain definition does not have to be changed due to hardware malfunctions. In the particular case of JSHOP2, this prevents useless repetitive compilation of domain-specific parts of the planner.

The planning examples shown in Section 6.4 suggest that HTN planning can be used for the robot domain. Supposably it will decrease the workload for the programmers compared to other representations of domain knowledge.

7 Integrated experiments

A theory is something nobody believes,
except the person who made it.
An experiment is something everybody believes,
except the person who made it.
Albert Einstein, 1879-1955

This chapter aims at putting the parts of the hybrid deliberative and reactive robot control architecture together that have been described and developed in the previous chapters. While Chapter 8 will summarize findings from this thesis that are of general applicability, this chapter may be understood as a summary of the implementation work which has been done to validate and demonstrate the proposed approach.

For the implementation part of this thesis, the two most significant achievements are the implementation of a library of atomic multi-modal robot skills and the definition of a hierarchical memory representation of abstract robot tasks based on these atomic skills.

The service robot TASER has been significantly enhanced within the work for this thesis. Specifically, the integration of a deliberative planning component has been overdue and provides new areas of application that have not been possible before. Furthermore, the scientific scope of the overall robot platform has profited from this work – investigations on symbolic AI-based problem solving have not been carried out on this platform before.

The following subsections take up the three scenarios from the introduction again and describe results from experiments carried out with a real robot system. As navigation is a prerequisite for the other experiments, it is not validated in isolation. The experiments focus on the integration of all parts of the system.

7.1. Pick-up and delivery services

Figure 7.1 shows the topology of the experimental environment. It consists of four rooms that can be accessed by the robot. All rooms are connected with the corridor, the laboratory is connected to it via two doors. Six tables are distributed over the rooms

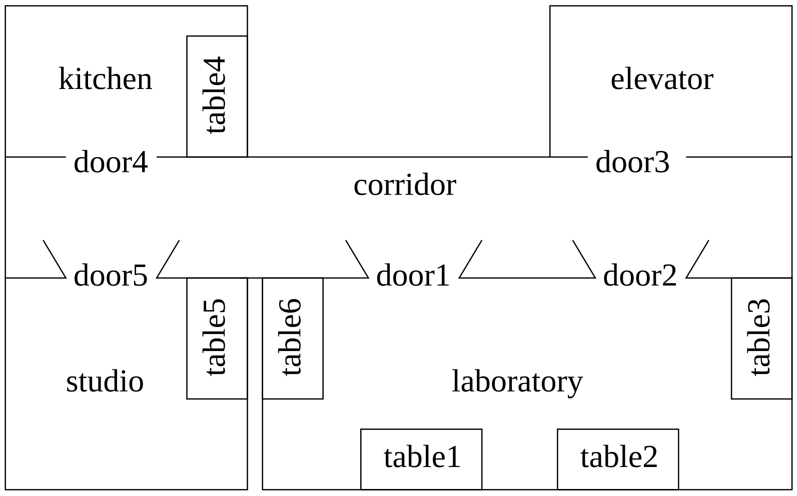


Figure 7.1.: Topology of the experimental environment that is used for integrated experiments. All shown rooms and tables can be accessed by the robot.

that can be accessed by the robot to put things on or grasp from them. The robot is able to grasp and place buckets in every room and the corridor. Several different objects like cups and small bottles are distributed on the tables. All graspable objects which are placed on and taken from the tables are symbolically labeled as cup since only the execution layer is affected by the specific type of an object. For all experiments, the robot initially started in the corridor with an empty hand and the manipulator in park position.

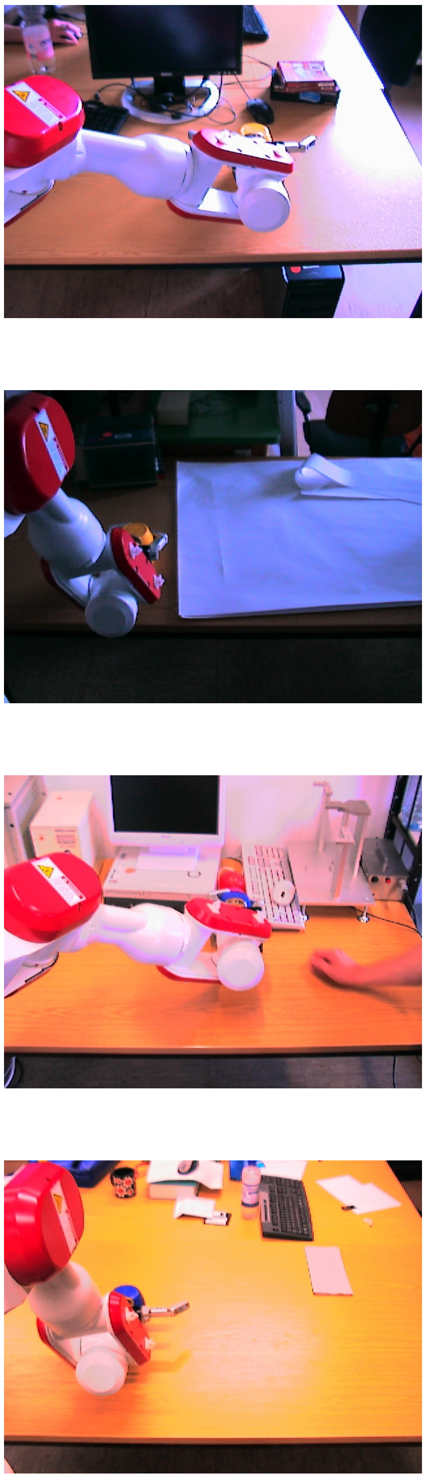
7.1.1. Long-term experiment

In a first task, the robot was instructed to bring cups into the kitchen. Two cups from tables in the laboratory and one from the studio. Since the robot can only carry one object at a time it has to deliver every cup separately. The generated plan for this task is shown in Figure 7.2(a). Images of the robot's execution of atomic tasks that are specified in the plan are shown in Figure 7.2(b). The images of the manipulator grasping and placing cups are taken from the perspective of the robot.

In this experiment, no unexpected events were introduced. Since all needed information for plan generation was known from the beginning, replanning was not needed. The experiment has been carried out three times, in all three trials the robot was able to carry out all subtasks without human intervention. On average, the robot needed 16:34 minutes to fetch and deliver all three cups.

```
(bring_all_cups_to kitchen) .................... (A)
 |__ (delivery_service cup1 table4) ............ (B)
 |    |__ (fetch cup1)
 |    |    |__ (approach table1)
 |    |    |    |__ (move_to lab)
 |    |    |    |    |__ (approach door1)
 |    |    |    |    |    |__ [1] (!approach door1) ...1:03
 |    |    |    |    |__ [2] (!cross_door door1) .....0:10
 |    |    |    |__ [3] (!approach table1) .........0:36
 |    |    |__ (pick_up cup1 table1)
 |    |         |__ [4] (!focus ptu cup1) .........0:01
 |    |         |__ [5] (!confirm camera cup1) .....0:04
 |    |         |__ (grasp cup1)
 |    |         |    |__ [6] (!reach_for cup1) .......0:13
 |    |         |    |__ [7] (!grab cup1) ............0:05
 |    |         |__ [8] (!reach_for transport)......0:13
 |    |__ (deliver cup1 table4)
 |         |__ (move_to kitchen)
 |         |    |__ (move_to corridor)
 |         |    |    |__ (approach door1)
 |         |    |    |    |__ [9] (!leave table1) ......0:03
 |         |    |    |    |__ (approach door1)
 |         |    |    |         |__ [10] (!approach door1)
 |         |    |    |              0:29
 |         |    |    |__ [11] (!cross_door door1) ....0:13
 |         |    |__ (move_to kitchen)
 |         |         |__ (approach door4)
 |         |         |    |__ [12] (!approach door4) ..1:23
 |         |         |__ [13] (!cross_door door4) ....0:10
 |         |__ (approach table4)
 |         |    |__ [14] (!approach table4) ........0:55
 |         |__ (place_down cup1 table4)
 |              |__ [15] (!focus ptu table4) .......0:00
 |              |__ [16] (!find_surface) ...........0:08
 |              |__ [17] (!reach_for surface) ......0:12
 |              |__ [18] (!release cup1) ...........0:07
 |              |__ [19] (!reach_for park) .........0:07
 |__ (bring_all_cups_to kitchen) ................. (C)
      |__ (delivery_service cup2 table4)
           |__ ........second and third cup......16:34
```

(a) Plan for delivery experiment, full listing for the first cup



(b) Robot perspective of grasp and place operator

Figure 7.2.: Plan and selected images for pick-up and delivery experiment. The initial task (A) is recursively decomposed into the delivery of one cup (B) and the delivery of all remaining cups (C). The recursion terminates if no other cups are to be delivered. The average execution time of the atomic robot skills (marked in blue) are given in [min:sec].

