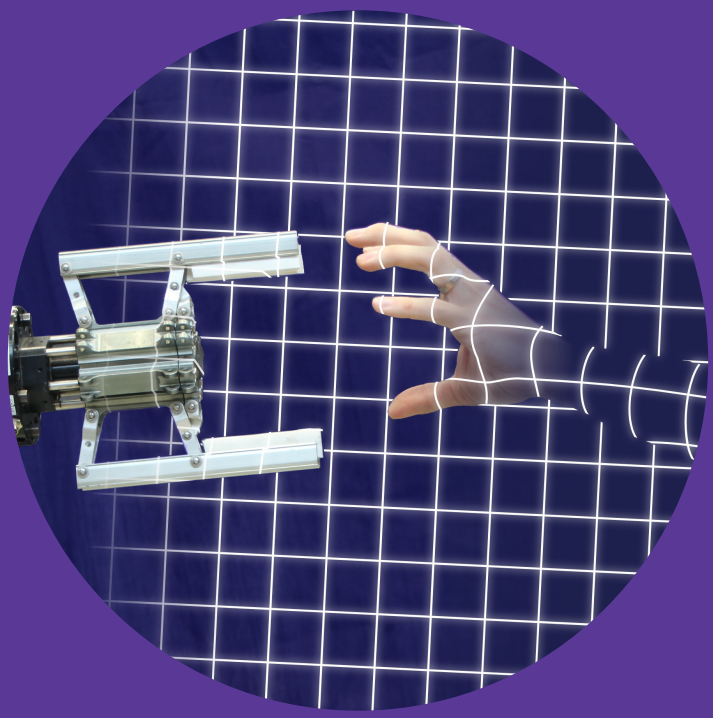
Department of Automation and Systems Technology

Building and sharing the cognition model of the environment with collaborative service robots

Sami Terho



Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 104/2011

Building and sharing the cognition model of the environment with collaborative service robots

Sami Terho

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the School of Electrical Engineering for public examination and debate in Auditorium AS1 at the Aalto University School of Electrical Engineering, TUAS Building (Espoo, Finland) on the 4th of November 2011 at 12 noon.

Aalto University
School of Electrical Engineering
Department of Automation and Systems Technology
Centre of Excellence of Generic Intellingent Machines

Supervisor
Professor Aarne Halme

Instructor
Professor Aarne Halme

Preliminary examiners
Professor Juha Röning, University of Oulu, Finland
Professor Ray Jarvis, Monash University, Melbourne, Australia

Opponents
Dr Monica Reggiani, Università degli Studi di Padova, Italy
Professor Juha Röning, University of Oulu, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 104/2011

ISBN 978-952-60-4332-6 (pdf)
ISBN 978-952-60-4331-9 (printed)
ISSN-L 1799-4934
ISSN 1799-4942 (pdf)
ISSN 1799-4934 (printed)

Unigrafia Oy
Helsinki 2011

Finland

The dissertation can be read at http://lib.tkk.fi/Diss/

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Sami Terho

Name of the doctoral dissertation
Building and sharing the cognition model of the environment with collaborative service robots

Publisher School of Electrical Engineering

Unit Department of Automation and Systems Technology

Series Aalto University publication series DOCTORAL DISSERTATIONS 104/2011

Field of research Automation technology

Manuscript submitted 24 August 2010 **Manuscript revised** 17 June 2011

Date of the defence 4 November 2011 **Language** English

☒ **Monograph** ☐ **Article dissertation (summary + original articles)**

Abstract

This research presents a shared cognition concept for building a cognition model collaboratively by robot and human. The proposed concept is based on an object-oriented approach that abstracts robot's perception, cognition, and knowledge to separate but interconnected elements. The shared cognition concept allows the human and robot share information related to objects in the environment for effective task execution and information exchange. The concept enables building a model using sensors, as well as inputting data from the user. User can utilize also robot's sensor data for inputting information. This utilizes the robot's perception of the environment in conjunction with human's cognitive understanding.

The proposed concept can be used for exchanging information on the objects, their classes, identities, appearances, locations, structures, and states. Representation of perceived entities is done through observed objects, robot's cognitive understanding of the objects is represented by real objects, and the knowledge of objects' classes by meta-objects. Observed objects are created with segmentation which can be autonomous or assisted by human. Real objects are used to transfer the actual object information between human and robot. Recognition of objects is done by matching real and observed objects. Uncertainties of recognition and localization are handled by probabilistic approach. The model enables learning from perceived information allowing the model to improve its performance as more data is gathered.

The cognition model can be used for collaborative task execution. Several aspects of the model are validated with two different platforms, WorkPartner service robot and Avant machine based semi-autonomous robot. Based on the tests, the proposed model is an effective mechanism for collaboratively exchanging spatial information between a robot and a human.

Keywords service robotics, shared cognition, robot cognition, robot perception

ISBN (printed) 978-952-60-4331-9 **ISBN (pdf)** 978-952-60-4332-6

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Espoo **Location of printing** Helsinki **Year** 2011

Pages 149 **The dissertation can be read at** http://lib.tkk.fi/Diss/

A! Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Sami Terho

Väitöskirjan nimi
Kognitiomallin rakentaminen ja jakaminen yhteistyössä toimivan palvelurobotin kanssa

Julkaisija Sähkötekniikan korkeakoulu

Yksikkö Automaatio- ja systeemitekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 104/2011

Tutkimusala Automaatiotekniikka

Käsikirjoituksen pvm 24.08.2010 **Korjatun käsikirjoituksen pvm** 17.06.2011

Väitöspäivä 04.11.2011 **Kieli** Englanti

☒ **Monografia** ☐ **Yhdistelmäväitöskirja (yhteenveto-osa + erillisartikkelit)**

Tiivistelmä

Tutkimuksessa esitetään jaetun kognition konsepti, jota käytetään robotin ja ihmisen yhteisen kognitiomallin muodostamiseen. Ehdotettu konsepti perustuu oliolähtöiseen lähestymistapaan, jolla abstraktoidaan robotin havainnointi, kognitio ja tietämys erillisiin mutta toisiinsa liittyviin elementteihin. Jaetun kognition konsepti mahdollistaa ihmisen ja robotin informaation jakamisen liittyen ympäristössä oleviin kohteisiin. Tällä mahdollistetaan tehokas tehtävien suoritus sekä tiedon jakaminen. Kognitiomalli voidaan rakentaa käyttäen robotin antureita sekä ihmisen syöttämää tietoa. Käyttäjä pystyy hyödyntämään robotin anturidataa informaation syöttämisen apuna. Tällä tavalla hyödynnetään robotin ympäristön havainnointi yhdessä ihmisen kognitiivisen ymmärryksen kanssa.

Ehdotettua konseptia voidaan käyttää tiedonvaihtoon liittyen kohteisiin, niiden luokkiin, identiteetteihin, ulkonäköön, sijainteihin, rakenteisiin ja tiloihin. Havainnot esitetään havainto-olioina, robotin kognitiivinen ymmärrys todellisina olioina sekä robotin tietämys kohteiden luokista esitetään meta-olioina. Havainto-oliot muodostetaan segmentoinnilla, joka voidaan tehdä autonomisesti tai avustaen. Kohteisiin liittyvä tieto siirretään ihmisen ja robotin välillä todellisten olioiden avulla. Kohteiden tunnistus tapahtuu yhdistämällä todellisia ja havainto-olioita. Tunnistuksen ja paikantamisen epävarmuudet käsitellään todennäköisyysjakaumamallien avulla. Malli mahdollistaa oppimisen havainnoista, jolloin mallin toimintaa voidaan parantaa sitä mukaa kun tietoa kerääntyy.

Kognitiomallia voidaan hyödyntää tehtävien suoritukseen. Mallin osia on testattu kahdella eri alustalla, WorkPartner-palvelurobotilla sekä Avant-työkoneeseen perustuvalla puoliautonomisella robotilla. Testien perusteella esitetty malli on tehokas menetelmä yhteistyössä tapahtuvaan tiedonvaihtoon robotin ja ihmisen välillä.

Avainsanat palvelurobotiikka, jaettu kognitio, robotin kognitio, robotin havainnointi

ISBN (painettu) 978-952-60-4331-9 **ISBN (pdf)** 978-952-60-4332-6

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Espoo **Painopaikka** Helsinki **Vuosi** 2011

Sivumäärä 149 **Luettavissa verkossa osoitteessa** http://lib.tkk.fi/Diss/

Preface

This research was conducted in Centre of Excellence of Generic Intelligent Machines (GIM) at Department of Automation and Systems Technology at Aalto University. The research and the related exchange periods were partly funded by Academy of Finland, Tekes, and FIMA ry.

I wish to express my gratitude to my supervisor, Professor Aarne Halme. He suggested me this exciting and challenging research topic, and guided me through the whole research.

I want to thank my pre-examiners, Professor Juha Röning and Professor Ray Jarvis. Their insightful reviews helped me improve my thesis significantly.

My thanks also go to my current and former colleagues at Department of Automation and Systems Technology, especially Docent Mika Vainio, Professors Arto Visala and Jussi Suomela, Dr. Jari Saarinen, Janne Paanajärvi, Juhana Ahtiainen, Seppo Heikkilä, Tapio Taipalus, Teppo Pirttioja, Mikko Heikkilä, and many others who have given me advice, inspiration, and support for my research.

I want to thank professors Paolo Fiorini and Monica Reggiani, and the research group at ALTAIR laboratory at Università degli studi di Verona. I also want to thank Dr. Steve Scheding, Thierry Peynot, James Underwood, Andrew Hill, and the research group at Australian Centre for Field Robotics at The University of Sydney. My exchange periods in these excellent research institutes gave me lots of new ideas and motivation for my thesis.

Many thanks go to my friends for supporting me during this project. I am also grateful to my family for being able to grow up in a science-friendly environment and getting lots of support.

Finally, my warmest thanks go to my wife, Terhi Karttunen. This project would not have possible without her endless support and patience.

Espoo, November 2011

Sami Terho

Contents

Author's contribution xiii

List of Figures xv

List of Tables xix

List of Algorithms xxi

List of Symbols xxiii

Glossary xxv

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Robot cognition . . . 2
  - 1.2.1 Shared cognition . . . 3
- 1.3 Problem definition . . . 3
  - 1.3.1 Use cases . . . 5
- 1.4 Scientific contribution . . . 6
- 1.5 Structure of the thesis . . . 6

2 Related work 7
- 2.1 Introduction . . . 7
- 2.2 Spatial cognition in robots . . . 8
- 2.3 Sharing spatial information between humans and robots . . . 11
- 2.4 Cognition model as a part of a higher level model . . . 12
- 2.5 Building the cognition model using sensors . . . 14
  - 2.5.1 Cameras . . . 14
  - 2.5.2 Ranging sensors . . . 17
- 2.6 Discussion and conclusions . . . 18

3 Proposed model for robot's perception and cognition 21
- 3.1 Introduction . . . 21
  - 3.1.1 Perception, cognition, and recognition . . . 21
  - 3.1.2 Shared cognition . . . 22
  - 3.1.3 Example cases . . . 23
- 3.2 Requirements for the model . . . 24

3.3 Components for building the cognition model . . . 25
3.3.1 Automatic and assisted segmentation . . . 26
3.3.2 Recognition . . . 26
3.3.3 Object's spatial states . . . 26
3.3.4 Teaching . . . 27
3.4 Objects . . . 28
3.4.1 Observed object . . . 28
3.4.2 Real object . . . 28
3.4.3 Meta-object . . . 29
3.4.4 Definition of objects . . . 29
3.5 Structure of the model . . . 30
3.5.1 From sensor data to observed and real objects . . . 30
3.5.2 Sharing the cognition with a human . . . 30
3.5.3 Dealing with uncertainty and ambiguity . . . 32

4 Building and using the cognition model 35
4.1 Using robot's sensors to build the cognition model . . . 35
4.1.1 Observations from sensor data . . . 35
4.1.2 Object recognition . . . 41
4.1.3 Handling uncertainty . . . 44
4.1.4 Learning the appearance . . . 46
4.1.5 Object location and orientation . . . 47
4.2 Sharing cognition . . . 50
4.2.1 Referring to the robot's sensor data . . . 50
4.2.2 Describing an object . . . 52
4.2.3 Getting the information from the model . . . 56
4.3 Task execution . . . 57
4.3.1 Objects in cognition model . . . 57
4.3.2 Related tasks . . . 57

5 Experimental validation 61
5.1 Test platforms . . . 61
5.1.1 WorkPartner . . . 61
5.1.2 Avant . . . 67
5.2 Experiments on building the model . . . 69
5.2.1 Building a shared cognition model . . . 69
5.2.2 Recognizing objects on the basis of a description from the user . . . 77
5.3 Experiments on applying the model in task execution . . . 83
5.3.1 Picking up litter . . . 83
5.3.2 Carrying boxes . . . 92
5.4 Conclusions from the experiments . . . 104

6 Discussion 107
6.1 Structure of the proposed model . . . 107
6.1.1 Abstraction of robot's perception and cognition . . . 107
6.1.2 Shared cognition . . . 108

6.2 Building the cognition model 109
6.2.1 Using the robot's sensors to build the model 109
6.2.2 Sharing the cognition 113
6.3 Using the model for task execution 114

7 Conclusions and future work **117**
7.1 Future work 117

Bibliography **119**

Author's Contribution

This dissertation titled "Building and sharing the cognition model of the environment with collaborative service robots" is completely written by the author, Sami Terho. The author has developed the theory presented in this dissertation based on research questions presented by Professor Aarne Halme. The methods presented as new in this research are developed by the author. The experiments are based on four datasets. The first one is a set of images collected by Jari Saarinen during GIM Integrator project. The unprocessed image series was used in the research, the processing and analyzation was carried out by the author. The other three datasets are collected by the author. The author has developed all the software used to process the perception data to produce the results presented in this dissertation.

The software used to collect the data and move the robots has been developed by GIM research group of Aalto University. The hardware has been developed by GIM research group and Department of Automation and Systems Technology of Aalto University. The author has been part of the software and hardware development teams before and during the doctoral research project.

Sami Terho, Author *Professor Aarne Halme, Supervisor*

List of Figures

1.2.1 Robot cognition, human cognition, and shared cognition 4

2.1.1 Building blocks of cognition model. 7
2.2.1 Components for building the cognition model proposed by Vasudevan [2008]. . . . 9
2.2.2 Combination of semantic and spatial knowledge with S-Box and T-Box. 10
2.4.1 Overview of Cognitive Map robot architecture. 13
2.4.2 System architecture of HRP-2P robot. 14

3.1.1 Observed objects, real objects, and their connection with recognition. 22
3.1.2 Robot's sensor data and a segment of it representing a sand pile defined by human. 23
3.1.3 Possible objects segmented from robot's sensor data. 24
3.1.4 Human using a pointer stick to indicate which one of the detected boxes is the target one. 24
3.3.1 Different object types: meta-objects, real objects, and observed objects. 25
3.3.2 Rich 3D data is created by combining image features with their 3D coordinates. . . 27
3.5.1 Components of the system, their relationships, related actions, and human involvement. 31
3.5.2 Transforming a human defined segment of robot's sensor data to observed object and matching that to real object. 32
3.5.3 Real and observed objects with multiple possible matches. 33

4.1.1 Sensor data are segmented and features are extracted. Observed objects are created on the basis of the result. 36
4.1.2 Recognition modalities that match observed objects based on the newest sensor data and previously created real, observed, and anonymous objects. 43
4.1.3 Observations outside the 2σ uncertainty region are considered to be too far from the previous location to represent observations of the same object. 45
4.1.4 Learning occurs from observed objects to real objects, and from real objects to meta-objects. Only characteristic features are copied. 46
4.1.5 Uncertainty of the measured location of an object. 48
4.1.6 Example of topological connections to determine the location of a target object. . . 49
4.1.7 Sign pointing towards the direction of the smaller ball. 50
4.2.1 Examples of machine readable codes that encode information. 53

5.1.1 WorkPartner service robot 62
5.1.2 Close-up photograph of WorkPartner's head 62

5.1.3 Different gripper configurations . . . 63
5.1.4 Coordinate origins and dimensions of WorkPartner's manipulator . . . 64
5.1.5 Geometry for estimating the locations of objects with monocular camera used in WorkPartner . . . 65
5.1.6 Avant machine based intelligent machine. . . . 68
5.2.1 Photo of the scene of the first experiment. The whole sand pile can be seen in the image. . . . 70
5.2.2 Phases of processing the first snapshot of sensor data. . . . 70
5.2.3 Raw 3D data points calculated from a stereo image pair . . . 71
5.2.4 Rich 3D data set calculated from the stereo image pair . . . 72
5.2.5 Boundaries of the pile defined by user. . . . 73
5.2.6 Real and observed objects after user has segmented and connected the data to the real object . . . 73
5.2.7 Elevation map and SIFT features after the first measurements . . . 74
5.2.8 Phases of processing the sensor data on next time instances. . . . 75
5.2.9 Automatic registration and segmentation of the pile from new sensor data . . . 75
5.2.10 Elevation map in different phases of the task execution. . . . 76
5.2.11 Target box of the task . . . 77
5.2.12 The scene in the beginning of the task . . . 78
5.2.13 Hue-Saturation -chart . . . 78
5.2.14 Real objects that define the appearance of the box and the attached circle code . 79
5.2.15 Image of the scene captured with the camera of the robot . . . 79
5.2.16 Observed objects, their measured parameters, the connections to corresponding real objects and the probabilities of the connections . . . 80
5.2.17 Snapshot images from the camera of the robot during the execution of the task . . 81
5.2.18 A human changing the setup . . . 82
5.3.1 Photograph of the area where the pieces of litter are collected. . . . 84
5.3.2 Diagram of the working area viewed from above . . . 85
5.3.3 Detected potential pieces of litter in two camera images taken to different directions 85
5.3.4 Observed objects extracted from the two camera images, and the corresponding real objects. . . . 86
5.3.5 Candidate objects shown to the user . . . 88
5.3.6 Phases of picking up a piece of litter from the ground. . . . 88
5.3.7 First three pieces of litter viewed with camera when the robot is ready to pick up the objects . . . 89
5.3.8 Another three pieces of litter viewed with camera when the robot is ready to pick up the objects . . . 90
5.3.9 Robot trying to grip an object that it thinks is a piece of litter . . . 92
5.3.10 Incorrectly detected pieces of litter viewed with camera when the robot is ready to pick up the objects . . . 93
5.3.11 Picture of the working area in the beginning of the box carrying -experiment . . . 94
5.3.12 The pointing methods used in this experiment . . . 95
5.3.13 View of the pile of the boxes . . . 96
5.3.14 Projection of the measured 3D point inside the object whose dimensions are known 97
5.3.15 User pointing different boxes using the pointer stick . . . 98

5.3.16 WorkPartner lifting one of the boxes from the pile to the platform 99
5.3.17 Bottom-most red box revealed behind the blue box 100
5.3.18 Indicating the objects with a flashlight 101
5.3.19 Pointing the order of the objects during the execution 102
5.3.20 Pointing the order of the objects with the flashlight during the execution 103

List of Tables

5.1.1 Denavit-Hartenberg parameters between WorkPartner's coordinate origin and laser point. 63
5.1.2 Parameters of WorkPartner's camera 66
5.1.3 Parameters of Avant's stereo cameras 69
5.2.1 Measured heights of the objects and the probabilities of the matches to the real object corresponding the blue box. 82
5.3.1 Measured locations of the objects and verbal description of the measured color histogram. 87
5.3.2 Measured locations of the observed objects and match probabilities between the observed and real objects shown in Figure 5.3.13. 96

List of Algorithms

2.1 Determining a dense stereo map with stereo cameras 16
4.1 Calculating rich 3D data 40
4.2 Building an elevation map on the basis of measured 3D points. 42
4.3 Reading and locating circle code 55
5.1 Measuring the location of an object with laser pointer in conjunction with the camera. 67

List of Symbols

f Focal length of a camera

f_L, f_R Focal length, left- and right-hand cameras of a stereo pair

o_x, o_y Image origin, x- and y-coordinates

x_c, y_c Undistorted metric coordinates in camera's frame of reference

x_i, y_i Pixel coordinates (possibly distorted geometry)

p_m Recognition match probability

p_f Feature-related recognition match probability

p_s Probability from recognition match score

p_c Probability from consistency of match

p Probability

$\mathbf{R}$ Rotation matrix

$\mathbf{P}$ Covariance matrix of location

$\mathbf{P}_x$ Covariance matrix of location of object x

$\mathbf{H}$ Projection matrix

$\mathbf{H}_L$, $\mathbf{H}_R$ Projection matrix, left- and right-hand cameras of a stereo pair

${}^x\mathbf{T}_y$ Transformation, that is, translation and rotation from coordinate frame x to coordinate frame y

θ_{pan} Pan (yaw) angle of a pan-tilt-unit

θ_{tilt} Tilt (pitch) angle of a pan-tilt-unit

Glossary

Robot's cognition model Robot's understanding of its environment through sensors and other data sources. A cognition model distinguishes entities from each other.