7.1.2. Dealing with unknown environments

In a second experiment, the robot had the task of bringing a bucket from the laboratory to the elevator. This experiment was carried out under four different conditions:

1. the environment is completely known and the door to be passed is open
2. the position of the bucket has changed. The `(confirm laser bucket)` action fails and causes replanning
3. the door that the robot has to pass was closed unexpectedly and thus the initial plan cannot be achieved. The robot has to replan based on the changed state
4. similar to the previous one, but with the difference that `door2` is also closed.

The initial plan for all conditions is the same. Except for the first condition, the initial plan could not be achieved and replanning was required. At execution time, the control programs of the failing action changed the symbolic representation of the world and caused replanning. The initial plan is shown in Figure 7.3(a). The operator [2] failed under Condition 2 and the operator [7] failed under Condition 3 and 4.

The recovery of the displaced bucket in the second condition is a perception routine which does not require deliberative replanning. Due to the formulation of the abstract task `(fetch bucket lab)`, the rough position of the bucket was known at symbol level (i.e. the room is known `(in ?bucket ?room)`). Figure 7.3(b) shows the revised plan for Condition 2.

Revised plans for the third and fourth condition are listed in Figure 7.4. Under Condition 3, the planner chose to pass the second door which connects the laboratory with the corridor, too. This required fewer primitive actions to be executed than opening the door. When the second door was also closed, which the robot was aware of, it was unavoidable to open a door. The robot had to set down the bucket before it opened the door. After that, it proceeded with the initial task of bringing the bucket to the elevator lobby.

The trouble-free execution of the task under the first condition needed 1:40 minutes in total. The additional time for the recovery of the displaced bucket under Condition 2 is negligible – the mobile approach to the changed position needed far more time depending on the distance of the bucket.

Recognition of the closed door, generation and compilation of a revised problem description and planning took less then 10 seconds altogether and is thus negligible. The detour through the alternative door in the third condition took 1:54 minutes in the experiment. This is more time than the trouble-free execution (Condition 1) needed in total, however, it is still less than opening the door. Under Condition 4, the robot had to open the door, which took more than 2 minutes (setting down the bucket, opening the door and refetching the bucket) with the current implementation of the robot skills.

```
(delivery_service bucket1 elevator)
  (fetch bucket1)
    (approach bucket1)
      [1] (!approach bucket1)
    (pick_up bucket1 floor)
      [2] (!confirm laser bucket1)
      (grasp bucket1)
        [3] (!reach_for bucket1)
        [4] (!grab bucket1)
      [5] (!reach_for transport)
  (deliver bucket1 elevator)
    (move_to elevator)
      (move_to corridor)
        (approach door1)
          [6] (!approach door1)
        [7] (!cross_door door1)
      (move_to elevator)
        (approach door3)
          [8] (!approach door3)
        [9] (!cross_door door3)
    (place_down bucket1 elevator)
      [10] (!reach_for drop)
      [11] (!release bucket1)
      [12] (!reach_for park)
```

(a) Initial plan: operator [2] will fail if the position of the bucket has changed and the operator [7] will fail if the door is closed unexpectedly

```
(delivery_service bucket1 elevator)
  (fetch bucket1)
    (pick_up bucket1 floor)
      (clean)
      [1] (!find_locally bucket1
      laser)
      (approach bucket1)
        [2] (!approach bucket1)
      (pick_up bucket1 floor)
        [3] (!confirm laser bucket1)
        (grasp bucket1)
          [4] (!reach_for bucket1)
          [5] (!grab bucket1)
        [6] (!reach_for transport)
  (deliver bucket1 elevator)........(A)
    (move_to elevator)
      (move_to corridor)
        (approach door1)
          [7] (!approach door1)
        [8] (!cross_door door1)
      (move_to elevator)
        (approach door3)
          [9] (!approach door3)
        [10] (!cross_door door3)
    (place_down bucket1 elevator)
      [11] (!reach_for drop)
      [12] (!release bucket1)
      [13] (!reach_for park)
```

(b) Revised plan: the robot has to find the bucket first [1] to proceed with the delivery (A)

Figure 7.3.: Initial plan for delivery task and alternative plan in the case of a changed position of the bucket.

```
(delivery_service bucket1 elevator)
 └─ (deliver bucket1 elevator)
     ├─ (move_to elevator)
     │   ├─ (move_to corridor)
     │   │   ├─ (approach door2)
     │   │   │   ├─ [1] (!leave
     │   │   │   │  door1)
     │   │   │   └─ (approach door2)
     │   │   │       └─ [2] (!approach
     │   │   │          door2)
     │   │   └─ [3] (!cross_door
     │   │      door2)
     │   └─ (move_to elevator)
     │       ├─ (approach door3)
     │       │   └─ [4] (!approach
     │       │      door3)
     │       └─ [5] (!cross_door
     │          door3)
     └─ (place_down bucket1
         elevator)
         ├─ [6] (!reach_for drop)
         ├─ [7] (!release bucket1)
         └─ [8] (!reach_for park)
```

(a) Plan for delivery of the bucket with unexpectedly closed door: the revised plan let the robot cross the second door to the corridor

```
(delivery_service bucket1 elevator)
 └─ (deliver bucket1 elevator)
     ├─ (move_to elevator)
     │   ├─ (move_to corridor)
     │   │   ├─ (drop bucket1)
     │   │   │   ├─ [1] (!leave door1)
     │   │   │   └─ (place_down bucket1 lab)
     │   │   │       ├─ [2] (!reach_for drop)
     │   │   │       ├─ [3] (!release bucket1)
     │   │   │       └─ [4] (!reach_for park)
     │   │   ├─ (approach door1)
     │   │   │   ├─ [5] (!leave bucket1)
     │   │   │   └─ (approach door1)
     │   │   │       └─ [6] (!approach door1)
     │   │   ├─ (open door1)
     │   │   │   ├─ [7] (!confirm laser
     │   │   │   │  door1)
     │   │   │   ├─ [8] (!reach_for door1)
     │   │   │   └─ [9] (!open_door door1)
     │   │   ├─ (fetch bucket1)
     │   │   │   ├─ (approach bucket1)
     │   │   │   │   └─ [10-13] ...
     │   │   │   └─ (pick_up bucket1 floor)
     │   │   │       └─ [14-16] ...
     │   │   └─ (move_to corridor)
     │   │       ├─ (approach door1)
     │   │       │   └─ [17] (!approach door1)
     │   │       └─ [18] (!cross_door door1)
     │   └─ (move_to elevator)
     │       ├─ (approach door3)
     │       │   └─ [19] (!approach door3)
     │       └─ [20] (!cross_door door3)
     └─ (place_down bucket1 elevator)
         ├─ [21] (!reach_for drop)
         ├─ [22] (!release bucket1)
         └─ [23] (!reach_for park)
```

(b) Plan for delivery of the bucket with unexpectedly closed door and no alternative open door: the robot had to open the door

Figure 7.4.: Plans for delivery of the bucket under challenging conditions. Note that the initial plan listed in Figure 7.3(a) was executed partially before this plans have been generated.

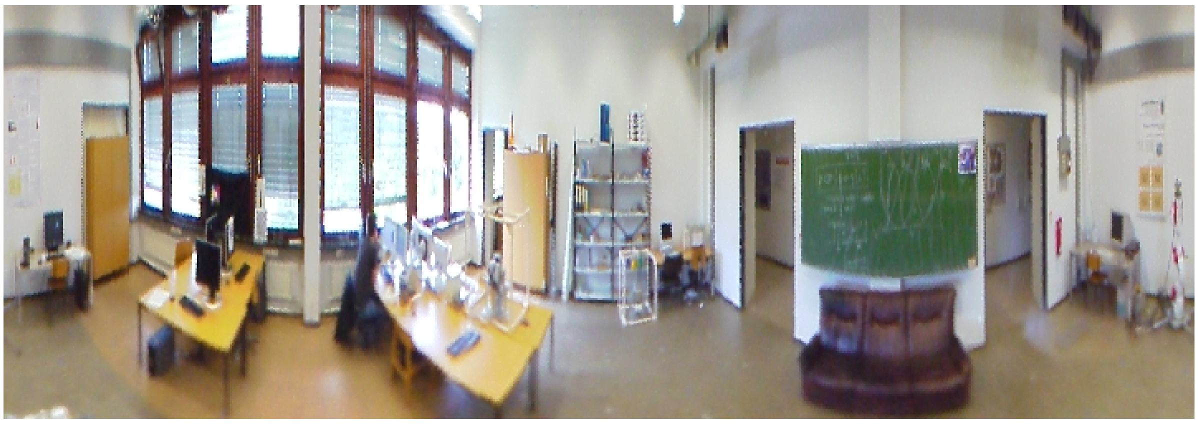


Figure 7.5.: Panoramic image of the experimental environment for proactive multi-modal perception.

7.2. Proactive multi-modal perception in natural environments

This experiment was carried out in the same environment as the previous experiments on delivery services (see Figure 7.1). The laboratory is used by several students working on different hard- and software projects and is not subject to strict tidiness regulations. Thus it provides a natural human environment with dynamic properties and clutter. This makes it a challenging environment for a robot to reliably perceive certain objects. Figure 7.5 shows a 360 degree panoramic image taken at the center of the laboratory.

In this experiment, the robot had the task to find tables as reliably as possible. The underlying idea of this experiment is that insufficient perception routines and sensor capabilities can be compensated by integration of multiple sensor modalities and active focusing of sensor devices on suspected objects. This experiment demonstrates the feasibility of proactive perception on the planning level and shows how multi-sensor integration takes place at the symbol layer.

The implemented perception routines utilize three sensor modalities, two of them need to be focused actively on a presumed target candidate. First, the robot detects possible table candidates using laser range scanners. These initial uncertain estimations are explored further by visual inspection. To permit simple but reliable visual perception, the robot actively focuses the camera on a corner of the table. If needed, it also approaches the assumed table with its mobile platform. To ensure the existence of the table, the robot touches the table top. Using this robot skill, additional information about the height of the table is obtained.

7.2.1. Symbolic implementation

Listing 7.1 shows two abstract HTN methods: the first and most abstract method is `(find_table_multimodal)`. It implements the pattern for deliberative replanning which is described in Section 6.3.3. The first time the planner is invoked, this method is decomposed into the atomic `(!find_table)` skill (see Section 5.2.5) and a skill that causes replanning deliberatively. Replanning is necessary to enable the symbol layer to deal with the table candidates that are newly introduced by this perception. The second decomposition process provides a plan to confirm or dismiss all initial table candidates. To do so, the `(find_table_multimodal)` method is reduced to the second method `(confirm_all_table_candidates)` which recursively process all table candidates.

The actual confirmation of the table candidates is done by the HTN method that is shown in Listing 7.2. This method has three different decomposition according to its progression and the results of the physical execution of the perception routines:

1. (line 34 et seq.) the considered table candidate will be confirmed visually by invoking the `(!confirm camera ?table)` skill. If necessary, the robot will approach the table.

2. (line 45 et seq.) Haptic confirmation of the table top. If this is successful (i.e. if no replanning is caused by the execution of the `(!confirm hand ?table)` skill), the symbolic world description will be updated.

3. (line 59 et seq.) If the considered table candidate cannot be confirmed in both sensor modalities, all bookkeeping literals will be deleted and a `(nosuccess ...)` marker will be added.

If a table candidate can not be confirmed, the `(confirm_all_table_candidates)` method will remove the candidate from the world state and proceeds with the remaining table candidates.

As all three physical perception routines are executed sequentially, multi-sensor integration takes place on the symbol level. Note that the symbol for the table itself puts no constraints on the type of sensor and actuator modalities that can be used to perceive and manipulate it.

The outcome of all primitive perception operators which are used in the HTN implementation of this task is uncertain and can not be predicted. They may fail and cause replanning by indicating an execution exception. Experiments showed that the recursive definition of this methods are sound in its ability to deal with exceptions due to failing perception at all stages of execution.

This robot experiment demonstrates the close coupling of the execution and the planning layer. The planning layer may invoke a control program and similarly a control program may cause the planner to generate a new plan. This experiment shows that the robustness of multi-modal robot perception regarding failure in unimodal low-level perception routines is increased, if suitable abstraction hierarchies for abstract tasks are

```
(:method (find_table_multimodal)
  ; initial estimation of table candidates
  ( (not (token)) )
  ( (!find_table) (!!add (token)) (!replan) )

  ; confirmation of estimated table candidates
  ( (token) )
  ( (confirm_all_table_candidates) (!!rm (token)) )
)

(:method (confirm_all_table_candidates)
  ; recursively confirm all table candidates
  ( (table_candidate ?table)
    (not (nosuccess confirm_table_multimodal ?table))
  )
  ( (confirm_table_multimodal ?table)
    (confirm_all_table_candidates)
  )

  ; remove table candidate if it could not be confirmed
  ( (table_candidate ?table)
    (nosuccess confirm_table_multimodal ?table)
  )
  ( (!!rm (table_candidate ?table))
    (confirm_all_table_candidates)
  )

  ; exit condition: no more table candidates to be confirmed
  ( )
  ( (clean) )
)
```

Listing 7.1: Listing of the HTN definition for proactive multi-modal table detection.

```

(:method (confirm_table_multimodal ?table)
  ; visual confirmation and recursive call
  ( (not (nosuccess confirm_table_multimodal ?table))
    (not (nosuccess confirm camera ?table))
    (not (nosuccess confirm hand ?table))
    (not (confirmed ?table))
    (use camera) (use mobile) (use ptu)
  )
  ( (approach ?table) (!focus ptu ?table)
    (!confirm camera ?table) (confirm_table_multimodal ?table)
  )

  ; haptic confirmation and change symbolic representation
  ( (not (nosuccess confirm_table_multimodal ?table))
    (not (nosuccess confirm camera ?table))
    (not (nosuccess confirm hand ?table))
    (confirmed ?table) (near robot ?table)
    (use arm) (use hand)
    (in robot ?room)
  )
  ( (!reach_for ?table) (!confirm hand ?table)
    (!reach_for park)
    (!!add (table ?table)) (!!add (in ?table ?room))
    (!!rm (table_candidate ?table))
  )

  ; can not confirm table
  ( )
  ( (clean) (!!add (nosuccess find_table_multimodal ?table))
  )
)
```

Listing 7.2: Listing of the HTN definition for proactive multi-modal table detection (cont'd.)

used. It has also been demonstrated that results from multiple sensor modalities can be integrated on the symbol level to amodal representations of external objects.

7.2.2. Evaluation of laser-based table detection

The performance of the overall robot system in this task depends strongly on the performance of the low-level perception routines. Especially the initial detection of possible table candidates turned out to be crucial. If no tables are detected initially, the robot will not be able to find a table at all. In turn, if too many table candidates have to be confirmed or dismissed, the complete perception process takes too much time. The experimental evaluation of the `(!find_table)` skill is described in the following. This has also been published in (Weser et al., 2008).

As described already in the chapter on implemented robot skills, the laser-based table detection algorithm takes only four-legged tables with a certain size into account. The tolerance of the angle among table legs and of the size of the tables are the only parameters that influence the quality of results regarding detection rate and number of false alarms. The influence of different parameter sets on the result of the table detector has been evaluated as follows.

Laser range scans at 114 positions in the laboratory have been recorded. The positions are chosen on a grid with 45 cm edge length and according to the circumstances of the environment. The positions of 5 tables have been annotated manually to provide ground truth. The laser-based table detection algorithm was tested at each position with different combinations of parameters: the tolerance in the size of the table candidates was chosen in the range of [0-200] mm and the tolerance in the angle among table legs was tested in the range of [0-0.16] rad. Figure 7.6 shows the number of correctly detected tables as a function of different parameter combinations. A tolerance of 120 mm in size and 0.1 rad in angle showed reasonably good results of 2 correct and 2.4 false tables detected per scan on average.

The detection of false alarms cannot be avoided using the implemented algorithm. As shown in Figure 5.5 (see p. 66), too many table leg candidates are found in the laser range scan. This is due to the similarity of table legs with chair legs, telephone cables etc. The search for rectangular patterns of such leg candidates as the underlying assumption for the recognition of tables may inevitably lead to incorrect table detections. The small number of 2 correctly detected tables out of 5 possible ones results from occlusion and measurement errors. However, these results are highly dependent on the environment and are hardly transferable to other experimental environments.