Shared cognition model A cognition model used collaboratively between a human and robot. A shared cognition model includes information from sensors and from human. In addition, the sensor input can be interpreted by a human.

Perception Observation of the environment using sensors. Perception does not yet contain information on object's identities or classes, but it contains a snapshot of the sensor data and features extracted from it.

Physical entity A concrete distinctive real-world object that can be distinguished. A physical entity does not necessarily have to be discrete, but it may also be, for example a pile of sand. However, a piece of flat textureless ground cannot be held to be a physical entity because it cannot be distinguished by using sensor data.

Object Instantiation of a physical entity in a cognition model.

Object class A group of objects, such as "human", "car" or "work machine". There can also be a more specific description such as "male", "Toyota Avensis" or "Avant 600". However, an object class does not yet identify the target. An exception is a case where there is only one of this kind of object in the world. In this case, a looser definition of the object class may be necessary.

Named object A unique entity. An instance of an object class.

Anonymous object An object whose identity and class are not known.

Appearance Describes what kind of response is got from sensors measuring the object. The appearance may be color, texture, or shape, but also the physical size of the object. Appearance does not only cover what the object looks like to a camera or to the human eye, but it also covers other sensors, such as laser scanners.

Detection Detecting something interesting. The class of the target is not yet known.

Object recognition Generic term for both object class recognition and object identification. Means anything where some recognition or identification information is got from sensor data.

Object class recognition Recognizing the class of the target.

Object identification Determining the identity of a named object.

Chapter 1

Introduction

1.1 Background

Robots are slowly but surely becoming part of our everyday lives. The service robots of the future need the ability to work with people in unstructured environments. Unlike industrial robots working in controlled environments, service robots may need to cope with very complex scenes, recognize objects of interest, and avoid collision with possibly unexpected obstacles. This requires a capability to understand the environment in a way that enables them to work in interaction with the objects that are present. This understanding is built from perceptual information based on the robot's sensor data. In addition, these data may be inputted by a human operator, for example in the form of a map of the environment.

The advantage of using robots is that they are able to execute repetitive, boring, heavy, and dangerous tasks independently. Still, creating a fully autonomous service robot that never needs any assistance seems unrealistic in the near future. In practice, the first wave of useful multi-purpose service robots will probably be collaborative companions instead of completely autonomous agents. When working in cooperation with a human, the robot can get assistance on various levels related to task execution and perception. A robot can ask for help when a problematic situation is encountered, or the user can intervene in the task when he or she considers it necessary. This may occur, for example, when a robot comes across a real-world object that it cannot recognize. In addition, the whole task does not necessarily need to be defined before the execution starts, but part of the task definition can be done during the course of the task. For example, a target object of a robot's task can be defined when the robot perceives it, and it is not necessary to define it explicitly before the task starts. Non-trivial tasks requiring adaptation to a real-world environment are called skilled tasks. [Heikkilä, 2009] In its simplest form, it means moving around without colliding with obstacles. In more complex tasks, adaptation means planning and executing manipulative actions to cope with real-world objects.

Cooperation between robots and humans needs to be efficient. Using a robot should not increase the human's workload, but after getting the required information, it should be able to cope with the task as independently as possible. The problem focuses on the question of how a human operator can describe tasks in a complex world to the robot in the easiest and most straightforward way. An important aspect is to form a way for the robot and the human to communicate with each other about real-world objects and their interrelationships in such a way that they both understand them in the same way.

We humans understand our environments through our senses. Our brain processes the signals from the sensing receptors in our eyes, ears, skin, mouth, and nose, and forms an understanding of what is actually there. In other words, our brain translates our *perception* to *cognition*, more specifically *spatial cognition*. In other words, cognition is the understanding of what we perceive. [Encyclopædia Britannica, 2010] With spatial cognition we understand where objects are, what *class* they belong to, what their *identities* are, and what their *spatial states* are, that is, what the locations and orientations of the objects are and their other properties, such as color, more detailed structure, or material. We are able to *recognize* objects that we have seen before, but also *learn* about new objects. The new objects may be taught to us by another person, we may have acquired the information on the objects by ourselves, or we may base the learning on our previous *knowledge* and understand their connection to what is being learned. In this context, objects are distinctive boundable parts of the world. They are not necessarily physically separated from other objects.

Spatial cognition is description of the environment. Using this description, we are able to perform tasks that require interaction with the environment, such as picking up and manipulating objects, avoiding obstacles while walking, and finding our way to an object whose location we know. Spatial cognition goes beyond perception in that we do not necessarily need to see an object to know where it is, if we have seen it before, provided it is not moving. Spatial cognition also involves memory of what has been perceived.

1.2 Robot cognition

To perform skilled tasks and to cooperate with humans, a robot needs to understand its environment. The sensor data are interpreted to the robot's perception. This is a low-level concept that deals directly with what the robot perceives. The perception information is translated to understanding. The robot's understanding basically means that it knows how to interpret the perceived data. This could be thought of as the *robot's spatial cognition*. More specifically, a *cognition model* of the environment is built. The robot's cognition is not an ability generated by evolution like the cognition of natural creatures, but it needs to be defined. Various approaches to robotic cognition have been proposed, for example by Vasudevan [2008] and Kaupp [2008]. The cognition model presented in this research focuses on the following three aspects.

1. What are the locations of the objects?
2. What are the types and identities of the objects?
3. What are the spatial states of the objects?

A robot's cognition is conceptually different from human cognition in that a robot's cognition only concentrates on the three above-mentioned things, and is not necessarily able to understand, for example, beauty or other emotional meanings of objects. In addition, the cognition model does not include higher-level interpretations of objects, such as what a person is doing or where he or she is going to.

Building a robot's cognition model requires data to be gathered from various sources and translated into cognitive information. Interpreting sensor data as cognitive information is called *recognition*, more specifically *object recognition*. The latter term emphasizes that a cognition model is built from separate *objects*. A classical theory of recognition in artificial intelligence looks for

methods and algorithms for automatic recognition. However, in many practical situations this is an extremely difficult task compared to what human brains can solve. Information can also be entered into a cognition model in a way that does not require the interpretation of sensor data. For example, if a human describes the class, name, and location of an object, this information can be directly entered into the cognition model. This can be used to assist a robot in recognizing the objects, and, in general, in understanding its environment.

1.2.1 Shared cognition

A robot's cognition describes how the robot understands its environment. However, the information on the environment does not necessarily come from the robot's sensors only. If the information is gathered with robot's own sensors, it may still be interpreted by an external agent, which, in practice, means a human. The human can be considered to transfer part of his or her cognitive understanding to the robot's cognition. *Shared cognition* is an extension of robot cognition that allows a human to input and read information from the cognition model. Once information has been inputted to the model, it is also part of the robot's understanding. Therefore the term shared cognition may be used. The information exchange can be partly based on the robot's sensor information while utilizing a human's cognition. This is a tool for a human and robot to cooperatively build an understanding of the environment.

A shared cognition model can also be presented to a human. If a robot is uncertain about how to interpret its perceptual information, it can ask the user for help in, for example, recognition. The shared cognition allows the robot and human to understand the scene in a similar way. This makes it easier for the human to assist the robot.

Shared cognition is used to share spatial and other measurable information between robots and humans. Robots are not, in general, able to understand emotional meanings, such as the beauty of objects. These are highly subjective, and dependent on a human's opinion, and therefore not suitable for sharing with a robot. Shared cognition requires the information shared through it to be objective and measurable.

Figure 1.2.1 illustrates some concepts related to human cognition, robot cognition, and shared cognition. The human has recognized the class of the object, and therefore is able to know that it is a box. He also estimates the location and color of the box on the basis of his intuition. The robot has also recognized the box, and describes it with its own measures. If the robot cannot recognize the box, shared cognition can be used. The robot shows its sensor data to the user, who in turn marks the region representing the box. The human has therefore used his cognition to interpret the robot's perceptual data. Because marking the object is done in the robot's own sensor data, it can immediately relate it to its own sensors and coordinate system.

1.3 Problem definition

The objective of this research is to create a concept that enables robots and humans to collaboratively build a shared cognition model, that is, a representation of the environment that not only states where the objects are, but also what their classes, names, and their spatial states are. The basic problem is how to do this in the best way so that the human-robot collaboration is effective and productive in task execution. The building of the model is based on human input and the robot's sensor information. As a result, the robot would be able to recognize objects it perceives

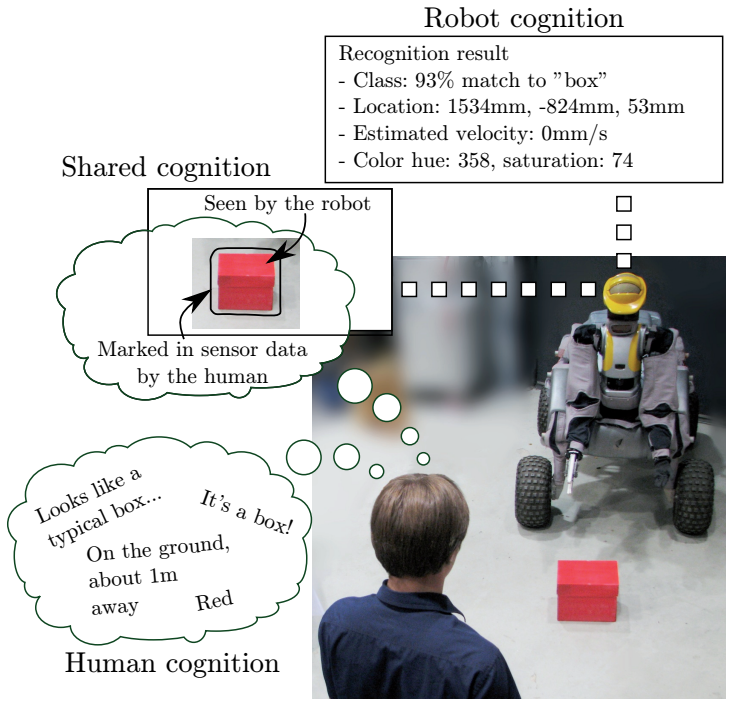


Figure 1.2.1: Robot cognition, human cognition, and shared cognition

with its sensors, and it should be able to determine their properties.

The human should be able to utilize his or her cognition as effectively as possible to assist the robot in recognizing and locating the objects and determining their states. This requires a robotic shared cognition model that can be used to communicate sensor information and its interpretation between the robot and the user in a collaborative manner. The cognition model should also be able to learn from prior knowledge of the objects. The cognition model may not be tied to one human, but any human should be able to use it to exchange information with the robot. Therefore, the cognition model cannot store subjective information, but only the measurable properties of objects.

The objective is not to create new algorithms for autonomous object recognition, but rather to utilize the methods and algorithms that already exist. However, the focus is especially on how the user and robot can collaboratively form the cognition model. This research does not define a software architecture that implements the cognition model, but it focuses on describing the model itself.

As a summary of the above, this research answers the following questions.

1. How can the robot's cognition be modeled in such a way that the model includes identifying information on objects: appearances, locations, spatial states, classes, and names?

2. How is the model used for recognizing objects using the robot's perception?

3. How is the information needed for recognition inputted to the model by a user, and what kind of robot optical perception data are considered in this process?

1.3.1 Use cases

The following use cases illustrate more specifically different aspects related to the above description of how a cognition model is built and of what kind. These cases are described in greater depth later when the related techniques are being presented. The tests also follow these use cases.

1.3.1.1 Marking an object from sensor data

An approach that focuses especially on shared cognition is using the robot's sensors along with a human's cognition to build the cognition model. The user defines which part of the robot's sensor data represents the object in question. This definition can be created, for example, from a three-dimensional set of points gathered with stereo cameras or a laser scanner. The robot learns part of the appearance on the basis of this definition. In such a case the user action replaces, e.g., an algorithm that tries to segment the sensor data. The robot can learn even more about the appearance of the object by tracking the object while approaching it and perceiving it from different angles.

Defining the sensor data can be performed by showing the user the data from the robot's sensors, after which the user defines a segment that represents the object. Another possibility is to communicate through the robot's perception by marking the object with signs, with a pointer stick, or by illuminating the object. The robot then gets the required input through its camera image.

An example of this type of case is one where the robot encounters an object that is hard to define and recognize automatically. One example of such an object is a pile of sand. As every pile of sand is different and even the boundaries of the pile may be ambiguous, collaborative recognition and object localization enable the user to utilize his or her cognition to build the shared cognition model that the robot can then utilize. With traditional object recognition methods it would be very difficult or even impossible to define the pile in such a way that the robot could recognize it automatically. The problem can be solved with a collaborative approach.

1.3.1.2 User describing an object to robot

The user can assist in building the cognition model by describing objects in various ways and inputting this information to the cognition model. Recognition is then based on this description inputted by the user. The user can input the information in various ways, describing the object's features verbally or numerically, defining a code associated with an object, showing the robot one or many photographs of an object, or defining which class the object belongs to.

An example of such a case is describing the color of the object to the robot. To make sure that the robot and human interpret the colors similarly, a predefined color map can be used. The colors on the map are interpreted by the user by looking, and by the robot by their numeric codes, which represent the recognition parameters. After that the robot can search for all the potential objects and present them to the user. If the robot can recognize the objects reliably, it can also perform the manipulation without asking for further advice. The need for further cooperation depends on the reliability of the recognition.

1.4 Scientific contribution

The research presents a model for the shared cognition of a robot and a human that is based on the robot's perception and human input. The structure of a cognition model, as well as a model for recognizing the objects, is presented. The research makes a scientific contribution in the following areas:

- the research defines the components needed for building a shared cognition that allows information to be exchanged between a robot and a human using the robot's perception combined with user input;
- a shared cognition concept is proposed. The concept abstracts the robot's perception, cognition, and knowledge, and presents them with objects;
- the concept enables the cognition model to be built and utilized collaboratively. The user can train the model by defining objects from the robot's sensor data, inputting external sensor data, and by defining rules for recognizing objects;
- the definition of the cognition model includes essential elements related to the recognition. The concept does not restrict the use of recognition modalities, and enables multiple sensors to be used for recognition.

1.5 Structure of the thesis

- Chapter 2 presents an overview of the current state of the art related to building a cognition model.
- Chapter 3 presents the theoretical framework of the thesis. This chapter defines the various concepts and principles which the cognition model is built on, requirements for the shared cognition model, and building blocks of the model.
- Chapter 4 describes how the model is built and used in practice. The chapter first presents the aspects of building the cognition model using its sensors, then the discussion is expanded to present human involvement, and finally the task execution utilizing the cognition model is presented.
- Chapter 5 evaluates the validity of the model through four experimental tests: measuring the shape of a sand pile on the basis of user-defined boundaries of the pile, recognizing an object on the basis of appearance description from the user, recognizing and collecting pieces of litter from the ground in cooperation with the human, and carrying boxes in the order indicated by human using a pointer stick and a flashlight.
- Chapter 6 goes through the whole model, and discusses the applicabilities of its features. This chapter discusses the results from the experiments and also aspects that were not validated experimentally.
- Chapter 7 presents conclusions to the dissertation guidelines for future research related to the field.

Chapter 2

Related work

2.1 Introduction

Building a cognition model is a task that consists of subtasks on many levels. In a shared cognition model, the information can be inputted both from the robot's sensors and from a human. Both geometric and appearance information are extracted and processed on the basis of the input. This information is the base of the cognition model. Figure 2.1.1 illustrates the building blocks of the cognition model and their relationships.

The rough division presented in Figure 2.1.1 is based on concentrating on two measurable features of the real world: the geometry of the physical entities and their appearance. 'Appearance' as used here means the response measured by any sensor, not merely the visual appearance as perceived by a camera. The features in the figure refer specifically to the description of the appearance. In a shared cognition between a robot and a human, both aspects of the cognition model should be able to be affected by both the robot and the human. This chapter reviews the current state of the art related to these subjects.

Section 2.2 covers different representations of the cognitive model of the robot. This section makes a state-of-the-art analysis of techniques used to form the cognition model basis presented in Chapter 3.

Section 2.3reviews different approaches for exchanging spatial information between a human and a robot. The analysis is the basis of Section 4.2 in Chapter 4.

Section 2.4 presents high-level architectures that consider the spatial representation as well as the human-robot interaction as parts of the architecture. This section provides the background for Section 4.3 in Chapter 4.

Section 2.5 reviews the techniques needed for building the cognition model. This section dis-

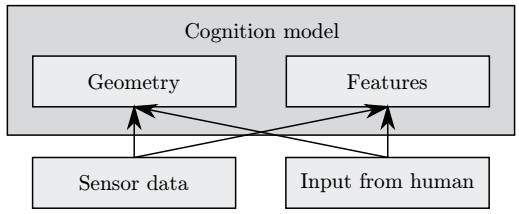


Figure 2.1.1: Building blocks of cognition model.

cusses building a geometric representation of the environment using the sensors, as well as different approaches to recognizing objects from the sensor data. This section presents the basic techniques related to Section 4.1.

Section 2.6 includes a summary of the reviews and a discussion about what aspects can be utilized from the architectures and methods that have been described and what limitations in the present state of the art need to be overcome.

2.2 Spatial cognition in robots

Cognitive understanding of the environment is an essential part of a mobile robot and is therefore studied widely. There are various approaches to forming a model of the environment. This section presents different representations for spatial cognition in robots. The following aspects of the presented approaches are evaluated:

- their ability to provide a general cognitive understanding
- the information contained by the representations and their applicability to enable manipulation and navigation
- their ability to cope with uncertainty
- how the objects are defined by both using predefined descriptions and by learning from the sensor data (possibly in cooperation with a human)
- how the cooperation with a human is considered in general

The simplest approach is to treat everything as obstacles. This approach facilitates autonomous navigation and obstacle avoidance. Simultaneous localization and mapping (SLAM) can be executed on the basis of data from a camera or a range sensor.

Cognitive environment maps go further as they build a semantic map of the environment. In addition to the raw sensor data needed for navigation, a semantic map contains information on the appearance and names of places.

Vasudevan et al. [2007] and Vasudevan [2008] proposed an object-oriented hierarchical probabilistic representation of cognitive spatial information. The representation aims to enable the robot to understand not only where it is in metric coordinates, but also to understand what kind of place it is in and possibly to recognize places visited earlier. The representation does not primarily aim to enable manipulation tasks. The representation involves both forming a high-level representation of space and understanding and reasoning about the place. In this representation, a place refers to a spatial abstraction. When working indoors, rooms can be considered as places. Recognizing objects is based on SIFT features, nearest neighbor searching, and feature grouping, as described by Lowe [2004]. The learning of the recognition is done automatically by mapping the environment. The research focuses mainly on the representation of the space as a hierarchical interconnected structure of objects. Places are objects that connect to each other through doors that are regarded as objects too. There can be objects recognized within places, and they are hierarchically connected as being part of the corresponding places. The locations as well as the identities of the recognized objects are presented as a probabilistic representation. The model is focused on classifying, recognizing, and learning about the places. The model defines a set of

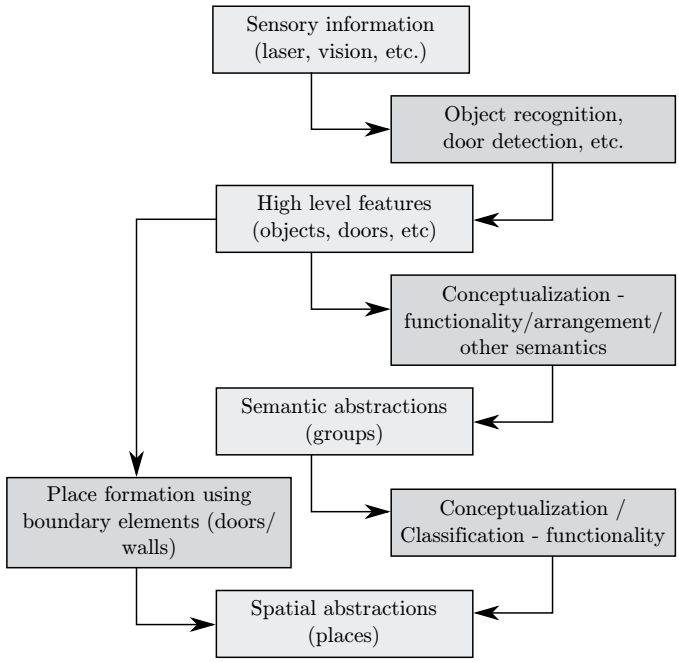


Figure 2.2.1: Components for building the cognition model proposed by Vasudevan [2008]. Adapted from [Vasudevan, 2008]

actions for reasoning related to the mentioned processes. The representation aims to be understandable by humans, but it does not enable input from a human operator. Figure 2.2.1 presents the components used to build the cognition model.