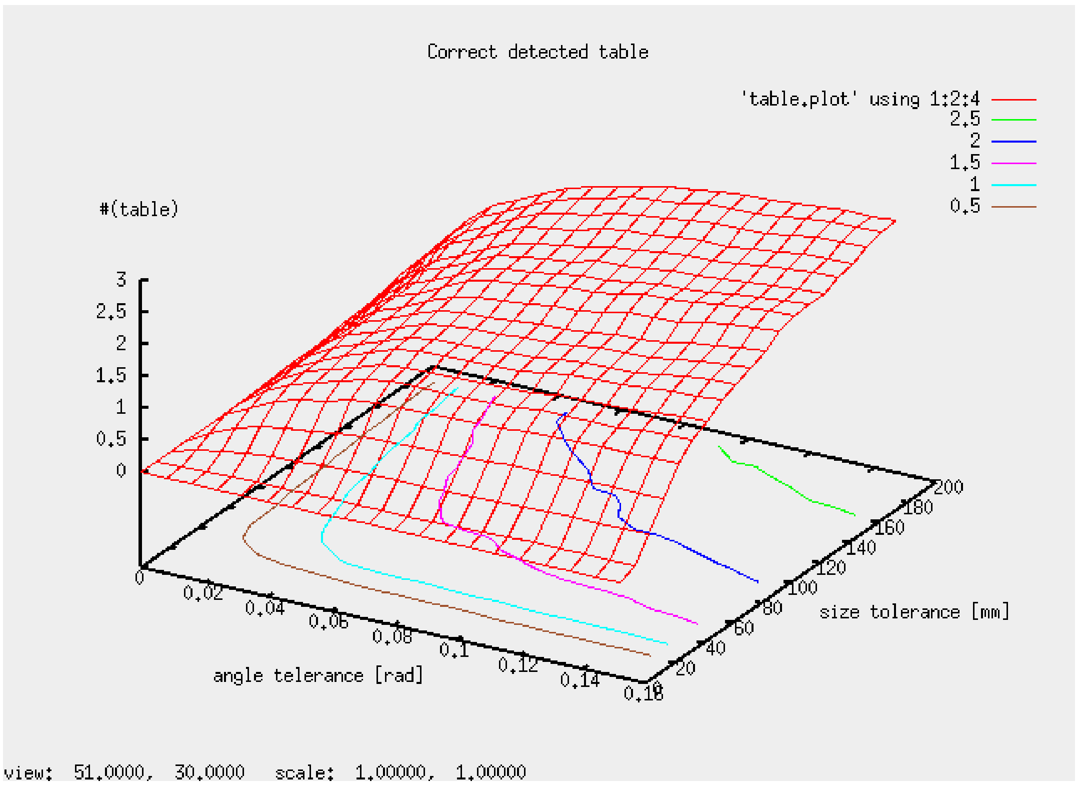


Figure 7.6.: Average number of correctly detected tables as a function of different parameters. A size tolerance of 120 mm and an angle tolerance of 0.1 rad showed reasonable results of 2 correct and 2.4 false tables detected per scan on average.

8 Summary and Conclusion

Once we accept our limits,
we go beyond them.
Albert Einstein, 1879-1955

In this work, an integrated software architecture for a mobile service robot has been developed. It comprises control modules of several hardware components, definitions and implementations of several atomic robot skills, and a state-of-the-art planning system which permits the execution of goal-directed action sequences with the robot system. The atomic robot skills have been combined on the symbol level to a hierarchy of abstract robot tasks. This symbolic representation provides the necessary domain knowledge for efficient planning.

The following section summarizes the main contributions of this thesis. Section 8.3 points out some limitations of the proposed approach. The thesis concludes with an outlook on future research directions.

8.1. Thesis summary

The main questions addressed in this thesis are:

a) Which level of abstraction of atomic robot skills effectively supports their use as primitive operators in AI-based planning systems as well as their implementation in control programs for embodied robots?

b) How can the integration of different sensor and actuator modalities substantially improve robot actions?

c) How should a hierarchy of robot skills be defined and implemented such that HTN planning methods can beneficially be used without major modifications to extend the robot's capabilities to achieve complex tasks?

A critical review of the literature on plan-based robot control identified the abstraction level of the basic building blocks for robot control, the atomic robot skills, as a crucial factor for the success of a robot control architecture. Based on this insight, the requirements for robot skills to be used as atomic operators for symbolic planning have been discussed in detail, which led to several requirements that have to be considered when designing a layered robot architecture. These requirements constitute a general guideline for the design and development of atomic robot skills in the context of layered robot architectures and provide an answer to question a).

A library of atomic robot skills has been defined and implemented that takes the requirements stated above into account. It has been shown that most robot behaviors can gain from multi-modal integration on the skill level. As usual, almost every robot action is guided by sensors. In addition, many skills that intend to perceive the external world make use of robot actions to focus the sensors deliberately on objects of interest. The reversal of the dependency of action and perception is prevalent in most implemented perception skills and has not been explored before to this extent. Proactive perception has so far only been applied to small application areas. Such application areas, in which this idea has proven to be beneficial, are reviewed in Chapter 3. The strategy to use physical robot action to support perception provides a substantial answer to the question b) stated above.

To answer question c), the robot has been enabled to achieve complex tasks by endowing it with state-of-the-art AI planning capabilities based on HTN planning. The problem of overcoming the closed-world assumption underlying every symbolic planning process has been solved by introducing deliberate replanning as part of the plan for complex tasks. The results of these investigations show that an AI-planning system can be applied to the domain of embodied service robots without major modifications. Furthermore, it was shown that a tight interaction of planning and execution layer effectively increases the reliability of robot behavior in the case of unsuccessful execution of atomic actions. In such cases, the deliberative component of the robot architecture attempts to find an alternative solution for the given task.

The described layered planning and control architecture has been implemented on the TAMS service robot. In a number of experiments, it has been demonstrated that the robot was substantially improved by the integration of the planning layer. Previously to this integration, it was not possible to request new tasks without modifying the system on the implementation level. Furthermore, the reorganization of fragments of control programs into atomic robot skills provides a strong basis for future research on plan-based robot control.

8.2. Aspects of multimodality in plan-based robot control

A common result of all parts of the work presented in this thesis is that isolated robot devices cannot achieve expedient robot behavior. Only the combination of multiple modalities led to the achievements described in this work. Multi-modal interactions constitute a key aspect in robot control at all levels of abstraction.

At the control level, different sensors are integrated to provide reliable information about the external world or to guide robot actuators. In turn, robot actuators are used to support perception. Several examples of such multi-modal integration have been given in the description of control programs in Chapter 5. In contrast to many other works in which perception is treated as an autonomous bottom-up process, perception is almost entirely initiated actively in this work. Integration of sensor data is not an ambient process that may or may not come up with useful information, e.g. interactions in different sensor data streams. It rather acts as a conscious search for previously expected patterns in the sensor data and thus supports or dismiss assumptions about the world. The assumptions that lead to the expectations for the results of multi-modal integration of actuator control and sensor data processing are given by the symbolic representation of the world which is ultimately provided by the developer of the system.

At the symbol level, primitive robot skills which deploy different modalities are combined to achieve complex behavior that would not be possible using only one modality. Atomic robot skills can usually be assigned clearly to single modalities. Examples are the `approach` of an object with the mobile platform or to `reach_for` an object with the manipulator. Although sensors are involved in such actions, the purpose of all involved modalities is clear. While some skills with focus on perception integrate different sensor modalities at low level (e.g. the `find_door` skill), most of them are tied to a sensor modality by instantiation of a variable that specifies the modality to be used.

The assignment of robot actions to modalities becomes more complex if more abstract skills, i.e. composed methods from the HTN tree, are considered. The method to `place_down` an object, for example, involves the robot hand, the robot arm, and possibly a mobile approach to the target position. This task may be referred to as *multi-modal*. However, there is no clear boundary in the transition from unimodal robot skills to multi-modal abstract tasks. Furthermore, tasks may be defined completely without consideration of the modalities being used. An example is a `find` task that uses modalities depending on what is available on the particular executing robot system.

Some tasks used in the planning process are of purely symbolic nature. All bookkeeping methods, i.e. to add or to delete a literal in the world state, as well as the skill that causes the robot to `replan` do not involve any physical modality. Adding and deleting of literals to or from the world state is similar to remembering and forgetting information at an abstract level. In this context, (symbolic) knowledge may be seen as a modality as well.

To sum up, multi-modal interaction plays a major role at every abstraction level in robot control. On a low abstraction level, e.g. in the control programs of robot skills, the dominant interaction is the integration of sensory data while at higher abstraction levels the (sequential) combination of different modalities is more prominent. Moreover, the common symbolic representation of the world and the deliberative component constitutes an indirect but omnipresent link that is shared by all modalities.

8.3. Limitations of the current architecture

The pragmatic approach to relate symbolic and continuous descriptions of objects using shared data structures in the implementation limits the perceptual capabilities concerning dynamic objects. While constrained dynamic aspects of objects, such as the state of a door, can be sufficiently modeled, moving objects can hardly be represented on the symbol level. The investigations described in this work focus on the deliberative aspect of perception. This prevents the integration of permanent perceptual processes to be integrated into the overall architecture. Such permanent perception services have to be implemented separately from the planning and control architecture. Their results have to be stored in the object structures which are used by the planner. This, in turn, introduces concurrent access on shared data and has to be protected somehow. The self-localization service of the robot is an example of ambient perception that runs independently of the general robot control.

Other limitations of the current architecture result form the particular implementations of the two layers. This limitations may be easily corrected due to the clear modular structure of the proposed layered architecture, still they will be described in the following.

8.3.1. Limitations of the symbol layer

Limitations that are introduced by the planner are the inability to handle the temporal aspects and concurrent execution of robot skills. Concurrent processes have to be implemented on the skill level, which puts strong constraints on their abstraction level. However, concurrent execution on the symbolic level is rarely intended and may be dispensable. Temporal aspects, in contrast, are important factors for robot planning. Many complex tasks require the robot to wait for responses of its surroundings. Small periods of time, e.g. waiting for a response after knocking on a door, may be accounted for on the skill level. Longer time spans, e.g. waiting for the dishwasher, have to be regarded on the planning level. A robot should be able to bridge such time gaps reasonably, which is only possible if it accounts for temporal aspects on planning level. Temporal aspects beyond the ordering sequence of primitive operators can hardly be represented with acceptable effort using JSHOP2 as the deliberative planning component.

A rich continuous representation of physical properties is required for reliable recognition and manipulation of external objects. The symbolic representation, in contrast, provides only highly abstracted information that is required to disambiguate objects on the planning level and in instructions given by humans. The decision of what needs to be represented in symbolic terms is a matter of design. Autonomous generation of symbolic properties from continuous representations would be desirable. For example, a user may request the *big* cup on the table. Currently, the system is not able to resolve situations in which information that is represented in the discrete domain needs to be accessed on symbol level.

8.3.2. Limitations of atomic robot skills

The focus of this thesis is the integration of several robot abilities into one coherent system that performs as a whole. Due to the number of involved behaviors, only little energy could be spent on the optimization of single robot skills. Some of them can be improved if investigated in isolation. For example, the simple grab which is currently used to pick-up any object may be substituted by precision grasps for small objects and by power grasps for heavy objects. A candidate implementation of elaborated grasping skills is the library which has been developed in (Baier-Löwenstein, 2008). The substitution of robot control programs causes only minor changes of the symbolic description of robot skills due to the layered structure of the proposed system.

8.4. Discussion on further research directions

To increase the perceptual and manipulative capabilities of the robot, it is advisable to include more sensors and actuators in the system. For example, the additional installation of a laser range scanner on the manipulator is currently under investigation. This will enable the robot to actively explore its environment by directing the laser beams into the direction of interest. Preliminary results of this research suggest that a significant increase in the accuracy of 3D perception of the robot's environment will be achieved. The hybrid architecture which has been developed in the present work provides the opportunity to seamlessly integrate additional perception skills into the robot system without the need for profound changes of the existing parts.

A scientifically challenging issue is the integration of learning mechanisms into the proposed system architecture. Machine learning techniques are in an advanced stage if applied to certain applications in isolation (see Section 3.5). These techniques may be integrated on the skill level to improve the control programs of certain robot skills. The clear definition of atomic robot skills provides a testing environment for learning algorithms on the skill level. Robot control programs can be substituted easily without changing the hierarchical representation of the planning domain.

The implemented library of robot skills provides a testbed for research on the autonomous learning of domain knowledge on the symbol level. While the learning of abstraction hierarchies for HTN planning has been gaining more and more interest recently (Nejati et al., 2006), it has not been applied to embodied robot platforms so far. This area of research provides the opportunity to increase the autonomy of embodied robots beyond what has been shown in this thesis.

Another line of research that arises from the results of this thesis is to investigate the properties of different AI planners and their application to the robotic domain. Since the basic operators for symbolic planning are represented in a language which is similar to PDDL, it is possible to substitute JSHOP2 as the deliberative component of the robot with other symbolic AI planners with minimal effort. Investigations on planning systems that inherently take temporal aspects of the world into account appear to be promising.

Replanning, which needs to be done in all cases of execution failure, may be avoided if not only the resulting sequence of primitive operators is considered as solution for a problem. If the complete hierarchy of instantiated HTN methods down to the primitive operators are stored as a plan, it would be possible to alter only parts of the hierarchy in case of failure of a particular action. This idea has recently been termed *failing upwards* and may increase the performance of replanning significantly. The current implementation of the robot architecture does not consider and store the instantiated HTN hierarchy as a part of the plan for a problem instance. Major modifications are needed to allow the implementation of failing upwards in the case of unsuccessful plan execution.

Last but not least, it needs to be shown whether the described robot system scales up to large real-world scenarios. To define and handle complex large-scale planning domains, elaborated editors and tools for consistency checking are needed.

A An EM-like Algorithm to Match 2D Shapes to Laser Range Data

The algorithm described below has been developed to estimate the two dimensional location of rigid objects based on laser range data. However, as the source of sensory data is not relevant for the algorithm, it may be applied to other data as well.

Preconditions for the proposed algorithm are a set of 2D data points and the 2D shape of the object which serves as a template of the object's appearance within the data. The algorithm resembles the iterative approach for maximum-likelihood estimation by (Dempster et al., 1977). It proceeds in two steps which are repeatedly executed until no further improvement will be expected. In the first step, a subset of data points is chosen that is assumed to originate from the object of interest. The second step adjusts the estimated position of the object according to the chosen data points. Both steps will be iterated until the change of the estimated object position becomes marginally, which indicates that a local minimum in the search space is reached. Algorithm 1 shows this procedure in more detail.

Adjusting the position of the object according to the data points (line 4) states the main difficulty in this algorithm. If the object would be restricted to a straight line, one could apply regression analysis to get an optimal estimation given the set of data points. However, to keep the algorithm as general as possible, the only restriction for the considered object is that it must be possible to compute the point s_i on the object's shape that is the closest to a given reference point p_i. Figure A.1 shows the lines defined by reference points and the corresponding points on the shape of a corner.

Algorithm 2 describes the procedure of changing the objects position according to a set of data points that is assumed to belong to the object. To determine the best

Algorithm 1 Iterative Algorithm to Match a Shape to a Subset of 2D Laser Range Data

```
Require: $P$: set of 2D data points; $S$: shape of object to be localized
1: repeat
2:     determine set of data points used in this iteration: $U \Leftarrow \{p_i \in P | dist(p_i, S) < \epsilon_p\}$
3:     store position of the object before transformation: $c = center(S)$
4:     adjust position of the object to the data points using Algorithm 2
5: until $dist(c, center(S)) < \epsilon_s$
```

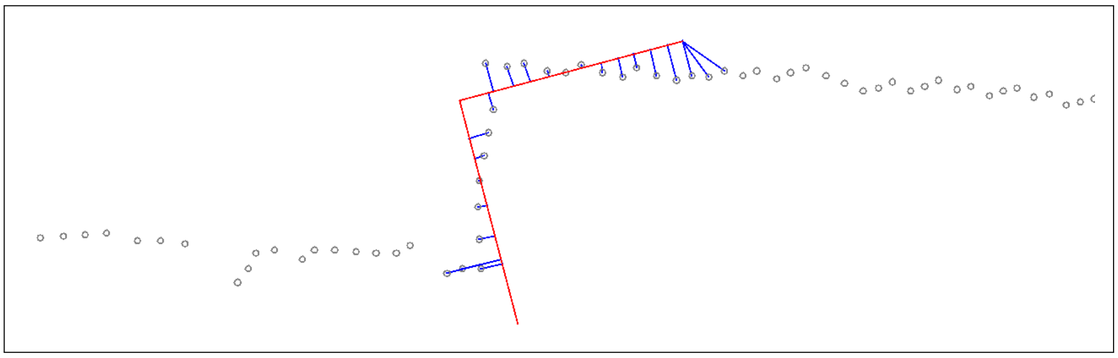


Figure A.1.: A simple corner shape and reference points obtained by laser range scans. The blue lines visualize which reference points belong to a point on the shape.

Algorithm 2 Algorithm to Match a Shape to a Set of 2D Points

```
Require: $U$: set of 2D data points; $S$: shape of object to be localized
1: init shift and rotation of object to zero: $t = 0$; $\varphi = 0$
2: for all point $p_i \in U$ do
3:     compute point on the shape with minimum distance to reference point:
       $s_i \Leftarrow near(S, p_i)$
4:     translate this $p_i$ and $s_i$ into the object's frame:
       $p'_i \Leftarrow p_i - center(S)$ and $v'_i \Leftarrow v_i - center(S)$
5:     record rotational part according to Equation 9.1: $\varphi = \varphi + rot(s'_i, p'_i)$
6:     record translational part according to Equation 9.2: $t = t + trans(s'_i, p'_i)$
7: end for
8: translate and rotate shape $S$ according to $t$ and $\varphi$
```

transformation for the object according to this points, each pair of a data point p_i and the corresponding shape point s_i is translated into the coordinate frame with the origin at the center of the object. The lines that are defined by corresponding point pairs p'_i and s'_i in object coordinates are split up into rotational and translational parts according to

$$rot(s'_i, p'_i) = atan(p'_{iy}, p'_{ix}) - atan(s'_{iy}, s'_{ix}) \tag{A.1}$$

and

$$trans(s'_i, p'_i) = \sqrt{(p'_{ix})^2 + (p'_{iy})^2} - \sqrt{(s'_{ix})^2 + (s'_{iy})^2} \tag{A.2}$$

where p'_{ix} and p'_{ix} are the the x- and y-components of the point p'_i in object coordinates. Similarly, s'_{ix} and s'_{iy} are the x- and y-components of the point s'_i. The average of the rotational and translational parts resulting from all considered data points determines the transformation that is finally applied to the shape (line 8).