A related approach was proposed by Thrun [1998]. The representation combines grid-based metric maps and topological maps to increase the reliability of the mapping of the environment. The metric maps are related to interconnected topological places. This approach is studied with a robot equipped with an ultrasonic sonar sensor. The representation is only aimed at autonomously solving the SLAM problem of the robot rather than cognitively understanding the environment. The uncertainty is modeled by probabilities of the occupancies of the grid representing obstacles.

Mozos et al. [2007] presented an approach that fuses 2D laser scanner and panoramic camera data to recognize places. This approach also combines metric and topological mapping. A laser scanner is used for building a geometric map of the environment and to classify regions based on geometric primitives representing walls and corners. This classification is used to determine the type of each region. Camera images are used for recognizing objects. The recognition is based on Haar-like features proposed by Lienhart and Maydt [2002]. In addition to the use of image data, the topological location is used as additional information. This is based on the probability of the occurrence of each object related to the location. For example, one is more likely to find a computer in an office than in a kitchen. Also in this representation, the cognitive understanding focuses on localization of the robot rather than manipulation of the objects. Building the model is based on supervised learning. Thus, human cognitive understanding is utilized in forming the model.

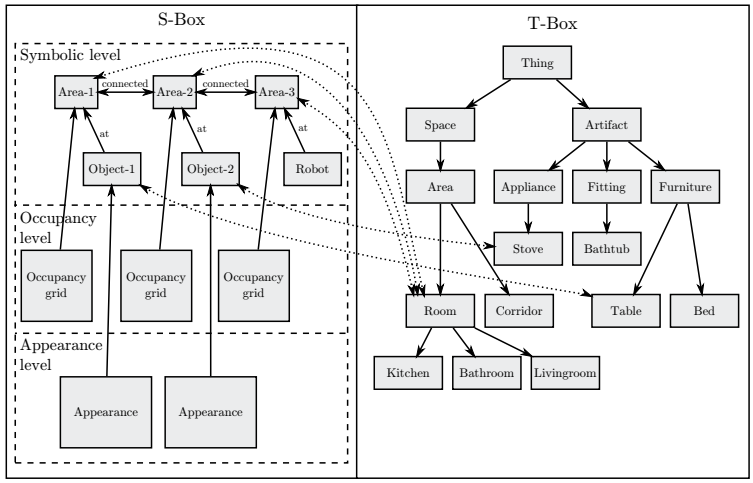


Figure 2.2.2: Combination of semantic and spatial knowledge. S-Box represents the spatial relations between the objects, and T-Box the semantic relations. Adapted from [Galindo et al., 2008]

Another supervised learning -based approach was proposed by Ranganathan and Dellaert [2008]. The approach uses stereo cameras for classifying places. The places are classified on the basis of the occurrence of objects and their relative locations. The recognition is based on SIFT features, combined with other corner detection methods, to describe the appearances. In addition, 3D models of the objects are built and integrated into the constellation describing the places. The uncertainty of the locations is modeled as Gaussian posterior distributions. The probability is also calculated for recognition on the basis of the prior probabilities of the components used for the recognition. As this approach is based on supervised learning, a human is needed to teach the classes of the objects represented by sensor data.

Nuchter and Hertzberg [2008] classified the environment using 3D data acquired with a laser scanner. Different components (such as the ceiling, floor, walls, doors, and humans) were classified by applying a Haar-like classifier to the depth data. In addition, the RANSAC algorithm was used to extract planes from the depth data. In practice, the approach basically leads to rules where, for example, horizontal flat surfaces are interpreted either as floors or ceilings and vertical surfaces as walls. The semantic labels that were extracted were attached to each piece of 3D data. No human involvement was considered in this representation. The cognition model was not used for any particular tasks, and only map building was tested. However, because the model includes pure 3D data, it could even be used for manipulation tasks. However, using this representation requires a sensor that is able to produce 3D measurements of the environment.

The actual purpose of building a cognition model is usually to guide a robot's actions. Galindo et al. [2008] studied the use of a cognitive environment map in robot task execution. The robot can be guided on the basis of its knowledge of the environment. The robot can be given commands such as "Go to the kitchen", because the location of the kitchen is known in the cognition model. Figure 2.2.2 shows a representation of the knowledge in the model. The description of the objects is done through T-boxes (terminological boxes), while the spatial information is stored in S-boxes (spatial boxes). The T-boxes use a heavily structured representation for the information, based on a verbal

description that can be considered a programming language. The descriptions of the objects do not involve using sensor data in teaching. No human involvement other than programming the descriptions is considered. The uncertainties are also not modeled in the representation.

2.3 Sharing spatial information between humans and robots

In a collaborative task, a robot and a human often need to share spatial information. This can be information about the location of an object, the location of a working area, or similar information related to the task at hand. As the previous section also evaluated approaches with some human involvement, this section focuses on techniques whose primary aim is to enable spatial and cognitive information sharing between the human and the robot. The technologies are evaluated based on the following:

- how the information is exchanged to and from the model
- modalities of describing objects to the robot
- human involvement in assisting the robot in recognition

Kaupp [2008] and Kaupp et al. [2007] studied the exchange of perceptual information between the parties in a human-robot team. The research presents a probabilistic framework for combining sensor data from one or more robots and input from a human. The approach integrates human to the framework simply by considering the human as a sensor. Human input is translated into machine representation through human sensor models that apply likelihoods to the observations. Another option is that the human applies the uncertainty to the observations. In addition to inputting information to the model, the user gets a view of the state of the world as stored in the model. The uncertainties are represented with multi-level Bayesian networks. The highly detailed sensor models consist of multiple Gaussian distributions. The framework primarily focuses on coping with obstacles and ground formations.

Suomela et al. [2005] studied collaboration of a multi-entity rescue team consisting of humans (particularly firefighters) and robots. Both the robots and the humans were localized within a pre-built map of the environment. The humans also carried a sensor set for this purpose. In addition to sensors, the humans could also update the state of the model using a graphical user interface. The state of the model was not affected by the sensor data apart from the locations of the entities. Therefore object recognition was not performed. The cognition model, called common presence, enabled the human to view the state of the model through a wearable display device. The aim was to provide a common representation for all the entities of the system.

Topp and Christensen [2006] proposed a framework for Human Augmented Mapping. The aim was to allow human label regions mapped by the robot using SLAM. The space is partitioned into regions either automatically or by a human. The regions are then labeled by a human. Recognition of the regions (rooms in this experiment) does not require global location of the robot. Instead, recognition is based on a statistical classification of the features measured from the region. The concept was made more concrete in Topp et al. [2006] through a practical experiment where the human gave the robot a “guided tour” of the environment. The robot followed the guide, who gave verbal commands to the robot. Peltason et al. [2009] further extended the concept by allowing the robot to first try to recognize the regions on the basis of its previous experience in similar places. The robot asked the user for confirmation, who either confirmed the robot’s hypothesis

or presented correct information. All the descriptions to the robot were given by describing the robot's sensor data.

Collaborative control of a robot executed by a robot and a human was studied by Nielsen and Bruemmer [2007]. The robot's world model was rendered as a 3D model and shown to a human. In addition to the 3D model, the camera images were incorporated in the visualization providing an efficient fusion of the snapshot of the robot's cognitive understanding and raw sensor data for human interpretation. Visualizing the cognition model through a 3D model ensures that the robot and human understand the world in the same way. The human gets his task-related perception through this visualization. This reduces the possibility that the robot and the human would interpret the scene differently. The human input is not utilized in the recognition, but just in controlling the robot on the basis of the model.

Nagata et al. [2010] studied object manipulation, specifically picking up objects from the table using a robot manipulator equipped with a gripper. The user instructed the robot in three phases. First, the human showed the target object on the camera image. The robot used this information to determine the location of the object. Then, the human determined the object class from a few options. The object classes did not necessarily represent the exact geometry of the object in question, but they represented a general classification such as "plate", "bowl", and "mug". Finally, the human instructed the robot on how it should grip the object. For each class, there were a set of predefined grasp modes for the gripper. The object classes and grasping modes were all predefined in the system. The only input from the human was the choice of the object. However, the robot's sensor information was used to determine the accurate position using stereo cameras.

2.4 Cognition model as a part of a higher level model

As mentioned earlier, the cognition models mainly tackle the problem of localization. Even though the discrete objects in the environment may be recognized, this information is often used just for recognizing the place where the robot is at. Higher level tasks, however, may need to execute tasks more advanced than merely navigating around. A typical high-level task is manipulating objects, possibly in addition to navigation. To perform such a task, the robot must be equipped with a suitable manipulator.

The review in this section evaluates the techniques with respect to the following features:

- their ability to cope with high-level task planning and motion planning
- suitability for manipulation tasks
- representations for the object description used to recognize objects
- user involvement in building and utilizing the models

Ng-Thow-Hing et al. [2007, 2008, 2009] studied task execution with the Asimo robot [Honda, 2011] with an architecture that considers the task execution and user interaction in addition to the cognition model. The architecture, called the Cognitive Map, stores the sensory inputs as well as inputs from higher-level data interpretation modules such as object recognition. The model is based on a blackboard architecture that stores the information so that any module can write or read the data using the Cognitive Map. The model has been used to demonstrate tasks such as playing a memory card game with a human and grasping various objects (such as a

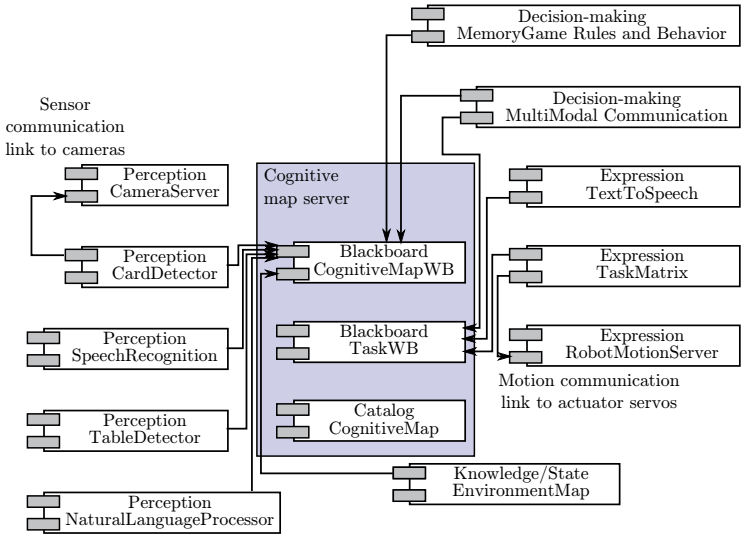


Figure 2.4.1: Overview of Cognitive Map robot architecture. Adapted from [Ng-Thow-Hing et al., 2009]

wine glass). A typical characteristics of this architecture is that it abstracts the building of the cognition model on a relatively high level. The object recognition is implemented in separate modules, and they produce very refined data. The architecture does not specifically define the internal structure of the recognition itself. The Cognitive Map can store appearance information about the objects, but its interpretation depends on the other modules. Figure 2.4.1 shows an overview of the Cognitive Map robot architecture. The different modules for recognizing objects and for user interaction communicate through the CognitiveMapWB component, which contains the actual map of the environment. The CognitiveMap catalog component is a database containing persistent information. Task execution occurs through the TaskWB component. The suitability of this approach for task and motion planning, as well as for user interaction, depends on the task modules. After all, the cognitive map does not restrict the data content much. Because of this, however, the architecture does not accurately define the exact representation of the cognition.

Sian et al. [2006] proposed a different kind of system for controlling biped robots HRP-2P [Yokoyama et al., 2003] and HRP-2 [Kaneko et al., 2004]. A system-level architecture is defined for controlling the robots. The architecture allows controlling both low-level actions (such as hand and leg movements) and high-level tasks based on behaviors. Figure 2.4.2 shows the block chart of the system architecture. The environment models of the robots are built using stereo cameras. Objects are recognized using a CAD-model based recognition modality proposed by Sumi et al. [2002]. Using this, the robots are able to execute relatively advanced tasks, such as grasping a can of soda and taking it out of a refrigerator and assisting a human in carrying heavy objects. The architecture is very specific to these robots and is not necessarily applicable to other applications. Also the software architectures of the later generations, such as HRP-3P [Akachi et al., 2005], are designed in a similar way.

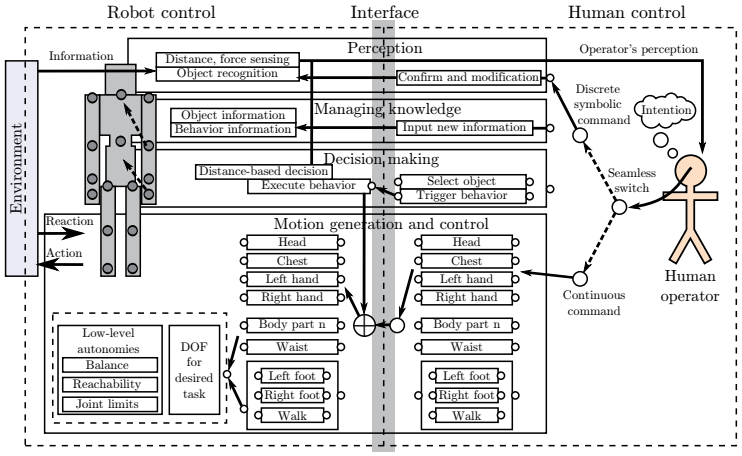


Figure 2.4.2: System architecture of HRP-2P robot. Adapted from [Sian et al., 2006]

2.5 Building the cognition model using sensors

This section reviews different techniques used to build cognition models, and their advantages and limitations. The sensors are used to provide a geometric model and a semantic interpretation of the environment, which together form the cognition model.

2.5.1 Cameras

Cameras can be used for recognizing objects, providing rich information about the environment. Stereo cameras (and to some extent monocular cameras) can be used to build a 3D geometric model of the environment. For these reasons, cameras are currently used in almost every robotic system. They have been used as a part of the system in all the examples described in this chapter.

While cameras are very useful in many situations, they also have limitations. Cameras don't work well in poor lighting conditions. Low light forces the camera to either use longer shutter times or allow more noise in the image, which are both undesirable. Therefore a camera with a sensitive sensor is required when working in low light.

2.5.1.1 Measuring the structure with cameras

Monocular camera Measuring the full 3D geometry with a monocular camera requires solving a structure-from-motion problem. The methods required have been presented by [Ma et al., 2003]. The method is based on simultaneously estimating the motion of the camera and the structure that the camera is imaging.

Full 3D geometry is not always needed. The locations and sizes of objects can often be estimated if assumptions about the locations of the objects can be done. For example, if the object is known to lie on flat ground and the observer's own location and orientation are known, the estimated position of the object can be determined by projecting the observation onto the ground plane [Hoiem et al., 2008].

If an object can be recognized using methods described in Subsection 2.5.1.2 below, the size can be calculated on the basis of that. This technique is used in Gordon and Lowe [2006]. The 3D

structure of the object is known, and the size in the image is related to this information.

Stereo cameras Stereo cameras provide more information than a single camera does. They can be used to generate a true 3D model even from a single image pair. Stereo cameras were used by Vasudevan et al. [2007] and Vasudevan [2008] to build a 3D model.

Stereoscopic vision simulates the average human's way of viewing the world with two eyes. The distance to the camera is calculated on the basis of the differences between the two images. Matching points are found in both images. The corresponding 3D points are calculated by triangulation.

There are basically two different ways to use stereo cameras for creating a 3D model: the dense approach and the sparse approach [Trucco and Verri, 1998].

Dense stereo matching compares camera images from two different views and tries to determine the correspondence between every point in both camera images. The result is called a disparity map. It contains lots of 3D information, but some of the disparities may be incorrect, because the disparity is also determined for the image regions with few image features. In these image areas, it may be difficult to unambiguously determine the correspondence between the images. The calculation of the disparities can be based on various techniques. The simplest one is based on calculating the sum of squared differences (SSD) between small image batches in the left- and right-hand images and comparing them to each other. Sub-pixel accuracy can be achieved by fitting a second-order curve to a local fitness function [Anandan, 1989]. Outliers can be partially eliminated by checking the match consistency from the left-hand image to the right-hand image and then from the right-hand image to the left-hand image. If the matches are equal, they can be considered reliable; otherwise, the match is discarded [Fua, 1993]. Algorithm 2.1 describes a combination of the two approaches, bidirectional matching and sub-pixel refinement.

The reconstruction of 3D points on the basis of disparities does not depend on the type of disparity map. The reconstruction is calculated for each disparity point with triangulation on the basis of the geometry of the cameras. If the stereo system is rectified, the reconstruction can be done simply with

$$\begin{bmatrix} x_{3D} \\ y_{3D} \\ z_{3D} \end{bmatrix} = \frac{t}{d} \cdot K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.5.2}$$

where x_{3D}, y_{3D}, and z_{3D} are the 3D coordinates, x and y are the pixel coordinates, d is the corresponding disparity value, t is the distance between the cameras, and K is the camera calibration matrix. With non-rectified images, noise in detected locations usually causes the rays to not intersect, and the reconstruction needs to be done on the basis of finding the point in space where the rays are closest to each other. This can be done using various methods. Hartley and Sturm [1995] presented a method that minimizes the polynomial describing the distance. Other methods are mainly based on estimating a solution to a linear case.

The calculation of the dense disparity map can be made faster by rectifying the images so that the matching features are always found from the same pixel row in both images. This means transforming the images in such a way that the epipoles are transferred to infinity. In practice, homography transforms that use projection matrices are calculated for the images [Fusiello et al., 2000].

While stereo cameras are useful for many situations, they have limitations. The maximum distance that can be measured with stereo cameras is limited. The accuracy of the distance measurement is also inversely proportional to the distance. The surface needs to have a visible

Algorithm 2.1 Determining a dense stereo map with stereo cameras

The algorithm assumes two calibrated input images with geometric distortions corrected and epipoles set to infinity. The following constants need to be determined.

- Size of image patch size used for comparison: w
- Disparity search range in pixels $[a, b]$

1. For each $w \times w$ image patch I_L at (x_L, y_L) in left-hand image
   (a) Loop x_R from $x_L + a$ to $x_L + b$.
      i. For each image patch I_R in the right-hand image, calculate the sum of squared differences between I_L and I_R:

$$\mathrm{SSD}\left(x_R\right) = \sum_{i=-\frac{w-1}{2}}^{\frac{w-1}{2}} \sum_{j=-\frac{w-1}{2}}^{\frac{w-1}{2}} \left(I_L\left(x_L + i, y_L + j\right) - I_R\left(x_R + i, y_R + j\right)\right) \quad (2.5.1)$$

   (b) Choose the x_R with the smallest calculated SSD.
   (c) Refine the x-coordinate by fitting a parabola to local SSD's around the chosen x_R: t

$$x'_R = x_R + \frac{0.5 \cdot SSD\left(x_R - 1\right) + SSD\left(x_R\right) - 0.5 \cdot SSD\left(x_R + 1\right)}{SSD\left(x_R - 1\right) + 2 \cdot SSD\left(x_R\right) + SSD\left(x_R + 1\right)}$$

   if $0.5 \cdot SSD\left(x_R - 1\right) + SSD\left(x_R\right) + 0.5 \cdot SSD\left(x_R + 1\right) > 0$ and $x_R - 1 < x'_R < x_R + 1$.
   (d) Save x'_R to the disparity map

2. For each $w \times w$ image patch I_R at (x_R, y_R) in the right-hand image
   (a) Loop x_L from $x_R - b$ to $x_R - a$.
      i. For each image patch I_L in the left-hand image, calculate the sum of squared differences between I_R and I_L using the formula 2.5.1.
   (b) Choose the x_L with the smallest calculated SSD.
   (c) If x_L and corresponding x_R do not match, discard the element in the disparity map.

texture for matching; otherwise, the matching becomes ambiguous. In case of a highly reflective surface, the different cameras may observe different lighting, which may negatively affect the performance. A fundamental problem with stereo cameras is the presence of occlusion. With two cameras and variations in depth, there are always regions that are only visible in one camera and not in the other. It is not possible to measure the distances to those points. With more cameras, this problem can be at least partially solved. In addition, on the sharp edges, the resulting 3D measurement may appear smoothed. If the image-to-image matching is done using a moving window, the window causes a smooth transition instead of a sharp edge.

2.5.1.2 Recognizing objects with cameras

Cameras are widely used for object recognition applications. They are used in almost every application described in earlier sections of this chapter. When building a cognition model, the semantic information is easiest to collect with cameras.

Sumi et al. [2002] used a CAD-based model of the target objects. The model was combined with depth information from stereo cameras to provide an accurate match. Based on the CAD model,

the location and orientation of the object with the known geometry could be determined. Ulrich et al. [2009] proposed a pyramid-based approach that enables the object to be searched for in a single image without any prior information. This approach also fits a geometry to the lines present in the image. This is done on many pyramid levels, which allows the shape to be searched for anywhere in the image instead of just near a predetermined initial position or previously observed position.

Local image features, such as Scale Invariant Feature Transform, SIFT Lowe [2004], and Speeded Up Robust Features, SURF Bay et al. [2006] locate keypoints from the images and extract feature descriptors on the basis of their environment in the image. The descriptors are in general invariant to changes in illumination, scale, and two-dimensional planar rotation. A group of such features describes the object when viewed from one direction. SIFT and SURF are widely used for object recognition because they are easy to apply, and the learning can be done on the basis of single images only.

If larger areas of image are analyzed, then segmentation techniques are required. Typical ways to perform segmentation is to find areas with similar colors, for example by searching for edges. Another technique is the MSER algorithm [Matas, 2004]. This algorithm finds any area that is brighter or darker than any of the surrounding pixels. Grayscale segmentation can also be generalized from the grayscale channel of an image to any channel, as in color analysis.

The color of an object is a very commonly used feature in object recognition . It can be combined with segmentation as a criterion for segmentation, or after segmentation the color can be analyzed from the extracted region. The color comparison is usually based on comparing color histograms of the reference image and the target image Swain and Ballard [1991].

Texture is a feature similar to color, as it describes the appearance of the surface. Various representations have been developed to present texture, for example, Local Binary Patterns [Ojala et al., 2000] and Textons [Shotton et al., 2006].

2.5.2 Ranging sensors

Laser scanners and radars are widely used sensors in robotics. Laser scanners are found in almost every platform used in robotics research. Their main advantage over cameras is their ability to get reliable distance measurements, even from long distances. Laser scanners are often used mainly for robot navigation and are installed with that purpose in mind. Typically, a 2D laser scanner is installed to scan in the horizontal plane to allow the detection of obstacles and landmarks. An environment model built with this kind of sensor is usually considered to be a 2D map viewed from above. Data of this type can also be used for limited object recognition if the geometric models of the objects are known [Fayad and Cherfaoui, 2007].

If the laser scanner can be pointed in different directions, the measurements can be combined to form a 3D model of the environment. This requires the sensor's location and orientation at each instant in time to be known. The locations of the measured points can be calculated on the basis of the known translation and rotation [Underwood et al., 2010]. The problem with this approach is, however, that the accurate determination of orientation often requires expensive sensors.

Using a 3D laser scanner often acquires the most accurate representations of the scene when compared to other sensors. The problem with these sensors is that they are generally very slow. They scan the entire scene in two directions using just one laser beam. 3D laser scanners are often used for the stationary monitoring of an area, or they are used to get reference data that are later

refined with other sensors.

Object recognition can also be performed using 3D data acquired with laser scanners. The advantage of this approach is that the actual geometric information can be utilized when finding the object in the sensor data. Point signatures [Chua and Jarvis, 1997] describe the structural neighborhoods of points to allow the points to be recognized in 3D data. The signatures describe the local shapes in terms of surface curvature in different directions. The shape is mapped to a one-dimensional vector that can be easily matched. Spin images [Johnson and Hebert, 1999] describe the intersections of 3D data in different directions. A keypoint is chosen in the 3D model. Intersections of the model are projected to small image patches, called spin images. Similar projections are sought from the 3D sensor data.

2.6 Discussion and conclusions

There are various approaches for representing the cognitive understanding of robots. Understanding the environment for the purpose of localization is an especially widely researched subject. The techniques described in Section 2.2 are all primarily aimed toward this purpose. None of the test platforms have a manipulator, therefore the robots were designed mainly for research on navigation. The same trend can be seen in the techniques that share the spatial cognition model with a human. In Section 2.3, there are mainly techniques that utilize human cognition in naming places on a map. Only the approach proposed by Nagata et al. [2010] was aimed at manipulating the objects. Human assistance was used for the difficult task of determining how the object should be gripped. Section 2.4 described systems that are used to perform advanced tasks. The manipulation is considered in these cases, but they are in general very task-specific.

The modularity of the presented techniques, and their applicability to different domains, varies a lot. This can be measured with two factors: the possibility to use the approach in another system, and the possibility to affect the model itself when it is running in the described system. The overall trend seems to be that the more advanced the system and its tasks are, the more specific the architecture is used to run the system. The biped robots Asimo and HRP series were designed to execute the most advanced tasks with movement and manipulation of objects. However, the spatial cognition is not very well defined, and the algorithms and models are mostly task-specific. On the other hand, the techniques for place recognition were well defined and clearly aimed for generic use. Their applicability for manipulation tasks is not known because it is not tested in practice.

Object recognition and measuring the geometry of the environment are key technologies in forming a cognitive understanding of the environment.

Stereo cameras usually provide very rich and useful information for environment perception. They are not a silver bullet for perception, though they are probably the most applied sensors with robots. In the presence of heavy occlusion or for large distances, a stereo camera pair does not provide any additional information than a single camera does. In addition, when working in low light, passive cameras are less applicable than laser scanners.

The most applicable and general object recognition methods are based on recognizing SIFT features, color, or shape of the objects. Each method works well with some objects but might not work at all with other objects. For example, SIFT features are good for recognizing solid objects with colorful covers, while they provide much less information when viewing large single-colored surfaces. Color recognition can be used in these cases, but they may not be the best choice for

recognizing objects that get dirty easily and thus change color. In addition, color recognition may suffer from variations in illumination. It is generally impossible to define one recognition method that would outperform all the others.

A reliable recognition is a very difficult problem in general. Many of the approaches described only rely on automatic recognition, but the practical experimental cases are usually greatly simplified. The external disturbances are minimized to allow the algorithms to work well. A robot working in a realistic environment has to allow human assistance, because sooner or later it will be needed in any case.

An alternative to object recognition is to only provide the cognition model with essential parameters and not worry about the recognition itself. In [Nagata et al., 2010], the robot did not really recognize the target object, but the user only marked the corresponding part of the sensor data to allow localization. The specific information required for the manipulation, such as grasping mode, was inputted by a human.

The assisted recognition and the inputting of the cognitive information can both be considered as part of the shared cognition between a human and a robot. While the robot provides its sensor data, the human provides cognitive understanding.

All the approaches studied limit the representations for object description and modalities to provide that information. Some approaches use model-based recognition of objects, while others learn to recognize objects using the sensor data of the robots themselves. None of these approaches allowed multiple modalities for recognition. In a real case, it would be crucial to allow a user to describe objects using different representations. Sometimes it is not possible to show the robot what an object looks like. In these cases, a description like "blue box near the location (X,Y)" could often be descriptive enough to unify the target. Then again, sometimes it is impossible to describe an object using simple attributes such as color. For example, many objects are distinguished by their shape. In these cases, the appearance could be taught using sensor data. The most difficult cases are objects whose appearance does not necessarily unify them. An example is a pile of sand surrounded by more sand. All of the sand looks the same, but only part of the sand forms the pile. The pile can be recognized by measuring its shape, but as soon as part of the sand is taken away from the pile, the shape changes. In this case, the pile should be defined by its location.

To cope with various kinds of tasks, a unified approach for forming a cognition model is needed. Its applicability should not be limited to a specific task, but it should be expandable to different domains. As it is virtually impossible to take into account all the specific details that the cognition model would need to cope with, the description should not go into too much detail (e.g., limiting the algorithms used), but it should concentrate on high-level elements and their interrelationships. In addition, the model should allow using various kinds of sensor data and not be limited to specific sensor types. Before designing a complete stand-alone system, a conceptual model of the cognition model and its building blocks needs to be formed and verified.

Chapter 3

Proposed model for robot's perception and cognition

3.1 Introduction

The objective of the research is to enable a robot and a human to collaboratively build a cognition model, that is, a model of the environment that not only indicates where the objects are, but also what their classes, identities, and states are. Building the model is based on human input and the robot's sensor information. As a result, the robot recognizes the objects it perceives with its sensors, and it is able to share their properties with the human using the shared cognition.

The human can assist in building the cognition model by defining which parts of the robot's sensor data represent the objects. In a way, a human inputs part of his or her cognitive understanding to the robot's cognition. After this, the robot is often able to gather more information by itself.

This chapter presents a shared cognition concept that enables the cognition model of the robot to be built in cooperation with a human. In addition, the model permits recognition and learning that is based on prior knowledge of the objects.

The model abstracts the perception and cognition of the robot to separate but interconnected concepts. A more exact definition of robot perception and cognition as they are considered in this model is given below. The advantage of the separation is that the cognition is not directly dependent on perception, but different modalities can be used to exchange data with the cognition model.

The concept presented in this chapter is illustrated with four example cases: recognizing and measuring a shape of a sand pile, locating a box whose appearance has been defined numerically, indicating a target using only robot's sensors, and detecting and locating pieces of litter on the ground.

3.1.1 Perception, cognition, and recognition

The first step towards the cognition model is to define more closely what robotic perception and cognition are, and how they are modeled and connected to each other through recognition. The discussion below is illustrated in Figure 3.1.1.

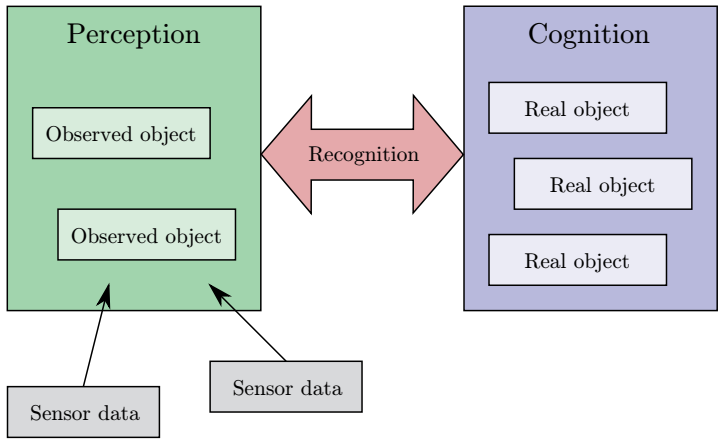


Figure 3.1.1: Observed objects represent the perception of the robot. The objects are created from sensor data. Real objects represent the cognition. Perception and cognition are connected to each other with recognition.

Perception is the input from the robot's sensors. However, perception is a higher-level concept than just sensor data. It represents information that is extracted from the sensor data. Preprocessing phases, such as *segmentation* and *feature extraction*, transfer the sensor information to *observed objects*. An observed object is a piece of sensor data that represents a separate physical entity. An observed object includes information on what the corresponding physical entity looks like to the robot.

Cognition is understanding of the perception. Cognition represents interpretation of what the robot perceives. It is the robot's understanding of the *identities* and *classes* of the entities. The physical entities are represented by *real objects* in the robot's cognition model.

The process that connects perception and cognition is called *recognition*. It compares the perceptual information with knowledge of the appearances of objects, and creates links between the observed and real objects.

3.1.2 Shared cognition

The abstraction of perception and cognition to separate elements enables the cognition model to be shared between a human and a robot. A human can also access the perception information by observing its visualization, but he or she cannot directly input anything into the robot's perception. Cognition, however, is a higher-level concept. The robot's cognition can be designed in such a way that a human can access it by both observing and modifying it. This is called *shared cognition*.

Affecting the cognition can be done in various ways. The factor common to all approaches is that a human helps in interpretation of the robot's perceptual information as cognitive information. The human can define the boundaries of objects seen by the robot, name them, or indicate which one is the target of the current task. Thereby the human transfers his or her cognitive understanding to the robot's cognition.

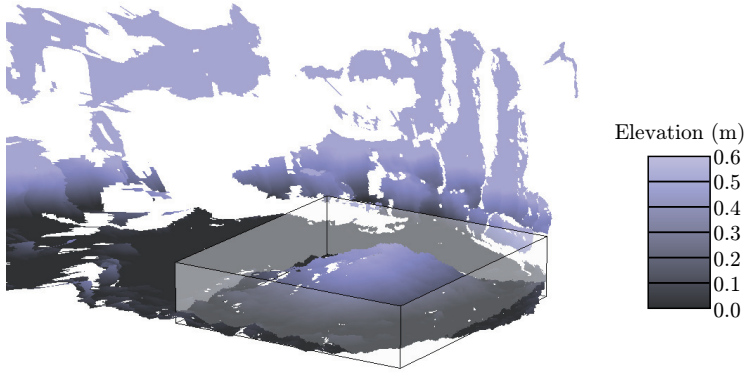


Figure 3.1.2: Robot's sensor data and a segment of it defined by human. The human has now defined a pile of sand as an object from robot's perception and transferred his of her cognitive understanding of the physical entity to robot's cognition.

3.1.3 Example cases

The first example of shared cognition is a case where a human marks a segment of a 3D point cloud to instruct that it represents a separate entity. In the cognition model, the object is defined as the volume inside the marked boundaries. It is defined using the spatial location. In practice the object represents the solid matter inside the boundaries. When the robot knows which part of its sensory information represents the physical entity in question, the robot can learn about its appearance on the basis of this segmentation. This example is depicted in Figure 3.1.2.

The second example presents a more traditional object recognition case where the user has defined the appearance of the object to the robot using numeric descriptions, such as hue of the color and size. The robot uses this description to identify the object as soon as it encounters an object that matches the criteria.

In the third example of shared cognition the user describes an object class called "litter". The definition of a piece of litter is that it is something that is on the ground but does not look like ground. The user does not describe the actual appearance of a piece of litter, not the location(s), and not even the number of the pieces of litter. After the robot has found candidate pieces, the user can help the robot by ruling out the false matches. Figure 3.1.3 shows an example of this case.

In the fourth example the shared cognition approach is evaluated in a box-carrying case. The user indicates a target object to the robot using real-world pointing methods. The human assists the robot in indicating the target object among many alternatives, therefore also this case represents shared cognition. Figure 3.1.4 shows a real example case where a human indicates the target object to the robot using this method.

The experimental validation of the proposed concept is based on these four cases. The actual experiments and the results are described in Chapter 5.

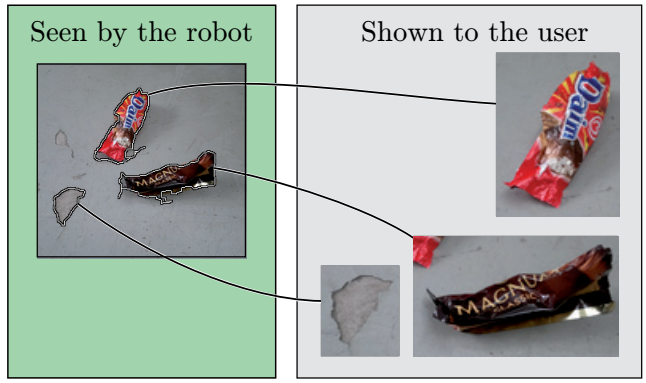


Figure 3.1.3: Possible objects segmented from robot's sensor data. The images of the objects are shown to the user, who then defines which ones really are the objects in question, and which ones are just false matches detected by the automatic segmentation algorithm.

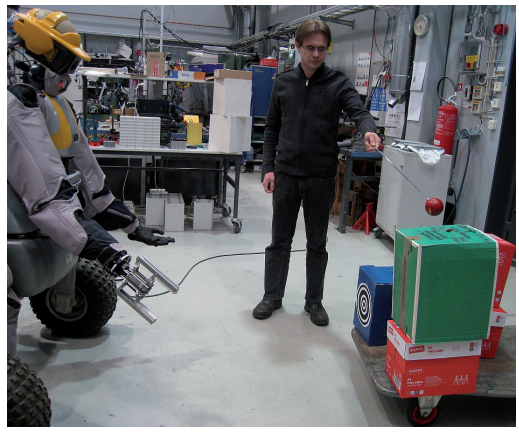


Figure 3.1.4: Human using a pointer stick to indicate which one of the detected boxes is the target one.

3.2 Requirements for the model

The above description assumes that real objects contain some description that can be used as a basis for the segmentation and recognition of objects. The object is initially perceived and processed by human cognition. In the model the description of the robot's cognition can be made in several ways. Basically, the different approaches are divided into two main categories: presenting a preprocessed description to a robot, or instructing the robot how the description is extracted from its own perceptual information. The former is a traditional approach to object recognition, and is based on a given database of features. It is a bottom-up approach, where the robot's understanding of the world is based on a description of how to interpret the perception. The latter represents a top-down approach, where the cognitive information is presented to the robot and it tries to learn the description itself. The model described in this thesis integrates these two approaches while concentrating on the latter, that is, the top-down approach.

The structure of the model is designed to answer the following two questions.

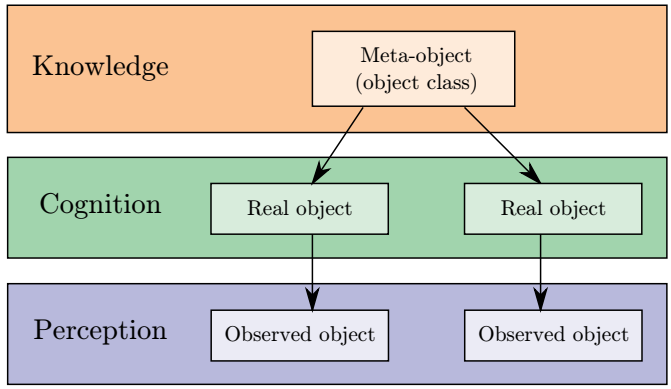


Figure 3.3.1: Meta-object represent the knowledge of the object classes, that is, the features that are common to all the objects of a certain type. Real objects are instantiations of real life entities in robot's cognition. Observed objects are extracted pieces of perception information that represent presumedly separate physical entities.

1. How can real and observed objects be matched on the basis of perceptual information and prior knowledge of the appearances of the objects?

2. How can the appearances of objects be taught in a way that permits both pre-defined description and learning about the appearance using the robot's own sensors in cooperation with a human?

By using the proposed concept, a cognition model of the robot's environment is built. As discussed earlier, the following three types of information are required from the cognition model:

1. locations of objects

2. types and identities of the objects

3. the objects' spatial states

With this information the robot will be able to cope with various complex tasks in real-life scenarios.

3.3 Components for building the cognition model

The model is based on dividing the "robot's mind" into three conceptually different blocks: perception, cognition, and knowledge. As described above, perception represents what the objects look like, and cognition describes how the robot interprets the perception. *Knowledge* represents a yet higher-level concept related to *object classes*. Cognition is built from real objects which are instantiations of real-life entities. Object classes represent groups of real objects each belonging to the same class. Object classes are represented by *meta-objects* that contain the information related to the class. Figure 3.3.1 shows an example case with two observed and two real objects. Both the real objects are of the same class, and therefore they are based on the same meta-object. More thorough explanation on objects is presented in Section 3.4.