The iterative process described in Algorithm 1 together with the estimation of a shape's location according to a set of data points (Algorithm 2) can in principle be used to locate arbitrary objects in laser range data. However, the algorithm depends on the thresholds ϵ_p and ϵ_s (line 2 and 5 of Algorithm 1) and may be trapped into local minima in the search space. Furthermore, it is not guaranteed that the algorithm converges at all. In adverse situations, the algorithm may oscillate and therefore not provide any solution.

The first alleged draw-back concerning local minima in the search space is actually wanted in the aspired application. For example, the laser range scan may be recorded in a room with two or more doors and the shape to be located is a door shape. Since the algorithm can provide only one location, it is wanted that the result is not the global best match of the shape to the laser range data but the local best match near an initial assumption of the location of the door.

B Symbolic Representaion of the TASER Domain

The following listing shows a problem description that describes the complete experimental environment as intial state.

```
(defproblem problem taser
  (
    ; available devices
    (use arm) (use ptu) (use camera) (use hand)
    (use laser) (use mobile)

    ; situation of the robot
    (manipulator park)
    ; or (manipulator transport) (have cup4)
    (in robot lab)
    ; maybe (near robot door3)

    ; rooms and doors
    (room lab) (room corridor) (room kitchen)
    (room studio) (room elevator)
    (door door1) (connect door1 lab corridor)
    (door door2) (connect door2 lab corridor)
    (door door3) (connect door3 elevator corridor)
    (door door4) (connect door4 kitchen corridor)
    (door door5) (connect door5 studio corridor)
    (closed door1)
    (closed door2)
```

Listing B.1: Problem definition: this part represents the initial state.

```
      ; tables
      (table table1) (in table1 lab)
      (table table2) (in table2 lab)
      (table table3) (in table3 lab)
      (table table4) (in table4 kitchen)
      (table table5) (in table5 studio)
      (table table6) (in table6 lab)

      ; buckets and cups
      (bucket bucket1) (in bucket1 lab)
      (cup cup1) (on cup1 table1) (color cup1 white)
      (cup cup2) (on cup1 table2) (color cup2 blue)
      (cup cup3) (on cup1 table5) (color cup3 red)
    )

    (
      ; list of problems to be carried out
      (delivery_service bucket1 elevator)
      (delivery_service cup1 kitchen)
    )
)
```

Listing: Problem definition (cont'd.): the bottom part shows the list of abstract tasks to be solved.

Bibliography

Acuity Research, I. (2000). Computer controlled pan-tilt unit: Models ptu-46-17.5 and ptu-46-70. www.acuitylaser.com/pdf/pan-tilt-unit-data-sheet.pdf.

Anderson, J. (1993). *Rules of the Mind.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Arkin, R. C. (1987). *Towards cosmopolitan robots: intelligent navigation in extended man-made environments.* PhD thesis, University of Massachusetts. Director-Edward M. Riseman.

Arkin, R. C. (1998). *Behavior-based Robotics.* MIT Press, 2 edition.

Asfour, T., Regenstein, K., Azad, P., Schrder, J., Bierbaum, A., Vahrenkamp, N., and Dillmann, R. (2006). ARMAR-III: An integrated humanoid platform for sensory-motor control. In *IEEE-RAS International Conference on Humanoid Robots*, pages 169–175.

Baier, T., Westhoff, D., Hüser, M., and Zhang, J. (2006). A flexible software architecture for multi-modal service robots. In *Multiconference on Computational Engineering in Systems Applications (CESA)*, pages 587–592, Beijing, China.

Baier-Löwenstein, T. (2008). *Lernen der Handhabung von Alltagsgegenständen im Kontext eines Service-Roboters.* PhD thesis, University of Hamburg, Department of Informatics.

Beaudry, E., Kabanza, F., and Michaud, F. (2005). Planning for a mobile robot to attend a conference. In *Canadian Conference on Artificial Intelligence*, pages 48–52.

Beetz, M. (2001). Structured reactive controllers. *Autonomous Agents and Multi-Agent Systems*, 4(1-2):25–55.

Beetz, M., Burgard, W., Fox, D., and Cremers, A. B. (1999). Integrating active localization into high-level robot control systems. *Journal of Robotics and Autonomous Systems*, page forthcoming.

Beetz, M., Stulp, F., Radig, B., Bandouch, J., Blodow, N., Dolha, M., Fedrizzi, A., Jain, D., Klank, U., Kresse, I., Maldonado, A., Marton, Z., Mösenlechner, L., Ruiz, F., Rusu, R. B., and Tenorth, M. (2008). The assistive kitchen — a demonstration scenario for cognitive technical systems. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN).* Invited paper.

Benjamin, D. P., Lonsdale, D., and Lyons, D. (2004). Designing a robot cognitive architecture with concurrency and active perception. In *AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics.*

Benjamin, P., Lonsdale, D., and Lyons, D. (2006). Embodying a cognitive model in a mobile robot. In *SPIE Conference on Intelligent Robots and Computer Vision, Boston.*

Bennewitz, M. (2004). *Mobile Robot Navigation in Dynamic Environments.* PhD thesis, University of Freiburg, Department of Computer Science.

Bistry, H., Pöhlsen, S., Westhoff, D., and Zhang, J. (2007a). Development of a smart laser range finder for an autonomous service robot. In *IEEE International Conference on Integration Technology (ICIT)*, pages 799–804.

Bistry, H., Westhoff, D., and Zhang, J. (2007b). A smart interface-unit for the integration of pre-processed laser range measurements into robotic systems and sensor networks. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 358–363.

Biundo, S. and Schattenberg, B. (2001). From abstract crisis to concrete relief—a preliminary report on combining state abstraction and htn planning. In *European Conference on Planning (ECP)*, pages 157–168.

Bonet, B. and Geffner, H. (2001). Heuristic search planner 2.0. *AI Magazine*, 22(3):77–80.

Borenstein, J., Everett, H. R., and Feng, L. (1996). *Navigating Mobile Robots: Systems and Techniques.* A. K. Peters, Ltd., Natick, MA, USA.

Boudol, G. (1989). Atomic actions. In *European Association for Theoretical Computer Science (EATCS)*, volume 38, pages 136–144.

Braitenberg, V. (1984). *Vehicles. Experiments in Synthetic Psychology.* MIT Press, Cambridge, Mass.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. In *International Conference on Robotics and Automation (ICRA)*, volume RA-2, pages 14–23.

Brooks, R. A. and Stein, L. A. (1994). Building brains for bodies. *Autonomeous Robots*, 1(1):7–25.

Bühler, M., Koditschek, D. E., and Kindlmann, P. J. (1990). A simple juggling robot: Theory and experimentation. In *International Symposium on Experimental Robotics I*, pages 35–73, London, UK. Springer-Verlag.

Bylander, T. (1991). Complexity results for planning. In *International Joint Conference on Artificial Intelligence (IJCAI).*

Cangelosi, A. (2006). The grounding and sharing of symbols. *Pragmatics and Cognition*, 14(2):275–285.

Carbonell, J., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., and Veloso, M. (1991). Prodigy: an integrated architecture for planning and learning. *SIGART Bull.*, 2(4):51–55.

Chahl, J. S. and Srinivasan, M. V. (1997). Reflective surfaces for panoramic imaging. *Applied Optics (Optical Society of America)*, 36 (31):8275–8285.

Chestnutt, J., Michel, P., Kuffner, James, and Kanade, T. (2007). Locomotion among dynamic obstacles for the honda asimo. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2572–2573.

Chung, J., Ryu, B.-S., and Yang, H. S. (1998). Integrated control architecture based on behavior and plan for mobile robot navigation. *Robotica*, 16:387–399.

Connell, J. H. and Mahadevan, S., editors (1993). *Robot Learning*. Kluwer Academic Publishers, Norwell, MA, USA.

Coradeschi, S. and Saffiotti, A. (2000). Anchoring symbols to sensor data: Preliminary report. In *AAAI Conference on Artificial Intelligence*, pages 129–135.

Coradeschi, S. and Saffiotti, A. (2003). An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96. Special issue on perceptual anchoring.

Cox, I. J. and Wilfong, G. T. (1990). *Autonomous robot vehicles*. Springer-Verlag New York, Inc., New York, NY, USA.

Davison, A. J. (1998). *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford.

Dempster, A., N.M., L., and Rubin, D. (1977). Maximum-likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*.

Do, M. and Kambhampati, S. (2003). Sapa: A scalable multi-objective metric temporal planner. *Journal of Artificial Intelligence Research*, 20:155–194.