3.3.1 Automatic and assisted segmentation

Extracting observed objects from sensor data requires the segmentation of the data. In practice, this means dividing the sensor data into blocks that are each presumed to represent a separate object. The segmentation can be automatic, using prior information on the objects and heuristic rules for segmentation, or human-assisted, using user input to determine the segmentation. After segmentation, the spatial location of each segment can be determined. The location can either be measured accurately on the basis of 3D sensor data, or it can be estimated on the basis of 2D measurements, such as a monocular camera image. Combining 3D sensor data and a camera image produces rich 3D data that contain even more information for segmentation.

As a result, the segmentation fulfills the first requirement of the cognition model, namely the locations of the objects.

3.3.2 Recognition

Object identification occurs when real and observed objects are connected to each other. The robot understands that an observed object represents a unique physical entity that it knows already. Another type of recognition is *object class recognition*. In this case the robot understands that an observed object belongs to some specific object class, but the object cannot be identified uniquely.

An example involving object identification is the case where the robot sees the pile for the second time. It has previously acquired enough information and is able to uniquely identify it when seen again. On the other hand, object class recognition occurs when the class "piece of litter" is defined. When the robot encounters a potential piece of litter, it is recognized with object class recognition.

Object recognition is not based on one type of sensor information only, but it can be based on the appearance of an object in images, 2D data, 3D data, and rich 3D data (a combination of 3D data and image features, described in Section 4.1.1.2, and shown in an example in Figure 3.3.2), as well as their locations, orientations, and motion information. In addition, object recognition may need to take into account the physical structure of an object, the relative spatial configurations of its components, and the shapes of deformable objects.

Recognizing an object is always based on the information gained through sensors, prior knowledge, and information inputted by the user. Still, all these recognition modalities introduce uncertainty. The recognition can never be considered absolutely sure. Therefore, the recognition needs to take into account the certainty of each recognition modality and include this in the model.

Object recognition can be divided into two subtasks: *feature extraction* and *pattern recognition*. In feature extraction, higher-level information is extracted from sensor data. In pattern recognition the features are matched against known features that exist in the database.

Recognition is the key to the second requirement of the cognition model. Recognition answers the question of what objects there are.

3.3.3 Object's spatial states

In addition to recognition, the location, orientation, shape, and motion components may need to be determined. Some of the parameters, such as location and speed, can be directly drawn from the recognition result. The determination of the shape of a deformable component or the relative orientations of the components of a multi-part object need the more thorough modeling of the

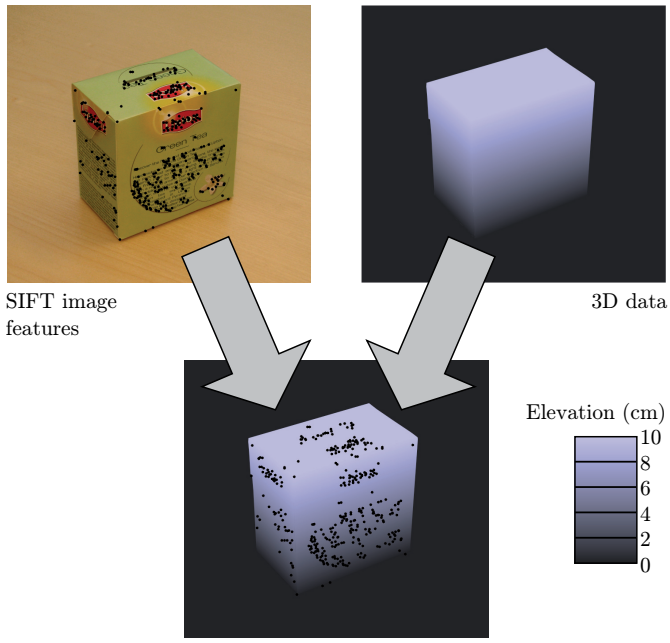


Figure 3.3.2: Rich 3D data is created by combining image features with their 3D coordinates.

object. For this reason, it is necessary to combine structural and perceptual information, as well as to determine the kinematic transforms between the components of a multi-part object.

An example of determining object's spatial state is measuring the sand pile. The robot uses its 3D vision capabilities to measure the shape in addition to the appearance of the pile. This shape information can later be utilized for example in planning how the pile is scooped with a bucket attached to the robot.

Determining the spatial state of an object is the third requirement of the robotic cognition model.

3.3.4 Teaching

Teaching an object to a robot basically occurs through real objects. A user or another agent inputs information into real objects that can then be used for recognizing the corresponding physical entities in the real world. The information can either be directly inputted to the real object, or a cue for interpreting the perceptual information can be given.

An example of direct input is a case where the user describes the size of an object, or defines its color from a color map. With this information the robot can then identify the object. Other examples are photographing an object to describe its appearance, or describing its shape, size, or surface texture. More complex methods of describing an object are its CAD model or mathematical edge model.

An example of perception interpretation information is a case where the user chooses a segment of the robot's sensor data and labels it as an object, as illustrated in Figure 3.1.2. The robot extracts features from the segment and associates them with the corresponding real object. After this, the

robot can observe the real-world entity more closely and gather more appearance information.

Teaching connects the human to the robot's cognition. As explained in the above examples, the proposed model enables a shared cognition to be created between a robot and a human.

3.4 Objects

The three aforementioned object types, observed, real, and meta-objects, form the basis of the model. Observed objects collect perceptions related to objects. Real objects represent physical entities in the robot's cognition. Meta-objects represent the classes of objects; that is, they describe features common to all objects of a certain type.

3.4.1 Observed object

The sensor data of real-world objects are stored as *observed objects* which represent the robot's *perception.* The observed object itself does not represent any physical entity in the robot's cognition, just observations related to it. If an observed object is recognized, it is connected to a real object as described below. In that case the observed object represents the sensor data of that real object. If an observed object is not recognized, it represents an *anonymous object*, that is, an object that is perceived but whose identity is unknown. In addition to sensor data, an observed object also contains the location and time when the data were acquired.

In the example cases, observed objects are created through segmentation. In the sand pile case, an observed object represents the segment of the sensor data that falls within the boundaries of the region defined by the user. In the user-assisted litter recognition -case, observed objects are created on the basis of the regions of the automatically segmented image. In the box case, an observed object represents the part of the image that has the predefined appearance characteristics, for example color. If the human uses a physical pointer to indicate the objects, the pointer segmented from the image also represents an observed object.

3.4.2 Real object

The actual *cognition* is represented by *real objects.* They represent *named physical entities*, that is, entities that have unique identities. Real objects contain all the available useful information on the corresponding entities. The information can be perception features, structural information, location, and miscellaneous metadata.

Real objects can be created in two ways. First, if a robot observes an object of a known class, and no real object has yet been associated with it, a new real object is created. An example of this type of real object creation is locating pieces of litter. The robot does not initially know the locations and not even the number of the pieces of litter. When a potential piece of litter is found, a corresponding real object is created. Alternatively, a user or a similar agent can create a real object. Information can be inputted to the real object either by the robot's perception or by a human. This information is then used for recognizing the actual object. An example of the latter type of creation is describing an object using some specific measures, such as location, color, size, or preferably a combination of all of these. The robot knows that there is one specific entity present, and its parameters are represented by the real object.

When a user updates information about a real object, it is possible to describe the appearance, location, or both for the corresponding entity. Therefore it is possible that the robot knows the

appearance of an entity, but does not know where it is. If the robot encounters the entity and recognizes it, the robot then also knows its location. As an opposite case, it may also be possible that the robot knows where the object is, but does not know anything else about it. For example, if the user described the boundaries of the sand pile on a map instead of using robot's sensor data, the robot may not yet see the pile but is aware of its location, and is able to recognize it on the basis of its location as soon as it arrives near the location.

The information content of a real object is based on what data the robot has been able to gather that are related to the object. It is possible that some of the information may be wrong. After all, a real object is just the robot's cognitive representation of an object, and it contains the best available information that describes the object. If there is wrong information, a real object can sometimes be associated with a wrong physical entity as a result of false recognition. The techniques presented in this concept try to ensure the information contained in a real object is as correct as possible.

Functional features In addition to the visual and measurable properties of the physical entities, real objects can also include non-measurable features, such as weight, fragility, or even instructions on how the corresponding physical entity is used. These attributes don't affect the recognition but they carry information that is useful for task execution. If such functional features are included to a real object, then as soon as the robot recognizes an object, it already knows how to handle it. This enables the information sharing between the robot and the human in a detailed level.

3.4.3 Meta-object

Meta-objects represent *knowledge* of object classes. These objects do not represent any named physical entities, but just general knowledge of all the objects of a certain type. In other words, a meta-object represents all the knowledge of a physical entity before it is actually seen. This knowledge can be used to recognize the classes of the entities that are perceived.

In practice, meta-objects may contain either specific pieces of information, such as "ball-shaped", or range information, such as "size between 1 and 2 meters". More generally, meta-objects contain extracted features from data acquired from several typical objects of a certain class. In addition to the feature information, accepted uncertainty ranges are stored.

A description of a piece of litter is a typical meta-object. The description may contain for example that the piece is something that is located on the floor and surrounded by an edge. If the floor is known to be sufficiently evenly colored, it does not contain any visible edges, but if the color of a piece of litter is not exactly the same as the floor, the piece can be detected on the floor.

Meta-objects generally do not contain position or orientation information, because the meta-objects are not associated with any physical entity.

The structure of a meta-object is generally similar to that of a real object. Basically, mera-objects are like real objects, but they just lack the specific location and identity information. Also meta-objects can include functional features that describe typical non-observable characteristics to all the objects of the corresponding class.

3.4.4 Definition of objects

The basis of an object is always created by a human. The definition of an object is based on one or more physical entities. The human transfers his or her understanding of the corresponding physical

entity to the robot by creating a real object or a meta-object. These objects are further used as a basis of creation of observed objects. In other words, objects are instantiations of the real-world entities in robot's cognition, but their representation is defined by a human thus utilizing human's cognition.

3.5 Structure of the model

Fulfilling the requirements of a cognition model requires a concept with the above components. The creation of observed objects from sensor data is handled, associating the observed objects with real objects and meta-objects with recognition, determining the spatial states of the objects, and teaching the model. The following subsections cover these aspects.

3.5.1 From sensor data to observed and real objects

A simplified case of one observed object and one real object is presented in Figure 3.5.1. The figure represents relationships of sensor data, real and observed objects, what data the objects contain, how they are interconnected, and in which phases a human can be a part of building the cognition model.

In the first phase, the sensor data are segmented to represent separate physical entities. As a parallel process, features are extracted from the sensor data. Observed objects are composed from the appearance features associated with the segments. In addition, raw sensor data related to an object can be stored to an observed object. Location and orientation information can also be extracted from the sensor data features and recognition result. Finally, the observed object stores a timestamp relating to when the corresponding sensor data were gathered.

In this example, the cognition model already contains a real object that is matched to the observed object with recognition. A probability is assigned to the match, representing the uncertainty of the recognition. The information from the observed object is processed and copied to the real object. The pieces of information that are immediately affected are the location and structural state. The location is updated simply on the basis of the observation. The spatial state of the object can be updated on the basis of the direct interpretation of sensor data and reasoning on the basis of the extracted features. The spatial state represents, for example, the spatial configuration of the components of a physical entity consisting of multiple parts.

In time, the model learns about the appearance. The more data that are gathered, the better the robot knows what the object looks like. Appearance features are copied from observed objects to real objects when similar features are detected several times.

Other cognitive information is anything that is related to other aspects than perceiving the object. For example, it could include information on what a machine is doing, or how a box should be grabbed to avoid breaking it.

3.5.2 Sharing the cognition with a human

Various parts of the model can be affected by a human. The human figures in Figure 3.5.1 show which parts of the model can be affected by a human.

Segmentation and recognition can be assisted by showing which part of the sensor data represents a certain physical entity. In the sand pile -example, the user defined which part of the sensor data represents a certain physical entity. The user instructs the robot that the region needs to be

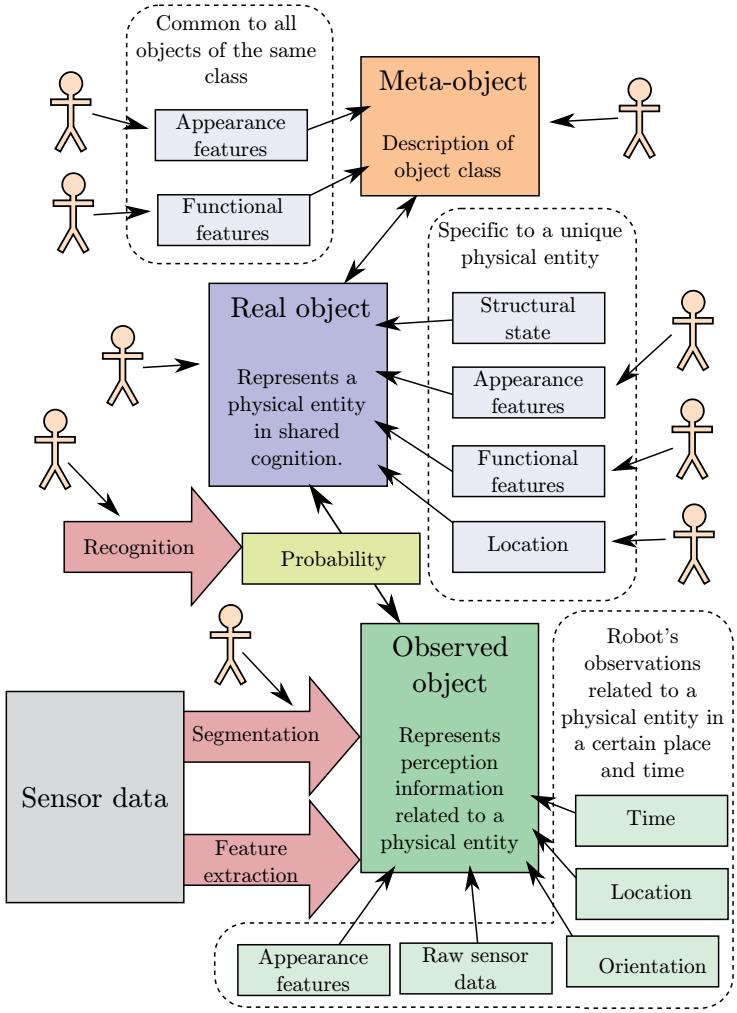


Figure 3.5.1: Components of the system. An observed object consists of features extracted from the segmented part of the sensor data. A real object is connected to the observed object with probability that describes the uncertainty of the recognition process. A meta-object describes features common to all objects of a class. Segmentation and recognition can be assisted by a human. In addition, human can create real and meta-objects and input various types of data them.

segmented to a separate observed object. Then the human forcibly connects the observed object to a real object in the robot's cognition. The human has therefore made a cognitive link to the robot's cognition by using his or her understanding of the world. Figure 3.5.2 extends Figure 3.1.2 by showing how this case is handled in the model.

Another way for a human to affect the robot's cognition is to directly modify real objects. A human can describe the appearance and location of a real object to allow the robot to recognize the object when it is encountered. These mechanisms directly modify how the robot understands the object. As an example, a human can describe the color of an object to the robot by modifying the

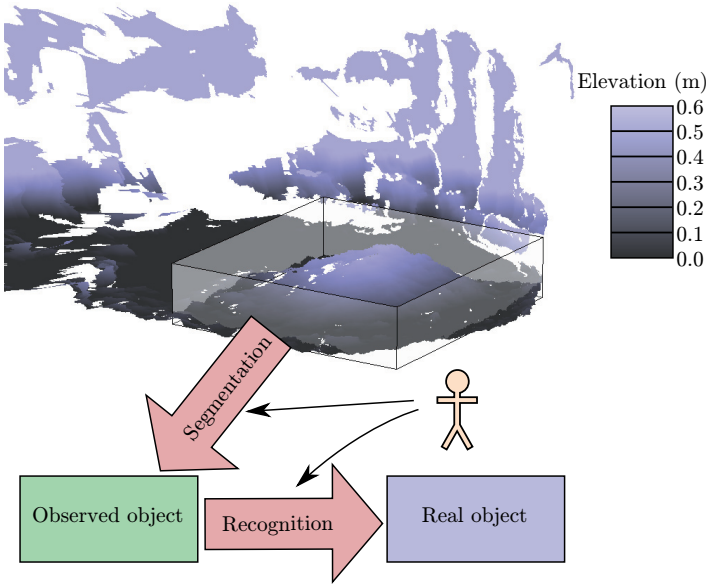


Figure 3.5.2: Transforming a human defined segment of robot's sensor data to observed object and matching that to real object. This example visualizes 3D data from stereo cameras viewing a pile of sand that is extracted from the data.

color parameter of the corresponding real object in the cognition model. This can be considered as teaching the robot. In addition, the human can affect the functional features that describe other cognitive information that is not part of the recognition process.

3.5.3 Dealing with uncertainty and ambiguity

Matching perception to cognition always introduces some uncertainty. Two different objects may look alike, especially when observed with sensors that do not provide enough appearance information. An example of such a sensor is a simple laser scanner that only measures a 2D plane and does not provide reflectance information. In addition, recognition algorithms may be imperfect in various ways. Feature extraction usually reduces the amount of information in sensor data, and this may lead to ambiguities. Therefore it is necessary not to assume that the recognition result is always one-to-one, but to allow one-to-many and many-to-many links in the recognition process. In addition, the connections should include information on how certain they are.

Figure 3.5.3 presents a case where multiple observed and real objects are connected to each other. In the figure there are also some objects that were not matched to any of the objects. Unmatched real objects just represent physical entities that have not been observed with any sensor. Unmatched observed objects, in turn, represent observations whose identity is unknown. They are called *anonymous objects*.

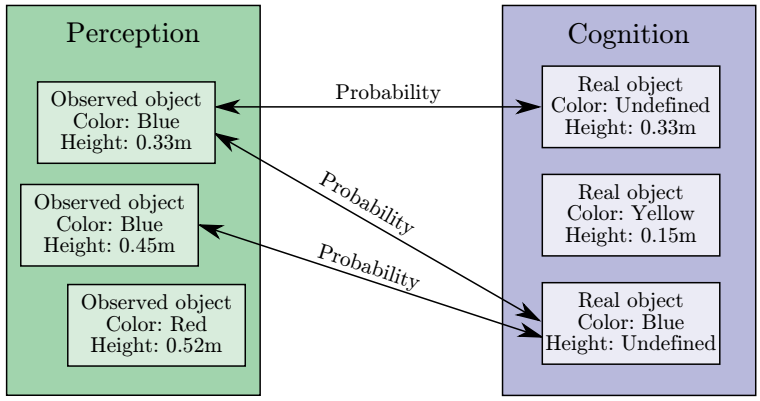


Figure 3.5.3: Real and observed objects matched. Each real object is matched to every observed object that looks enough like the corresponding real object. This may lead to situation where many real objects are linked to one observed object, and in turn, one real object may be linked to many observed objects, if they look similar enough. Probability of the links denotes the reliability of the matches.

Chapter 4

Building and using the cognition model

This chapter presents the principles and techniques of building and using the cognition model presented in Chapter 3. Section 4.1 describes how the robot's sensors are used in building the model. Section 4.2 describes how a human can share the cognition model with a robot. Section 4.3 describes how the cognition model can be utilized in task execution.

4.1 Using robot's sensors to build the cognition model

The cognition model represents the robot's understanding of its environment. Inputted sensor data are interpreted to elements of the cognition model. The first phase is turning the sensor data to observed objects. In the second phase observed objects are recognized against the database of known objects and object classes. Only the objects that are needed in the present application case are matched. Other objects are considered as obstacles.

4.1.1 Observations from sensor data

First, the sensor data are interpreted to observed objects that represent observations from the world using the robot's sensors. The phases are illustrated in Figure 4.1.1.

Automatic and assisted segmentation The first phase is segmentation. Data from different sensors are fused together on the basis of their relative positions and orientations. The segmentation of these data can be done on the basis of depth information, image edges, motion, or recognition of objects, or the segment can be defined by a human. If the user has defined an object from the sensor data by determining its location or features in the sensor data, this information will be used for the segmentation.

Appearance feature extraction The second phase is the extraction of appearance features from the sensor data. Appearance features describe various aspects extracted from the sensor data, for example the color or shape of an object, or they can be descriptions of the appearances and relative positions of local keypoints in an image or similar sensor data. Depending on the algorithm, feature extraction is executed either to segmented parts individually or to the whole

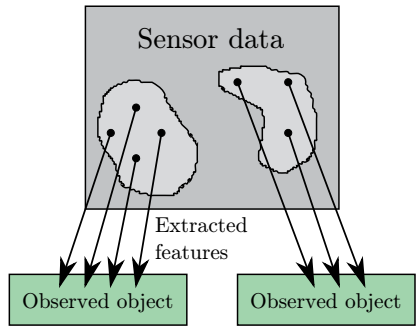


Figure 4.1.1: Sensor data are segmented and features are extracted. Observed objects are created on the basis of the result.

batch of sensor data. The extracted features are stored to a dynamically allocated array that is not yet associated with any observed object.

Creation or association of observed objects The third phase is the comparison of the location of the segmented region and the content of the extracted features to previously created observed objects. If the current observation is close to one of the previously observed objects, the current feature data are associated with that. If there is no observed object that matches the current observation, a new observed object is created, and the feature data are associated with it. All the available data of the features and object location are stored to the observed object.

Structural feature extraction The fourth phase is an optional extraction of structural features. Real objects contain information on whether structural features need to be extracted. This is described in greater depth in Section 4.1.1.2. The structure data consist of the orientations of the parts of the object and the shapes of parametrically defined geometric structures, such as elevation maps of the ground.

4.1.1.1 Segmentation of an object

Segmentation means dividing the sensor data into parts that belong to different objects. In practice, when working with a limited set of objects, it is not necessary to perform a complete segmentation of the scene, but just extract the objects of interest.

Recognition-based automatic segmentation The most reliable means of segmenting an object is based on recognition. If the class of an object is recognized, or if the object is identified, its appearance is already known, and it is easier to find it from sensor data. This applies mainly to image data or 3D or rich 3D data, because identification by means of range data produced by scanning sensors is ambiguous, and knowledge of the appearance of the object may not provide additional information when compared to depth-based segmentation.

Recognition-based segmentation does not fully conform to the model where first the segmentation is needed, and the recognition is done only after that. If the recognition is already done in the segmentation phase, the object recognition phase takes place simultaneously with segmentation.

Depth-based segmentation If available, depth is very useful information for segmentation. Physical real-world entities can often be distinguished by depth information. An exception is a case where the objects touch or are very close to each other. If no *a priori* information on the behavior of the objects is available, it may be impossible to distinguish the objects from each other.

Segmentation based on full 3D information can be improved with simple rules. For example, the ground must be distinguished from the other objects. Separate objects must remain separate even if the ground forms a physical connection between them. In addition, if the connection is very thin, it can be neglected. For example, if two people shake hands, their physical connection is very thin in relation to their total volumes. Depth-based segmentation with 3D data can therefore be performed on the basis of the following principles:

1. each object not touching other objects can be considered separate
2. the connection of the object to the ground is neglected
3. the connection of the object to another object through a thin connection is neglected. Classifying a connection as "thin" depends on the object class.

Depth-based segmentation can also be performed with data from a 2D scanning sensor. The sensor data are limited in the sense that they provide just a slice of the 3D world. The best assumption based on this data is that every part of the scan data that appears to be separated is considered as a separate object.

Edge-based segmentation In image data, edges often separate objects from each other. The edges can be found, for example, with Canny's edge detector [Canny, 1986]. The edge detection can be improved with post-processing operations, for example morphological dilation or closing. The image can then be divided into regions split by the edges.

The downside with edge-based segmentation is that it does not usually produce very good segmentation in practice. Edges can exist inside an object, in addition to the physical boundaries. The edges may be caused by the surface texture, shadows, or partial occlusion. In addition, separate objects may not have a visible edge between them. Many of these phenomena are unforeseeable, and may cause the segmentation to fail. Edges should mostly be used in conjunction with other techniques.

Motion-based segmentation The relative motion of the objects with respect to each other or to the observer is a very powerful cue for segmentation in the segmentation of an image. Motion in an image can be determined with optical flow, for example with the method developed by Lucas and Kanade [1981] and further refined by Bouguet [1999]. The problem with motion-based segmentation is that different parts of one object may move at different speeds and even in different directions with respect to the observer. The other problem is that not all the changes in images are caused by motion, but also by changes in illumination, or even the appearance of the object itself. An example of the latter case is a machine with a blinking warning light.

Motion can also be used as a means of segmentation with depth information. If different movements can be identified, there is a strong probability that the regions belong to different objects. This assumption may fail in cases when the object is not rigid, or when parts of an object occlude and reappear.

Location-based segmentation If the location of an object is known in the sensor data, segmentation basically just involves extracting the corresponding piece of data. However, the robot may be equipped with multiple sensors, and the location of the object may only be known in the coordinate frame of one sensor. In this case the known object boundary coordinates need to be projected to the coordinate frames of other sensors. This requires a model for inter-sensor transformations. This is usually determined with a calibration procedure, for example with the procedure proposed by Underwood et al. [2010].

Another problem with location-based segmentation is caused by the movements of the object and the robot itself. If the robot moves, the coordinates in the sensor data can be updated on the basis of navigation data. In addition, the localization of the region can be improved by tracking it in the sensor data. In the case of a moving object, the location changes, and therefore other means, such as recognition-based segmentation, need to be used.

Human-assisted segmentation Sometimes it is not possible to perform the segmentation of sensor data automatically. In this case, human assistance may be needed. A human can determine manually which parts of the sensor data belong to different objects. It is usually enough to perform the segmentation on the basis of only rough outlines, the bounding boxes of the objects, or even one point of interest. When one point is determined, the surrounding area in the sensor data is also considered. In an image, a circle around a point is interpreted as being part of the object. This may lead to a case where the object is not recognized as a whole, but only a part of it is recognized. If more accurate segmentation is needed, then edges can be drawn. After human-assisted segmentation, the recognition can either be done automatically or assisted by a human.

It is not feasible to assume that the user performs the segmentation of the sensor data continuously. After the segmentation and recognition have been performed, the segmentation can later be performed on the basis of recognition or the location of the object.

Choice of the segmentation modality Choosing the segmentation technique requires understanding the constraints of the task and the environmental conditions at hand. Therefore it is not usually possible to do this automatically, but the choice needs to be made by a human. If the appearance of the object can be defined well, then a recognition-based segmentation is usually the most effective modality. On the other hand, if the location of the object is the most accurate information, then location based segmentation is a natural choice. In cases where the appearance or location are not known accurately, edge-, motion-, or depth-based segmentation modalities can be used depending on the sensors. In situations with no information enabling automatic segmentation, then human-assisted segmentation can be used. The human input can be handled in a similar way as the outputs of other segmentation methods, and therefore the human-assisted segmentation can be used interchangeably with other modalities.

In this research, recognition-, edge-, and location-based segmentation modalities, as well as human-assisted segmentation are used as a part of experimental evaluation presented in Chapter 5. The recognition-based segmentation is used in cases where the appearance is defined with local image features or color. Edge-based segmentation is used in a case where the objects are known to be distinctive from background, and user interaction is provided to validate the result. Location-based segmentation is used after human-assisted segmentation in a case where the location of a vaguely defined object, a sand pile, is segmented from a sensor data.

4.1.1.2 Extracting information

After segmentation, appearance and structure information are extracted from the sensor data.

Appearance features Features are used for recognizing the object or its parts. Several different feature types can be extracted from the sensor data. The extraction can be done either during or after the segmentation step.

The extraction of some features requires heavy preprocessing of the sensor data, and it is often feasible to execute the extraction for the whole set of data instead of smaller subparts one by one. Features extracted in this way can later be divided into individual objects. This kind of feature extraction requires the features to be local. One feature should not span several objects. An example of features that should be extracted in this way is SIFT and SURF features. They require heavy filtering of the image, and usually it is much faster to process the whole image just once instead of piece by piece.

The other way to extract features is to do it after the segmentation step. Some features do not require heavy processing, but instead they depend on the boundaries of the region. An example of this type is a color histogram. The histogram is calculated to a relatively large area, which should correspond to the actual object. The histogram cannot be calculated before the division of the image.

Structure Extracting the structure information means measuring the physical properties of an object from the sensor data. There are various aspects related to this: measuring the orientation and location of the object; measuring the relative poses of the subparts of a multi-part object, and measuring the shape of a parametric geometric model, such as an elevation map. The location and orientation information of objects and their subparts is acquired as a part of the recognition process, and therefore the structure measurement phase consists only of measuring the parameters of a parametric geometric model.

Determining the structure requires 3D data from the sensors. Estimating the shape of an elevation map or another non-rigid object is not possible from 2D data only, and therefore 3D data are needed. An algorithm for extracting an elevation map from 3D data is presented in Subsection 4.1.1.3.

Combined appearance and structure It is often possible to combine 3D structure data with appearance features from, for example, an image. This rich 3D data can later be used for more reliable post-processing than plain appearance or structure data. To calculate rich 3D data, 3D information is needed at the points where appearance features are calculated. In practice, the 3D coordinates are projected to an image where the features are calculated, and the closest coordinates are found using interpolation. This procedure is explained in Algorithm 4.1.

4.1.1.3 Algorithm for measuring an elevation map

Measuring an elevation map on the basis of 3D data basically requires projecting the measured 3D point cloud to ground plane and calculating the elevations with respect to the base level. This procedure basically consists of two steps. First, the ground orientation needs to be normalized because the 3D point cloud acquired for example with stereo cameras is not necessarily aligned along the ground level. The nominal ground level is usually simultaneously approximated. The

Algorithm 4.1 Calculating rich 3D data

This algorithm is used for calculating rich 3D features based on image features F calculated from image I, and array of 3D coordinates (x_{3D}, y_{3D}, z_{3D}).

1. For each coordinate tuple (x_{3D}, y_{3D}, z_{3D}) that does not yet have corresponding image coordinates (x_I, y_I) calculated
   (a) Project the coordinates (x_{3D}, y_{3D}, z_{3D}) to the image plane to coordinates (x_I, y_I).
2. For each feature F
   (a) Find the group of closest projected image coordinates (x_I, y_I)
   (b) Interpolate the 3D coordinate for F with the 3D coordinates corresponding to the chosen image coordinates
   (c) Store the 3D coordinates to the same datastructure with feature vector F.

second step is to calculate the distances of the 3D points from the ground level, and to project these distances as elevations to the elevation map.

The first phase, ground-level alignment can be done with various approaches. V-disparity approach [Labayrade et al., 2002] is a method designed to measure road profiles on the basis of stereo image pair. Its strengths are at detecting discrete obstacles and its computational effectiveness. Another possibility is to fit a plane to the 3D point cloud. This assumes that the ground can be seen in the data as a relatively large plane. Further assuming that there are no other large planes in the 3D data, the found plane can be considered as the normal level of the ground. Its orientation with respect to the sensors denotes the coordinate system of the ground related to the sensor coordinates. Two main techniques used for fitting the plane are Hough transform [Vosselman et al., 2001] and RANSAC algorithm [Se and Brady, 2002]. They both result in parameters of the estimated plane. As discussed by Tarsha-Kurdi et al. [2007], RANSAC algorithm is in general more effective for fitting planes from 3D data, and therefore it is chosen for the purpose.

The following description is an adaptation of the RANSAC algorithm as described in [Se and Brady, 2002]. The algorithm requires a relatively large number of measured 3D points. This can be acquired with a relatively wide-angle stereo system or with a 3D laser scanner. The ground plane is found with the RANSAC algorithm, which fits the plane equation

$$ax + by + cz + d = 0 \tag{4.1.1}$$

to the data set. After the plane has been found, the 3D data points are translated and rotated with

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} c_\alpha & s_\alpha & 0 \\ -c_\beta s_\alpha & c_\alpha c_\beta & s_\beta \\ s_\alpha s_\beta & -c_\alpha s_\beta & c_\beta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix}, \tag{4.1.2}$$

where c_x is the short form of $\cos(x)$ and s_x for $\sin(x)$, x, y, and z are the original coordinates, x', y', and z' are transformed coordinates, and h is the height of the ground plane from the XY-plane.

The parameters are calculated with the known plane equation coefficients

$$\begin{cases} \alpha &= \frac{b}{c} \\ \beta &= -\frac{a}{c} \\ h &= \frac{d}{a}. \end{cases} \tag{4.1.3}$$

As a result of the ground level alignment, the 3D data is now transformed to the same coordinate system as the ground plane. Therefore, the x-, and y-coordinates of the 3D data now represent the corresponding coordinates of the ground plane and the z-coordinate is the height from the ground plane. The second phase, coordinate projection, is basically just associating the x- and y-coordinates with the corresponding bins in the elevation map, and determining the elevation on the basis of the z-coordinate. However, there are two problems. There is usually more than one point associated with each elevation map bin, and there are often outliers in the 3D data. Oniga et al. [2007] chose the highest of the points that falls into the elevation map element. However, outliers in the data may cause high peaks on the elevation map if the highest of the 3D points is chosen. To avoid this, all the points that are associated with the same bin are grouped together and analyzed. There is usually a condensation of points around ground level, while the outliers are just individual points or small groups. The most effective way is to build a linked list for each elevation map bin and then analyze the points on the basis of that. The points in the list are sorted and the condensed set is found by sliding a window through the data and looking for the group with the smallest standard deviation. The corresponding z- coordinate is chosen as the height on the elevation map.

Algorithm 4.2 summarizes the alignment of the ground plane and the projection of the 3D points to the elevation map. The ground plane alignment is a similar to the algorithm described by Se and Brady [2002]. The projection algorithm is developed by the author.

4.1.2 Object recognition

After the observed objects are created or features are associated with previous observed objects, the object recognition is executed. In practice, object recognition links observed and real objects to each other. Recognition is based on comparing cognitive information in real objects and perceived information in observed objects. The different types of object recognition and tracking are shown in Figure 4.1.2. The following subsections describe the recognition procedure. All of the described recognition modalities are used in the experimental validation in Chapter 5.

Matching the features to the database

The first phase of object recognition is matching the features of the observed object to a database of features from all the real objects and meta-objects. The features are indexed separately to speed up the matching procedure. Matching with real objects or with meta-objects is technically equal and can thus be combined in one matching operation. In addition to feature vectors, the physical structure model and prior information on the locations and orientations of real vectors are also used for recognition. The pose of the object affects which side of it is visible to the robot. This information can be utilized in determining the orientation. The physical structure also affects which parts of the object are visible.

As a result of the matching procedure, the observed object can be matched to either one or

Algorithm 4.2 Building an elevation map on the basis of measured 3D points.

The elevation map consists of an array $E(x_E, y_E)$ with discrete bins. The values of the bins correspond to heights. The calculation is based on an array of 3D points (x_{3D}, y_{3D}, z_{3D}). The algorithm consists of two phases: ground normalization and projection.

Ground normalization

1. Choose three points (x_{3D}, y_{3D}, z_{3D})
   (a) Fit a plane to the points, and determine the coefficients for Equation 4.1.1.
   (b) Count the number of 3D points in the plane.
   (c) Save the chosen points and the result of (b) to an array A.
2. Repeat phase 1 n times.
3. From array A
   (a) Choose the set of three points corresponding to the largest number of points in the corresponding plane.
   (b) Fit a plane using all the matching points, and determine coefficients a, b, c, and d.
4. Rotate the 3D points using equations 4.1.2 and 4.1.3.

Projection to elevation map

1. For each 3D point (x_{3D}, y_{3D}, z_{3D})
   (a) Calculate the coordinates of the corresponding element (x_E, y_E)
   (b) Add the point to the linked list associated with (x_E, y_E)
2. For each element $E(x_E, y_E)$
   (a) Sort the associated linked list on the basis of the z-coordinates
   (b) Run a moving window through the linked list
      i. Calculate the standard deviation of the z-coordinates for each window
      ii. Store the minimum standard deviation and corresponding z value
   (c) Choose the z-coordinate corresponding to the smallest standard deviation and store it to $E(x_E, y_E)$

several real or meta-objects. Probabilities based on the reliabilities of the matches are stored to the links between real and observed objects. The calculation of the probabilities is described in Section 4.1.3.

Object identification

An object can be identified in two ways. The observed object can either be connected to a matching real object, or the observed object can be connected to another observed object that, in turn, is connected to a real object. The two topmost diagrams in Figure 4.1.2 present these two cases.

If sensor data are associated with an observed object that is already associated with a real object, the connection between them is retained. If an observed object is not yet connected to any object, but there is a match between the observed object and a real object, the connection between them is established.

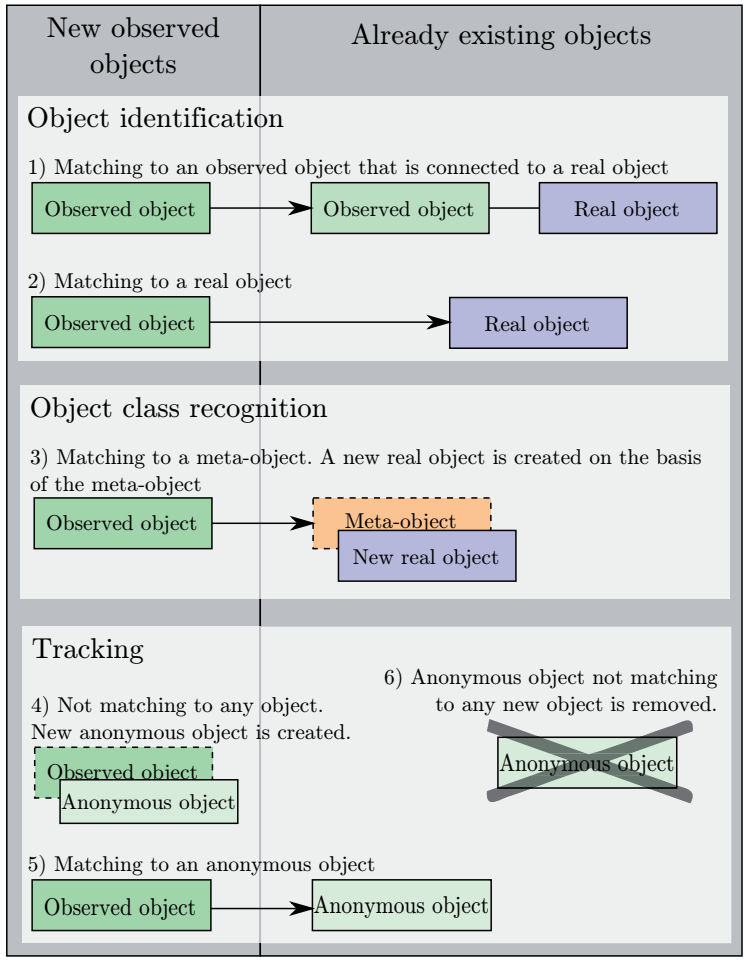


Figure 4.1.2: On the left side are new observed objects that are based on the newest sensor data, on the right side are real, observed, and anonymous objects that have been created before. The relationships between different recognition modalities and object types are shown in the numbered diagrams.

Connecting an observed object and a real object also requires spatio-temporal conditions to be fulfilled. If the spatial location of the real object is known, but the location of the new observation is not feasible, the connection is discarded. It is also possible that only a relative topological location is known. In this case, the feasibility is evaluated against the rules of topology. If the feasibility of the topological location is unknown, the connection is retained. An example of such a situation is a case where the only information on the location of the object is that it is indicated by a certain sign, but the sign itself has not yet been seen. In this case it is not possible to know whether the object is indicated by the sign or not, and the feasibility cannot be explicitly confirmed or rejected.

If there is a match between an observed and a real object, the possible matches to meta-objects are discarded, because object identification is much stronger information than just knowing its

class.

Object class recognition

If the features and spatio-temporal properties do not match any real object, but the class of the object is recognized, a new real object is created. The third diagram in Figure 4.1.2 presents this case. The current observed object is associated with it. Automatic generation can occur erroneously as a result of inaccurate sensor data or information on the positions of objects. Therefore the real object that is created will retain the information that it was automatically generated. It can be merged with another real object at a later time if it seems probable that the two objects actually represent the same physical entity.