Ferch, M. C. (2001). *Lernen von Montagestrategien in einer verteilten Multiroboterumgebung*. PhD thesis, Universitaet Bielefeld.

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application oftheorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208.

Forlizzi, J. and DiSalvo, C. (2006). Service robots in the domestic environment: a study of the roomba vacuum in the home. In *ACM SIGCHI/SIGART Conference on Human-Robot interaction (HRI)*, pages 258–265.

Fox, D., Burgard, W., and Thrun, S. (1998). Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207.

Francis, A. and Ram, A. (1993). The utility problem in case-based reasoning. In *AAAI Workshop on Case Based Reasoning*, pages 160–167.

Fu, K. S., Gonzalez, R. C., and Lee, C. S. G. (1987). *Robotics: Control, Sensing, Vision and Intelligence*. McGraw-Hill Book Company.

Fung, W., Leung, Y., and Chow, M. (2003). Development of a hospital service robot for transporting task. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pages 628–633.

Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *International Conference on Advanced Robotics (ICAR)*, pages 317–323.

Ghallab, M., Alami, R., Hertzberg, J., Gini, M., Fox, M., Williams, B., Schattenberg, B., Borrajo, D., Doherty, P., Morina, J. M., Sanchis, A., Fabiani, P., and Pollack, M. (2006). A roadmap for research in robot planning.

Ghallab, M., Nationale, E., Aeronautiques, C., Isi, C. K., Penberthy, S., Smith, D. E., Sun, Y., and Weld, D. (1998). PDDL - the planning domain definition language.

Ghallab, M., Nau, D., and Traverso, P. (2004a). *Automated Planning, Theorie and Practice*, chapter Hierarchical Task Network Planning, pages 229–262. Elsevier Science.

Ghallab, M., Nau, D., and Traverso, P. (2004b). *Automated Planning Theory and Practice*. Elsevier Science.

Gupta, S. (1999). Sheet metal bending operation planning: Using virtual node generation to improve search efficiency. *Journal of Manufacturing Systems*, 18(2):127–139.

Hanebeck, U. D., Fischer, C., and Schmidt, G. (1997). Roman: A mobile robotic assistant for indoor service applications. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 518–525.

Harnad, S. (1990). The symbol grounding problem. *Physika D*, 42:335–346.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall.

Hoffmann, J. (2001). Ff: The fast-forward planning system. *AI Magazin*, 22:57–62.

Holzapfel, H., Neubig, D., and Waibel, A. (2008). A dialogue approach to learning object descriptions and semantic categories. *Robotics and Autonomous Systems*, 56(11):1004–1013.

Ilghami, O. (2006). Documantation for JSHOP2. Technical Report CS-TR-4694, University of Maryland, Department of Computer Science.

Ilghami, O. and Murdock, J. W. (2005). An extension to PDDL: Actions with embedded code calls. In *International Conference on Automated Planning & Scheduling (ICAPS)*, pages 63–68.

Ilghami, O. and Nau, D. S. (2003). A general approach to synthesize problem-specific planners. Technical report.

Kambhampati, S., Parker, E., and Lambrecht, E. (1997). Understanding and extending graphplan. In *European Conference on Planning*, pages 260–272.

Katz, D., Horrell, E., Yang, Y., Burns, B., Buckley, T., Grishkan, A., Zhylkovskyy, V., Brock, O., and Learned-Miller, E. (2006). The UMass mobile manipulator UMan: An experimental platform for autonomous mobile manipulation. In *IEEE Workshop on Manipulation for Human Environments.*

Kautz, H. and Selman, B. (1998). Blackbox: A new approach to the application of theorem proving to problem solving. In *Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98*, pages 58–60.

Kautz, H., Selman, B., and Hoffmann, J. (2004). Satplan: Planning as satisfiability. In *International Planning Competition at the International Conference on Automated Planning and Scheduling (ICAPS).*

Kemp, C., Edsinger, A., and Torres-Jara, E. (2007). Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):20–29.

Kennedy, W. G. and Trafton, J. G. (2007). Long-term symbolic learning. *Cognitive Systems Research*, 8:237–247.

Kieras, D. and Meyer, D. (1997). An overview of the epic architecture for cognition and performance with application to human-computer interaction. *Human–Computer Interaction*, 12:391–438.

Kilian, C. (2005). *Modern Control Technology.* Delmar, 3 edition.

Kim, D., Kang, J.-H., Hwang, C.-S., and Park, G.-T. (2004). Mobile robot for door opening in a house. In *Lecture Notes in Computer Science.* Springer Berlin / Heidelberg.

Kim, H. S., Jung, Y. C., and Hwang, Y. K. (2005). Taxonomy of atomic actions for home-service robots. *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, 9(2):114–120.

Klanke, S., Lebedev, D., Haschke, R., Steil, J., and Ritter, H. (2006). Dynamic path planning for a 7-dof robot arm. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3879–3884.

Kleinehagenbrock, M., Lang, S., Fritsch, J., Lmker, F., Fink, G. A., and Sagerer, G. (2002). Person tracking with a mobile robot based on multi-modal anchoring. In *IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, pages 423–429.

Klug, S., Lens, T., von Stryk, O., Klug, S., Möhl, B., and Karguth, A. (2008). Biologically inspired robot manipulator for new applications in automation engineering. In *Proceedings of Robotik, Munich.*

Kobilarov, M. and Sukhatme, G. (2006). People tracking and following with mobile robot using an omnidirectional camera and a laser. In *IEEE International Conference on Robotics and Automation*, pages 557–562.

Konidaris, G. (2008). Autonomous robot skill acquisition (thesis summary). In *Doctoral Symposium, National Conference on Artificial Intelligence (AAAI)*, pages 1855–1856.

Konolige, K., karen Myers, and Ruspini, E. (1997). The saphira architecture: A design for autonomy. *Experimental and Theoretical Artificial Intelligence (JETAI)*, 9:215–235.

Konolige, K., Myers, K., Ruspini, E., and Saffiotti, A. (1993). Flakey in action: The 1992 aaai robot competition. Technical Report 528, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025.

Kuhn, T. S. (1996). *The Structure of Scientific Revolutions.* University Of Chicago Press.

Laird, J. E. and Congdon, C. B. (2006). The soar users manual version 8.6.

Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., and Thompson, K. (1991). A design for the icarus architecture. *SIGART Bull.*, 2(4):104–109.

LaValle, S. (2006). *Planning Algorithms.* Cambridge University Press.

Levy, D. and Newborn, M. (1991). *How Computers Play Chess.* Computer Science Press, New York.

Li, Y. F., , He, B., Chen, S., and Bao, P. (2005). A view planning method incorporating self-termination for automated surface measurement. In *Measurement Science and Technology*, pages 1865–1877.

Lindstrom, M. Oreback, A. C. H. (2000). Berra: a research architecture for service robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3278–3283.

Lloyd, J. E. and Hayward, V. (1992). Multi-rccl user's guide. McGill University.

Lörken, C. and Hertzberg, J. (2008). Grounding planning operators by affordances. In *International Conference on Cognitive Systems (CogSys)*, pages 79–84.

Low, K. H., Leow, W. K., and Ang, Jr., M. H. (2002). A hybrid mobile robot architecture with integrated planning and control. In *International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 219–226.

Lowe, D. (1999). Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 2:1150–1157.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Luo, R. C. and Kay, M. G., editors (1995). *Multisensor integration and fusion for intelligent machines and systems*. Ablex Publishing Corp., Norwood, NJ, USA.

Lyons, D. M. and Arbib, M. A. (1989). A formal model of computation for sensory-based robotics. *IEEE Journal of Robotics and Automation*, 5(3):280–293.

McDermott, D. (1992). Robot planning. *AI Magazine*, 13:55–79.

Mei, C. and Rives, P. (2006). Calibration between a central catadioptric camera and a laser range finder for robotic applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 532–537.

Michaud, F. (2002). Emib - computational architecture based on emotion and motivation for intentional selection and configuration of behaviour-producing modules. *Cognitive Science Quaterly*, 3-4.

Michaud, F., Letourneau, D., Frechette, M., Beaudry, E., and Kabanza, F. (2006). Spartacus, scientific robot reporter. In *Mobile Robot Workshop American Association for Artificial Intelligence (AAAI)*.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Companies Inc.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2003). Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1151–1156.

Moore, A. W. (1990). *Efficient Memory-based Learning for Robot Control*. PhD thesis, Cambridge, UK.

Morriset, B. and Ghallab, M. (2002). Synthesis of supervision policies for robust sensory-motor behaviors. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Müller, A. (2008). *Transformational Planning for Autonomous Household Robots using Libraries of Robust and Flexible Plans.* PhD thesis, Technische Universität München.