Tracking an observed object

If the observed object cannot be matched to any real or meta-object, it is tracked. The observed object will remain unmatched to any real object, and it will be marked as an anonymous object. This means that there is no meta-information related to it, but the only information is the sensor data gathered from it. The diagram 4 in Figure 4.1.2 presents this case. If any anonymous objects already exist, the features of the current observed object are matched against those of the anonymous objects. This is presented by the diagram 5 in Figure 4.1.2 presents this case. If no observed object is associated with a previously created anonymous object, it is removed, as shown in the diagram 6 in Figure 4.1.2. In the event of unreliable sensor data, the anonymous object is not removed immediately, but only after being unmatched for a few rounds.

Tracking may be necessary in cases where the robot knows the target but cannot recognize it. If the user has marked the target in the sensor data, the robot may need to approach the object before it can really recognize it. Tracking is generally a relatively unreliable procedure, and errors can be accumulated, so tracking-based object localization cannot be used for planning manipulation where recognition of the target object is required.

4.1.3 Handling uncertainty

4.1.3.1 Match certainty

Some features are more reliable cues for recognition than others. For example, the color of an object is not a very unifying feature, but a set of rich 3D features can be used for much more reliable recognition. Another aspect affecting the match certainty is the match score. If the observed and reference features have a very close match, the recognition can be considered more certain than when the match is not as good. Further, the observation can be compared to available data, and the consistency can be evaluated. If there are many similar observations of a certain object, it is probable that a new observation is similar to the previous ones. If the new observation differs considerably, it may represent some other real object.

These three aspects, feature type, match score, and consistency, affect the probability of the recognition. This probability, here called *match certainty*, is associated to the link between the observed and real objects. Match certainty, written p_m, and based on feature-related certainty p_f, match score p, and match consistency p_c, is calculated simply with

$$p_m = p_f p_s p_c. \tag{4.1.4}$$

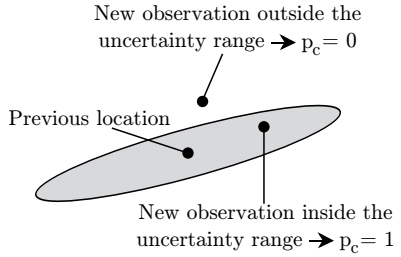


Figure 4.1.3: Observations outside the 2σ uncertainty region are considered to be too far from the previous location to represent observations of the same object, and therefore the p_c score is assigned value 0. Observations inside the uncertainty region get $p_c = 1$. This value is used in Equation 4.1.4.

Feature-related certainty can be determined on the basis of experience. It is usually not possible to determine an exact certainty related to a feature type.

The certainty drawn from the match score requires the score to be scaled to the $[0, 1]$ range. Generally, the match score of one feature is related to the Euclidean distance between the observed and reference features. With several features grouped together, the sum of the individiual scores, as well as the number of matches, also affects the score.

Match consistency p_c gets a discrete value of 0 or 1 depending on the result of the comparison of the current observation to the previous ones. The locations of the objects are compared. If the location of the new object is within the location uncertainty region of the other object, p_c is set to 1; otherwise it gets the value 0. The acceptable location uncertainty region is 2σ of the normal distribution covariance. The location uncertainty covariance is described in Section 4.1.5. Figure 4.1.3 shows the principle for calculating p_c.

4.1.3.2 Multiple candidates

Several observations may be associated with the same real object. Each match has its own certainty. In many cases, several candidates can be retained. Over time, one of the candidates may become more reliable than the others. If a decision needs to be made while there are multiple candidates, the one with the highest certainty can be chosen. If none of the candidates is superior to the others, an ambiguity may remain. In these cases, user assistance may be needed.

There may be multiple candidates for one object. All the candidates are included; that is, there may be links from one observed object to several real objects.

4.1.3.3 Choosing just one candidate

The most probable candidate is chosen to represent which observed object relates to which real object. The other candidates are also preserved in case the chosen cancidate was wrong. The candidate is chosen on the basis of the match probabilities. If the probability of the best match is more than double than the second best match, and the probability of the best match is more than 0.5, the candidate match can be considered as the best match.

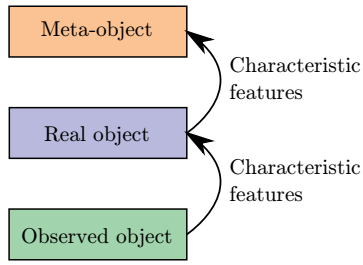


Figure 4.1.4: Learning occurs from observed objects to real objects, and from real objects to meta-objects. Only characteristic features are copied.

4.1.3.4 Comparison to other techniques

This kind of object tracking problem has been considered in robotics research, and various other solutions have been proposed. JPDAF (Joint Probabilistic Data Association Filter) [Bar-Shalom and Fortmann, 1987] deals with unrecognized measurements in space. It aims to associate new localized objects to previous ones even without knowing their identities. The association is done on the basis of the locations and velocities of the objects only. However, this kind of approach requires estimation of the velocities using a Kalman-filter. Such estimation would require constant measurements of the objects. The model presented in this research does not require tracking the motion, but can be applied on single measurements as well. Therefore the much simplified approach is applicable. The approach is used in this research uses the similar principles as JPDAF in determining if the locations of the detected objects in two time instances are consistent, and therefore, if they represent the same physical entity or not.

4.1.4 Learning the appearance

The model allows learning from actual sensor data. The observed objects contain features that are matched to real objects, but also features that are not matched. The non-matched features may be caused by noise, or they may actually be features characteristic of the object. When enough similar features are related to an object, they can be regarded as characteristic. This process can be considered as learning. The learning occurs on two levels, from observed objects to real objects, and from real to meta-objects. The different types of learning are depicted in Figure 4.1.4.

4.1.4.1 Learning from observed objects

The observed objects contain features from actual sensor data. The features that are determined as characteristic are copied from the observed object to the connected real object. In this way the model learns about new aspects of a real object by gathering new sensor data. The copied features are marked as less reliable than the previously existing features. When a feature is matched to a real object, its reliability is reinforced. Over time the reliability of the features will depend on how often they are matched to the same object as the other related features.

This type of learning is evaluated in the experiment described in Section 5.3.1.

4.1.4.2 Learning from real objects

Real objects represent samples of an object class. If there are several real objects that have the same feature, it can be considered as characteristic of the whole object class. The feature can be added to the corresponding meta-object, that is, an object that describes the object class. Automatic learning from just one real object is not possible, because one object may not represent the whole class, and therefore at least two different real objects are needed to enable the similar features to be copied to a meta-object.

This type of learning is not experimentally evaluated in this thesis. However, its applicability is discussed in Section 6.2.1.2. The practical implementation of this type of learning is part of the future work of the research.

4.1.5 Object location and orientation

The pose of the object, that is, its location and orientation, determines its spatial state.

4.1.5.1 Spatial location

Spatial location means the exact x-, y-, and z-coordinates in the three-dimensional space. This is the actual location of the object. The coordinates are based on a predefined origin that represents the location $(0, 0, 0)$. There can be several origins defined, and one needs to determine which origin the coordinates are based on. Additionally, the origin of the object itself needs to be defined.

In many cases, the exact location of the object is not known. The uncertainty of the location can be expressed with a covariance matrix. This approach assumes that the uncertainty is an (x, y, z)-centered Gaussian distribution. This is a typical way to model location uncertainty robotic applications. [Thrun et al., 2005] The representation based on Gaussian distribution is compatible with Kalman filter and other similar algorithms.

The covariance of the location uncertainty is generally determined on the basis of the model of the measurement process itself. In addition, dynamic objects can move, which causes more uncertainty about the location as time has passed since the last measurement. The covariance of the location right after the measurement can be determined with the formula

$$\mathbf{P} = \mathbf{R}\mathbf{P_0}\mathbf{R^{-1}} \tag{4.1.5}$$

where $\mathbf{P_0}$ is the unrotated covariance matrix

$$\mathbf{P}_0 = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{4.1.6}$$

σ_x, σ_y, and σ_z being the standard deviations along the axes of the local coordinate system, and $\mathbf{R}$ is the rotation matrix

$$\mathbf{R} = \begin{bmatrix} c_\alpha c_\gamma - c_\beta s_\alpha s_\gamma & c_\gamma s_\alpha + c_\alpha c_\beta s_\gamma & s_\beta s_\gamma \\ -c_\beta c_\gamma s_\alpha - c_\alpha s_\gamma & c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & c_\gamma s_\beta \\ s_\alpha s_\beta & -c_\alpha s_\beta & c_\beta \end{bmatrix}. \tag{4.1.7}$$

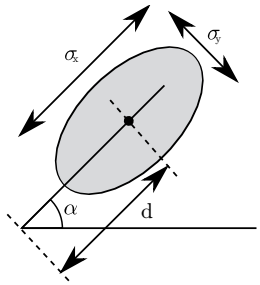


Figure 4.1.5: Uncertainty of the measured location of an object. α is the angle to an object related to reference coordinate system, d is the distance, and σ_x and σ_y are the standard deviations along the axes.

Here the sin and cos functions are written in a shorter form; $\sin(x) = s_x$ and $\cos(x) = c_x$. The angles α, β, and γ are the angles from the sensor to the object in Euler coordinates. The covariance matrix is often assumed to be constant, but it can also depend on, for example, the movements of the sensor itself, the environmental conditions, or other variables. Figure 4.1.5 illustrates the uncertainty area. In reality, because normal distribution is used, there is no sharp edge for the distribution. A threshold, such as σ or sometimes 2σ, can be used to determine where the distribution "ends".

The calculation of the uncertainty caused by the time since the last measurement is not as straightforward. The simplest model would be to assume an equal probability of movement in any direction. In this case the covariance $\mathbf{P}$ of the location at time t could be calculated with

$$\mathbf{P}(t) = \mathbf{P}_0 \, (t - t_0) \, p_m v, \tag{4.1.8}$$

where $\mathbf{P}_0$ is the covariance of the measurement, t_0 is the time instant when the measurement was made, p_m is the probability of the object moving, and v is the average movement velocity of the object. The term p_m is used because in some cases the object is more likely to move than in other cases. A person taking part in the task execution is more likely to be in a specific position than a person who is just passing by.

The assumption that the object moves in any direction is very simplifying. For example, a non-holonomic vehicle, such as a car, would more probably move forward than sideways. The movements also depend on the dynamics of the objects. This kind of calculation of the covariances requires a model and measurements of the kinematics and dynamics of the object. In addition, the movements of the object can be predicted on the basis of some prior knowledge. For example, a car is more probably going to stay on the road than drive into the gutter. If additional information is not available, Equation 4.1.8 can be used. Basically, this represents a worst-case scenario of location uncertainty. More thorough models are not discussed in this research.

4.1.5.2 Orientation

In many cases, the orientation of the object also needs to be known. Basically, orientation means rotation from global coordinates to the object's own coordinate system. There are several parametrizations to express the orientation of an object. Euler angles are widely used because of their easy interpretation. They define three angles, α, β, and γ, that are rotation angles around

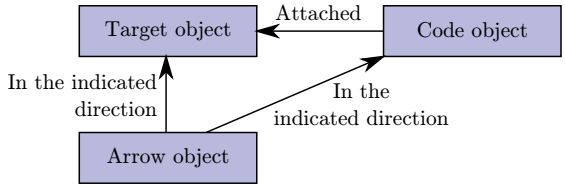


Figure 4.1.6: Example of topological connections to determine the location of a target object.

the x-, y-, and z-axes, respectively. The rotation of the coordinate system can be calculated with

$$\mathbf{C}' = \mathbf{R}\mathbf{C}, \tag{4.1.9}$$

where $\mathbf{C}$ is the original coordinate system defined by a 3×3 matrix, $\mathbf{C}'$ is the rotated coordinate system, and $\mathbf{R}$ is the rotation matrix calculated with Equation 4.1.7.

In world coordinates, the x- and y-axes are elongated along the ground plane, and the z-axis points up. With objects, the x-axis is defined as the *principal axis*. This means that if the object points in some specific direction, the x-axis defines this direction. Furthermore, vehicles and other moving objects have their x-axis pointing forward, or in the direction in which their movement usually occurs.

4.1.5.3 Coordinate origins

The spatial location and orientation are based on a coordinate system origin. It is not feasible to measure the locations of all the objects in the same global coordinate frame, but local coordinate frames are needed. One coordinate frame could be fixed to one point in the working area, another coordinate frame can be fixed to the base station of the robot, and yet another coordinate frame could be located at some global origin like the GPS point $(0.0, 0.0)$.

Using navigation technology such as GPS it is usually easy to determine the inter-origin translation and rotation. In this case, the 4x4 transformation matrix between the coordinate origins can be defined. The origins are organized as a hierarchical tree, where the global origin is the base of all the other coordinate systems.

Though possible, the absolute translation between the coordinate origins is usually not required. The tasks are generally executed in a bounded area, and only one local coordinate origin is needed. A coordinate system can be fixed to a local landmark that is used as a point of reference. The local coordinate system can be determined with the robot's own localization techniques.

4.1.5.4 Topological position

It is not always necessary or even possible to know the exact position of an object. It is often enough to know if an object is next to, on top of, or attached to the side of another object, or if it is a sign pointing at another object or indicated by a sign. Figure 4.1.6 shows an example of determining the position of the target object using the topological connections between objects. These cases may occur especially when the information is not gathered by the robot using its sensors, but the information is inputted by a human. The topological connections are not necessarily fixed, but may depend, for example, on the direction of another object. If an arrow-shaped sign is pointing to another object, the orientation of the arrow effectively defines the target object in question.

Figure 4.1.7: Sign pointing towards the direction of the smaller ball. [Heikkilä et al., 2006]

Figure 4.1.7 presents an example of such a sign. The smaller ball is interpreted as the tip of the arrow. By measuring the locations of the two balls, the pointing direction can be determined.

Topological connections between two objects can be defined with two properties: distance and direction. Both of them can contain lower and higher limits. For example, the distance to an object could be between 2 and 5 meters from the reference object, and its direction between 30 and 60 degrees. The direction can be defined either in a global coordinate system, or in the reference object's own coordinate system. In the latter case, the orientation of the reference object affects the pointing direction. If an object is attached to another object, its distance can be defined as 0.

The topological position is different from the spatial position in the sense that it does not necessarily uniquely define the location of an object. Furthermore, when actual manipulative operations are being conducted on an object, its spatial location needs to be known. The topological location can mainly be used to determine target objects for the robot. When doing the actual manipulation, the robot needs to locate the object in terms of its spatial coordinates. This does not have to override the topological location information, but both modes of location information may coexist.

4.2 Sharing cognition

Humans and robots can share their cognitions by exchanging information on different levels. On the lowest level, the user can label parts of the sensor data to represent a certain object. This can also be done by affecting the scene viewed by the robot with a pointer stick, signs, or illumination. A human can also describe the object to a robot either by giving a verbal or numeric description of the object, or by inputting external sensor data, such as a photograph of an object that represents a certain object. The human can also get information from the model on the basis of the existing real and observed objects. This information can be used for delivering information from a remote scene, as well as for verifying that the robot has interpreted the scene correctly.

4.2.1 Referring to the robot's sensor data

One way of sharing the cognition is to use the human's cognition to interpret the robot's sensor data. The interpretation can be done by accessing the acquired sensor data, or delivering the information through the scene that the robot is viewing.

4.2.1.1 Explicitly selecting sensor data

The user can select a segment of the sensor data to define it as an observed object and associate it with a real object. The user therefore executes the segmentation and recognition for the robot. In the segmentation phase, an observed object is created. Features extracted from the sensor data are then associated with this object. In the recognition phase the user either creates a new real object with which the observed object is associated, or he or she associates the observed object with a real object that already exists. This procedure is depicted in Figure 3.5.2 in Chapter 3.

4.2.1.2 Pointing objects in the real world

A human can point to objects with a physical object or otherwise make a target entity distinctive to the robot. This is related to the previous case, but now the robot needs to perform the sensor data interpretation itself. The user just assists the robot and makes the interpretation easier. So instead of having to understand the whole scene, the robot just needs to recognize a predefined object or detect an illuminated area to recognize the target object. Different means for pointing are using a distinctive-looking stick, using predefined signs, like arrows, or illuminating the target entity with a laser pointer or a flashlight.

From the following pointing methods, the pointer stick and illumination are used in the experiment described in Section 5.3.2. The use of an arrow-type sign is experimentally evaluated by Heikkilä [2009].

Pointer stick A distinctive-looking pointer stick is defined as a real object in the robot's cognition. When recognizing the stick, the robot follows the direction it is pointing in and thus finds the target object. In practice, the stick is defined to have a topological connection towards the direction it is pointed in. The connection is not initially linked to anything, but when it is recognized, the robot determines the target of the topological link.

Signs An arrow-type sign can be used to identify objects in a similar way to a stick. Another way of using signs to mark objects is to surround an object with them. When they are seen, the robot knows that they surround the target, and it is able to perform the segmentation of the sensor data on the basis of this. A sign can be deliberately made very distinctive, and thus it is easy to see it with the robot's sensors. However, determining the orientation of a sign may cause errors if it is done with only one camera. Because the orientation of the sign is often important information, a pair of stereo cameras is usually necessary to interpret signs.

Pointing with illumination Another way of leading the robot's attention to an object is to use a light source, such as a laser pointer or flaslight. The robot can see what is being pointed to, but the pointer itself is not a rigid object. The pointer just modifies the perceived scene. In practice, a virtual real object is created. The only definition in the object is that it is recognized with hue segmentation, possibly combined with background extraction. The real object is therefore not a physical entity, but it just represents the light. When the illuminated target is found, it is defined as being the object in question.

4.2.2 Describing an object

The user can describe an object to the robot. The description may include complete information, including the location, orientation, and appearance of the object, or it may contain only some information, for example its color. Any of this information can be utilized when building the cognition model. The reliability of the features affects the certainty of the recognition, so the more information is available, the better the recognition is.

Describing an object is done by creating a new real object or processing an already-existing one. This can be done offline with a separate computer, or it can be done directly with the actual online cognition model. When working offline, the changes and additions are made to the actual model after the editing is finished. In the online case the information can be directly associated with a real object that can even be used in a task. After the real objects are modified, the feature database and its index need to be updated for feature matching.

The following subsections describe a few case examples of how the user can input data to the model and how this is utilized in task execution. The modalities used in the experiments in Chapter 5 are numeric descriptions described in Subsection 4.2.2.3, unifying codes described in Subsection 4.2.2.4, and object's location as described in Subsection 4.2.2.5.

4.2.2.1 Photographs of an object

Taking one or more photographs can be used to train the robot in the appearance of an object. Features are extracted from images, and they are associated with the real object. With one photograph, the features represent only one side of the object, and thus the object can be recognized from that side. If the images are taken from several directions, the recognition becomes more reliable. If only a few photos are used, many different types of features need to be extracted to improve reliability. In practice, SURF features should be extracted for accurate appearance recognition, and a color histogram description for coarser long-distance recognition. The robot has to be equipped with a camera to recognize the object trained with the camera image, because the images do not contain any 3D information that would be needed for recognition with scanning sensors, such as laser scanners.

It may be impossible to determine which part of the photograph belongs to the object and which one to the background. When a human takes a photograph of an object, it is a reasonable assumption that the object is usually in the middle of the image. Therefore the certainty of the features at the center of the image is higher than the certainty of the features located at the edges.

4.2.2.2 Plain or rich 3D data

A pair of stereo images can be used to produce a set of 3D points that describe the object. In addition, the feature information can be assigned to the points to create rich 3D data. With data of this kind fewer samples are needed for training than with just 2D image features, but the training is more complicated.

4.2.2.3 Numeric description

Human-understandable measures such as size or color can be used as weak descriptions of an object. All these kinds of features are usually ambiguous, and they are not sufficient for reliable recognition when used without any additional cues. However, when combined with information on

(a) A text string written with machine-readable OCR-A font.

(b) A code-128B standard barcode with number 42 encoded.

(c) A semacode area code with text ABCDE encoded.

(d) Concentric circles with variable widths that unify the symbol.

Figure 4.2.1: Examples of machine readable codes that encode information.

the location of the object, these features increase reliability. A numeric description of the size of the object can also be used together with photographs to allow recognition with a scanning sensor too. The numeric information inputted by a human may not be exact, and therefore the numbers cannot always be used as strict criteria for recognition, but ranges of numbers need to be used.