Müller, A. and Beetz, M. (2007). Towards a plan library for household robots. In *Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices for Planning in Execution (ICAPS)*, Providence, USA.

Nau, D., Au, T. C., Ilghami, O., Kuter, U., Murdock, W., Wu, D., and Yaman, F. (2003). Shop2: An htn planning system. *Journal on Artificial Intelligence Research*, 20.

Nejati, N., Langley, P., and Konik, T. (2006). Learning hierarchical task networks by observation. In *International Conference on Machine Learning (ICML)*, pages 665–672.

Newell, A. and Simon, H. A. (1975). Computer science as empirical inquiry: Symbols and search.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence.* Springer, Berlin, Heidelberg.

Nishida, T., Takemura, Y., Fuchikawa, Y., and Kurogi (2006). Development of outdoor service robots. In *International Joint Conference SICE-ICASE*, page 2052 2057.

Orebaeck, A. (2004). *A Component framework for Autonomous Mobile Robots.* PhD thesis, University of Stockholm, Sweden.

Pope, A. R. and Lowe, D. G. (1996). Learning appearance models for object recognition. In *International Workshop on Object Representation in Computer Vision II (ECCV)*, pages 201–219.

Ratanaswasd, P., Gordon, S., and Dodd, W. (2005). Cognitive control for robot task execution. In *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, pages 440–445, Nashville, TN.

Roy, D. (2005). Semiotic schemas: A framework for grounding language in the action and perception. *Artificial Intelligence*, 167(1-2):170–205.

Rubrecht, S., Bouzid, R., Mechbal, N., Bellot, D., and Vergé, M. (2008). HAMMI: Intelligent Robot of Assistance to People with Motor Disabilities. In *International Conference on Control Application (CA)*, page a, Québec Canada.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Pearson Education.

Russo, M. and Jain, L. C. (2000). *Fuzzy Learning and Applications.* CRC Press, Inc., Boca Raton, FL, USA.

Saffiotti, A. (1994). Pick-up what? In Bäckström, C. and Sandewall, E., editors, *Current Trends in AI Planning*, pages 266–277. IOS Press.

Saffiotti, A. and LeBlanc, K. (2000). Active perceptual anchoring of robot behavior in a dynamic environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3796–3802.

Scheibe, K. and Scheele, M. (2004). Multi-sensor panorama fusion and visualization. Communication and information technology research technical report 141, CITR, The University of Auckland, New Zealand.

Scherer, T. (2004). *A Mobile Service Robot for Automisation of Sample Taking and Sample Management in a Biotechnological Pilot Laboratory.* PhD thesis, University of Bielefeld.

Schönherr, F. and Hertzberg, J. (2002). The dd &p robot control architecture. In *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg.

Sedgewick, R. (2001). *Algorithms in C++ Part 5: Graph Algorithms.* Addison-Wesley Professional, 3 edition.

Shieh, M., Hsieh, J., and Cheng, C. (2004). Design of an intelligent hospital service robot and its applications. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, page 4377 4382.

Simmons, R. (1992). Concurrent planning and execution for autonomous robots. *Control Systems Magazine, IEEE*, 12(1):46–50.

Simmons, R. and Apfelbaum, D. (1998). A task description language for robot control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1931–1937.

Simmons, R., Goldberg, D., Goode, A., Montemerlo, M., Roy, N., Sellner, B., Urmson, C., Bugajska, M., Coblenz, M., Macmahon, M., Perzanowski, D., Horswill, I., Zubek, R., Kortenkamp, D., Wolfe, B., Milam, T., Inc, M., and Maxwell, B. (2003). Grace: An autonomous robot for the aaai robot challenge. *AI Magazine*, 24:51–72.

Simmons, R., Pecheur, C., and Srinivasan, G. (2000). Towards automatic verification of autonomous systems. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1410–1415.

Smith, S. J. J., Hebbar, K., Nau, D. S., and Minis, I. (1997). *Knowledge Intensive CAD*, chapter Integrating electrical and mechanical design and process planning, pages 269–288. Chapman and Hall.

Smith, S. J. J., Nau, D. S., and Throop, T. A. (1998). Computer bridge - a big win for ai planning. *AI Magazine*, 19(2):93–106.

Steels, L. (1999). The talking heads experiment. Words and meanings. Technical report, VUB, Brussels.

Steels, L. (2007). The symbol grounding problem is solved, so what's next? In De Vega, M., Glennberg, G., and Graesser, G., editors, *Symbols, embodiment and meaning.* Academic Press, New Haven.

Steels, L. and Kaplan, K. (2001). Aibos first words: The social leaming of language and meaning. In *Evolution of Communication 4.*

Steels, L. and Spranger, M. (2008). The robot in the mirror. *Connection Science, Taylor & Francis*, 20 (4):337–358.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning).* The MIT Press.

Sutton, R. S., Precup, D., and Singh, S. P. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Tate, A. (1977). Generating project networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 888–893.

Tate, A., Drabble, B., and Kirby, R. (1994). *O-Plan2: An Architecture for Command, Planning and Control.* Morgan Kaufmann.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hähnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). MINERVA: A second generation mobile tour-guide robot.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141.

Townsend, W. T. (2000). The BarrettHand grasper – programmably flexible part handling and assembly. *Industrial Robot: An International Journal of Applied Signal Processing*, 27, no. 3:181 – 188.

Tsai, R. Y. (1986). An efficient and accurate camera calibration technique for 3d machine vision. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 364–374, Miami Beach, Fla. IEEE, IEEE Computer Society Press.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings London Mathematical Society*, 2(42):230–265.

von Collani, Y. O. (2001). *Repraesentation und Generalisierung von diskreten Ereignisablaeufen in Abhaengigkeit von Multisensormustern.* PhD thesis, University of Bielefeld.

Waibel, M. (2007). *Eviolution of Cooperation in Artificial Ants.* PhD thesis, Technische Universitaet Wien.

Wang, M., Ma, S., Li, B., Wang, Y., He, X., and Zhang, L. (2005). Task planning and behavior scheduling for a reconfigurable planetary robot system. In *International Conference on Mechatronics & Automation (ICMA)*, pages 729–734.

Wasson, G., Kortenkamp, D., and Huber, E. (1998). Integrating active perception with an autonomous robot architecture. In *International Conference on Autonomous Agents*, pages 325–331.

Wei, G., Wetzler, C., and von Puttkamer, E. (1994). Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 595–601.

Weng, J. (2004). A theory of developmental architecture. In *International Conference on Development and Learning (ICDL).*

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. (2001). Autonomous mental development by robots and animals. *Science, 291.*

Wermter, S., Weber, C., Elshaw, M., Gallese, V., and F., P. (2005). Grounding neural robot language in action. *Biomimetic Neural Learning for Intelligent Robots*, pages 162–181.

Weser, M., Jockel, S., and Zhang, J. (2008). Fuzzy multisensor fusion for autonomous proactive robot perception. In *World Congress on Computational Intelligence (WCCI)*, pages 2262–2267.

Weser, M., Westhoff, D., Hüser, M., and Zhang, J. (2006a). Multimodal people tracking and trajectory prediction based on learned generalized motion patterns. In *International Conference on Multisensor Fusion and Integration (MFI)*, pages 541–546.

Weser, M., Westhoff, D., Hüser, M., and Zhang, J. (2006b). Real-time fusion of multimodal tracking data and generalization of motion patterns for trajectory prediction. In *International Conference on Information Acquisition (ICIA)*, pages 786–791.

Weser, M. and Zhang, J. (2007). Proactive multimodal perception for feature based anchoring of complex objects. In *International Conference on Robotics and Biometrics (ROBIO)*, pages 1069–1074.

Weser, M. and Zhang, J. (2009). Autonomous planning for mobile manipulation services based on multi-level robot skills. In *IEEE International Conference on Robotics and Systems (IROS).*

Westhoff, D., Stanek, H., and Zhang, J. (2006). Distributed applications for robotic systems using roblet-technology. In *International Symposium on Robotics and Deutsche Fachtagung Robotik*, Munich, Germany. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik.

Wilkins, D. (1988). *Practical Planning: Extending the classical AI planning paradigm.* Morgan Kaufmann Publishers.

Wilkins, D. E. (1990). Can ai planners solve practical problems? *Computational Intelligence*, vol. 6, no. 4:232–246.

Zadeh, L. (1965). Fuzzy sets. *Information Control*, 8:338–353.

Zhang, J., Collani, Y. V., and Knoll, A. (1999). Interactive assembly by a two-arm robot agent. *Robotics and Autonomous Systems*, 29:91–100.

Zhang, J. and Knoll, A. (2003). A two-arm situated artificial communicator for human-robot cooperative assembly. *IEEE Transactions on Industrial Electronics*, 50(4):651–658.

Zhang, Q. (2004). Extrinsic calibration of a camera and laser range finder. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2301–2306.

Index