If a numeric description is used as the only feature of an object, the robot can pick all the suitable objects as candidate targets and check with the user which one of them is the right one. This technique was described in Section 4.3.2.5.

Color Humans do not understand colors in terms of numbers but with names. Each name covers a range of different hue-saturation combinations. The ranges overlap, because a human's color peception is heavily subjective, and it may be difficult to determine if an object is pink, magenta, purple, rose, or just light red. A robot and a human can understand the colors in the same way if a color chart is used. Every color in the chart is defined in terms of a numeric color definition. The human can compare the colors in the chart to those in the real world. This enables a shared understanding of the colors to be attained.

Color-based recognition is not always reliable because of its low information content. Color is usually not a very unifying feature in man-made environments, or even in the wild world of nature. Therefore the color information always needs another description if an object is to be uniquely recognized.

Size The size of an object is a very ambiguous concept. There are various metrics for measuring the size of an object. Simple cube-like objects such as furniture are usually measured in three orthogonal dimensions named "width", "height", and "depth". With round conical objects, such as trees, the only available measurement may be the perimeter or diameter of the base. When measured with a scanning sensor, the intersection may affect the measured size, and this can also cause false matches. Like color, size description is not a very unifying feature, and in practice it needs to be combined with other features.

4.2.2.4 Unifying code

A special feature type is a code, such as a barcode, dot code, color sequence code, machine-readable text, distinctive symbol, RFID tag, or a similar code that can be uniquely read by a machine by using a camera or other sensors. These codes are often understandable also by humans, or at least a descriptive text can be attached next to the codes. Figure 4.2.1 shows examples of machine readable codes and text. In addition to this type of codes, there are various optically recognizable

tags used mainly in augmented reality applications, such as ALVAR augmented reality software. VTT [2011] Because this kind of tags are not designed to carry arbitrary encoded data, they are not considered in this research.

The OCR-A font shown in Figure 4.2.1a was originally designed for easy optical character recognition. Modern optical character recognition algorithms, however, are able to read text written with various typefaces. Because the variety of different characters and symbols is wide, the optical character recognition is not always reliable. In addition, character recognition requires relatively heavy computing.

A barcode depicted in Figure 4.2.1b is originally designed for robust machine reading. Barcodes are used for countless applications, for example to label consumer products. However, the barcode is designed to be read perpendicularly. The information is encoded to the widths of the bars, and perspective distortion could negatively affect the reliability of reading the code.

An extension to traditional barcodes is a 2D code. Figure 4.2.1c shows an example of a Semacode-type code. This kind of codes are often used to be read with mobile phone cameras, and they are usually used to encode addresses of WWW pages or similar information. As with barcode, the 2D codes are also designed to be read perpendicularly, and viewing from different directions may reduce the readability of the code.

A different kind of approach is a circle code depicted in Figure 4.2.1d. The data is encoded to the variations of the diameters of the circles. This kind of coding scheme does not allow encoding much data into the code. However, because the code only consists of concentric circles, its readability is good even when not viewing perpendicularly. Locating an ellipse from an image is a trivial task. It is equally easy to locate all the circles, and because they are concentric, the validity can be confirmed. RCA company developed a standard called bull's-eye for circle codes. This standard, however, is not currently used, and there exist no software for decoding this type of codes. Therefore, an algorithm for this purpose is developed in this research.

Detecting a circle code Circle codes are a useful form of identification because reading them is relatively reliable. Determining if a region is circular can be determined by examining the Hu moment invariant

$$I_1 = \frac{\mu_{2,0}(S) + \mu_{0,2}(S)}{\mu_{0,0}(S)^2}, \tag{4.2.1}$$

where I_1 denotes Hu's first moment invariant, S is the shape being analyzed, and $\mu_{i,j}$ are the shape moments. [Hu, 1962] The circularity of the shape S can be determined by comparing the moment to $1/2\pi$. [Žunić et al., 2010] The concentric nature of the circles is a strong cue for recognition. In addition, data can be encoded into the variable line widths. Algorithm 4.3 describes the location of a circle code from an image and reading its code. The algorithm returns a numeric description for each circle code. In addition to the midpoint coordinates, a vector describing the actual code is returned. The vector includes the relative diameters of the circles. This can be used to unify different codes.

A great advantage of using codes is that they can be made very distinctive and unifying. The problem is that it is not always possible to attach a code to an object. The code may also be distorted by the deformation of the object, or parts of the code may be occluded. Codes are rarely designed in such a way as to tolerate partial occlusion, and in such a case an unreliable match may occur.

Codes are special cases. Because a code is usually not part of an object, but it is just attached

Algorithm 4.3 Reading and locating circle code

The algorithm finds a circle code from a grayscale image $I(x, y)$. The circularity threshold $T \in [0, 1]$ needs to be defined. Higher T means a stricter requirement for circularity.

1. Execute Canny edge detector to extract the edges from the image.
2. For each 8-connected pixel region
   (a) Calculate Hu's moment invariant I_1 using Equation 4.2.1.
   (b) If $\frac{1}{2\pi I_1} > T$
      i. Calculate midpoint and radius for the region
      ii. Store the parameters to an array A
3. For each parameter group in array A
   (a) Compare the midpoint to every other midpoint in array A.
   (b) For every circle with an equal midpoint
      i. Group the circles to array B
      ii. Remove from array A
   (c) Normalize the radii of the circles in array B from 0 to 1.
   (d) Sort B.
   (e) Return a vector containing the midpoint and vector B. This is the description of the circle code.

to it, the code is treated as a separate object, and a topological connection between the code and the actual object is defined. Therefore, a real object describing an object does not contain the code information, but just a topological "attached-to" connection to the code. When the code is found, the actual object is also found by following the topological link.

4.2.2.5 Object's location

An object can also be defined by determining its spatial or topological location. Neither of these definitions is unifying, and these also need to be used in conjunction with other feature types.

Spatial location A spatial location describes the location of the object on a map. If the object is static, this information leads to a correct location, but if manipulation is needed, a problem may occur if the object is not recognized. In that case it is impossible to determine how to operate the object. Usually, the location is used just as additional information, not the only description.

Topological location If the spatial location of the object cannot be determined accurately, a topological location definition may be used. This may also be advantageous when the object is not static. The object used as a reference for the topological location can therefore also be moved. For example, if the reference object is an arrow sign, it can be moved to point to the target object if it moves. The challenge in the definition of a topological location is that it is always tied to a reference object. If the location of the reference object is not known, it needs to be determined first.

4.2.2.6 Functional features of the object

The object description can include non-perceivable features of the object in addition to the visible features. These features describe the functional properties of the object. They are not used for recognizing the object by appearance, but the information can be utilized on a higher level deduction. This model does not limit what kinds of functional features can be used. The features are described as pairs of the name of the feature along the value or textual description.

The simplest example of a functional feature is a numerically measurable non-perceivable feature, such as a weight of an object. The weight may have an effect to how the robot uses its actuators or if it is able to lift an object at all. An example of a more loosely defined functional feature is contents of a box. Also this does not have an effect on the appearance, but it affects how the robot is supposed to handle the object. A box with fragile contents, such as glass, should be carried with more care than a box with for example books. Yet another type of functional features is the way an object is used. For example, a table can support objects, therefore they can be laid on top of it.

These types of functional features are typically entered by human and they are utilized in a task execution. These features don't affect the other functionality of the model, they just are associated to the corresponding objects for later use.

4.2.3 Getting the information from the model

The user can also get data from the cognition model. This can be needed in three different situations. The user can check if the robot has recognized the objects correctly, and if it has actually detected all the objects. Second, the robot may have surveyed an area, and information is delivered to the human. In this case the robot is used as the "remote eyes" of a human. In a third scenario, the robot may have detected various potentially interesting objects, and shows them to a user, who can then decide which ones are significant and which ones are false alarms. All these actions are done through a graphical user interface.

The first case, checking the correctness of recognition, can be used in situations where wrong actions are not acceptable; otherwise the robot can sometimes be allowed to execute the wrong action, because usually it is able to detect later that it is doing the wrong thing. When the robot sees an object from a closer distance, correct recognition becomes more probable. Checking the recognition result may also be needed in cases when a new object is being trained and its performance is being evaluated.

The second case, getting a survey result, is used in cases when the user cannot or does not want to enter some area. The reasons can be, for example, a long distance or dangerous conditions. After a surveying task the cognition model contains a real object for each recognized target. In addition, there may be compressed image snapshots of each object in the model. In this way the user can also evaluate the correctness of the recognition.

A practical example of the third case is presented in Section 4.3.2.5. Snapshots of one or more candidate objects are presented to a human operator, who needs to choose the interesting objects from among the observed objects. Choosing the object is done either by associating a real object with one of the observed objects or by choosing an observed object as the target.

4.3 Task execution

The robot's decisions during the task execution are based on the present cognition model. The movements and actions are planned in a way that depends on whether the objects exist in the cognition model and what their locations are. Target objects are approached and manipulated, and obstacles are avoided. The cognition model gives support to all such operations.

Sometimes reactive perception is also needed. This is not handled in the cognition model, but should be executed as a parallel process. An example of reactive perception is emergency braking, when an object is detected with a proximity sensor, such as a laser scanner. In this case, there may be no further information needed about the object than the mere fact of its existence. As an analogy to humans and many other animals, this kind of perception can be considered as a "reflex" of the robot, and therefore does not utilize the cognition model.

4.3.1 Objects in cognition model

The high-level task description defines the tasks and their targets. In practice, the targets are objects. A real object is generated for each target object of the task. These real objects are passed to the cognition model, which uses them to build the actual cognition model. This is done with the procedure described in Section 4.1. Building the cognition model is done automatically on the basis of the object data from a higher level. Basically, the task does not need to take care of the perception, but it can utilize the cognitive information as it is generated from the sensor data.

The cognition model also contains information about obstacles. Obstacles are anonymous objects that are not necessarily associated with any meta-object, but the only information in the cognition model is the sensor data related to the objects. The model knows the locations, sizes, and shapes of the obstacles. The robot can use this for planning its movements.

4.3.2 Related tasks

The following discusses a few case examples and how they are handled by the model. The examples are typical tasks executed by a mobile service robot.

4.3.2.1 Path planning

The most elementary task that a mobile service robot can execute is moving from one place to another. In a completely empty working area this is a very elementary task that requires only local positioning based on, for example, odometry, and knowing the kinematics of the robot in order to plan the movements. However, in an area with obstacles, more advanced path planning is required. Planning is based on information on obstacles present in the area. There are several algorithms for path planning, for example, A* [Hart et al., 1968] and D* [Stentz and Mellon, 1993]. The algorithms generally try to find a balance between finding the shortest, fastest, or most energy-saving path while keeping the path collision-free.

Cognition model can be used as a tool for path planning. In practice, the cognition model is built using robot's sensors. This requires 3D mapping or 2D distance scanning capability to produce a map of the locations and sizes of the objects. To plan a collision-free path it is not required to know the identities of the objects but they can all be considered just as obstacles. This can be thought of as hiding the objects under "hoods" or replacing them with boxes of equal size. The locations of the objects are then inputted to the data structure of the path planner.

The path planners are usually based on an occupancy grid that represents the space as an array. The elements of the array represent certains point in the space, and the values of the elements indicate if the corresponding points are traversable or not. The objects in the cognition model are now placed to the occupancy grid. The elements that are on or inside the objects' boundaries are marked as untraversable, while everything else is considered free space. Finally, if the path planning algorithm does not take into account the robot's size, the obstacles are extended by the robot's size, and the robot can be considered as a zero-dimensional point.

One challenge is caused by moving objects that may require replanning of the path during the course. This means that the cognition model needs to be updated continuously in order to detect moving obstacles. Parts of the cognition model may not be up to date continuously. In addition, inaccuracies of the sensors may cause uncertainty in the locations of objects. This inaccuracy is included in the model with the covariances of the locations. The simplest approach is a pessimistic approach; that is, everything that is not known to be free space is assumed to be untraversable. Another approach is to plan the robot's actions to gather more data from the areas with incomplete sensor data. This needs to be executed as a part of higher-level task planning, because, depending on the task, it may be unacceptable to perform maneuvers that are not part of the actual task.

4.3.2.2 Planning the robot's actions to gather the required information

If the robot needs to get information from an area that is not yet covered by sensors, it may need to move itself to another position to get a better view of the area. In general, this is called perception planning. The movements need to be planned on a higher level, because they may affect the actual task execution. The planning is done on the basis of the present cognition model.

Each observed object in the cognition model contains a timestamp that indicates when the object was last measured. This also applies to anonymous objects, obstacles, and walls, basically, everything that has been measured. In this way the perception planning module can determine which parts of the working area have not been updated lately, and can plan the movements accordingly. In practice, the goal of the task is then to move to a position where it sees the missing areas. The sensor model is used to determine how the area is covered by sensors when moving to a certain position.

4.3.2.3 Working with a completely described object

If the appearance and location of the object are known, the manipulation of the object is done in three phases: coarse approach, fine approach, and manipulation.

A coarse approach means that the robot moves towards the known location of an object. In this phase the movement can rely on the robot's navigation capabilities. The path is planned as described above, avoiding possible obstacles on the course.

When the robot is close to the object, for example, within about three meters, it needs to start the accurate approach phase. In this phase, it is no longer feasible to assume that the location of the object is accurate enough if it is not measured relative to the robot. The task needs to make sure that the robot does not carry on with the task execution before the accurate location is measured and updated to the cognition model. Because the object is measured using the robot's sensors, the relative location of the object is accurate, even though there may be error in the self-localization of the robot. The more accurate approach can thus be carried out on the basis of the location information from the cognition model.

In the manipulation phase, the robot needs to know the position and possibly the orientation of the target object. The model is used to measure these parameters. They are then used for planning the manipulation actions of the robot.

4.3.2.4 Finding an object on the basis of appearance and approximate location

When the appearance of an object is known but its location is not explicitly defined, the robot may need to search for the object. In this case, the user transfers his or her cognitive understanding of the object to the robot's cognition. In practice, the target object is defined as a real object that has some features describing its appearance. In the first phase, a perception planning subtask is required. This part of the task plans the robot's movements in such a way that every part of the working area is seen by the sensors. The cognition model is updated continuously. The searching area is also included as a location definition to the real object. The location may be an approximate spatial location, or a topological location defined relative to another object. After the target area has been covered, if the robot has recognized the object, one or more observed objects have been associated with the real object, and thus the location of the object should be fixed. From here on, the next phases are the same as when working with a completely described object.

4.3.2.5 Finding candidate objects for later manipulation

As a further extension of previous cases, sometimes the number of target objects is unknown. There may not be any objects or there may be many. Sometimes the definition of an object may be so weak that false positive matches may occur. For example, when searching for garbage on the ground, a piece of garbage may be defined as a bump on the ground. Because there may be other bumps too, for example rocks, clumps of grass, etc., there will probably be lots of false positives in addition to the true positives. The robot stores compressed images of each observed object it encounters. After the search has finished, the user can view the images related to the observed objects, and determines the types of candidates which are correct matches and those which are just false positive matches. The location of each object is stored in the corresponding real object. With this information the robot can later process the objects again.

4.3.2.6 Determining the robot's own state with perception

Many robots do not get perfect measurements of their own state. The manipulator angles may not be known accurately, and the robot's own position in relation to the target may not be as accurate as is needed for manipulation. If the sensors are able to see some of the robot's parts, their states can be measured with similar methods to those used for the states of other objects. This is called proprioceptive sensing, or proprioception. The robot can be defined as one object. The most accurate method is to build a CAD model of the robot or its manipulators, and use model fitting mechanisms for measuring the position and orientation of the moving parts. A less accurate method, but one which is simpler to use, is just to search for the moving parts on the basis of their appearance. In this case, their appearance is defined as a set of image or 3D features that are used for finding the moving parts. The information on the poses of the robot's moving parts is utilized in the higher-level task. This can be used as feedback to the motion control, for example, when visually servoing the robot's manipulator.

Chapter 5

Experimental validation

The proposed model for robot cognition is validated through tests that evaluate the model from several different perspectives. The experiments are divided to two sections. Section 5.2 goes into building the cognition model in collaboration with the human. The experiments presented in this section do not include any task execution, but concentrate entirely on building the model. Section 5.3 presents two cases in which the robot performs a simulated task on the basis of the information included in the cognition model.

The concept is not implemented as a complete system, but only separate concepts of the system are tested. The algorithms that are tested are mainly implemented with Matlab, and do not work in real time. The robot's movements in the experiments with actual tasks are manually controlled by the author to ensure repeatability and full control on the test setup.

5.1 Test platforms

Two different kinds of test platforms, the WorkPartner service robot and Avant-based semi-autonomous machine, are used in the tests. They differ in their mechanical structure, their sensors, and their application area.

5.1.1 WorkPartner

WorkPartner is a centaur-like service robot at the Department of Automation and Systems Engineering of Aalto University. [Halme et al., 2003] The subsystems of WorkPartner have been extensively reported by Ylönen [2006]. An image of WorkPartner is shown in Figure 5.1.1. The robot was mainly built between 1998 and 2005, but the further development of its systems, especially related to task execution [Heikkilä, 2009] and mobility [Leppänen, 2007], has continued since then. The robot has a hybrid locomotion system with both legs and wheels in order to equip it with a capability to cope with various kinds of terrain. The manipulator is a human-like torso consisting of two hands, an upper body, and a head. The robot is actuated by electrical motors.

WorkPartner's environmental sensing is based on five different types of sensors: one camera, a laser pointer, a laser scanner, three ultrasonic range meters, and a pair of stereo microphones for hearing. Most of the sensors are attached to the robot's manipulator and head, which enables them to be pointed in different directions.

The camera is a color camera whose image is grabbed to an Axis image server that delivers

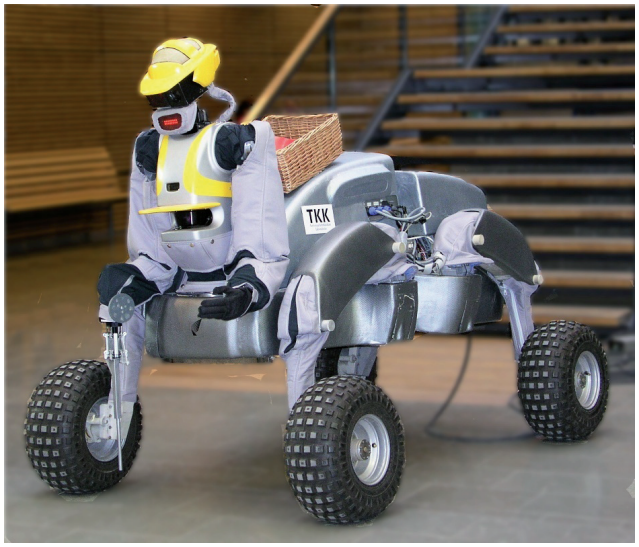


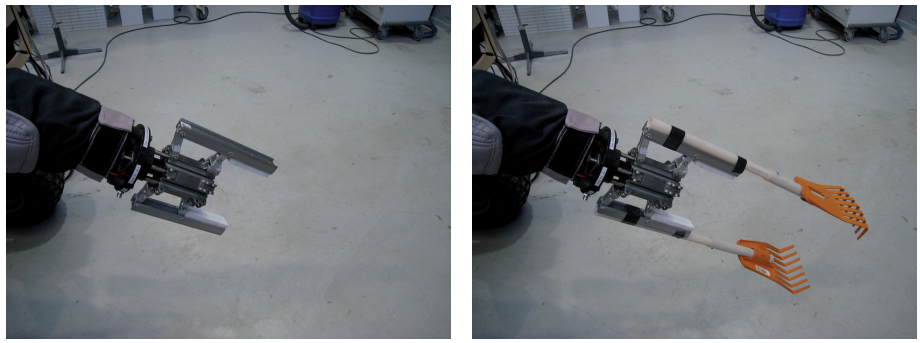
Figure 5.1.1: WorkPartner service robot

Figure 5.1.2: Close-up photograph of WorkPartner's head equipped with laser pointer and color camera.

images to programs. The camera is attached to the robot's head, which can be turned with a pan-tilt unit. It can therefore be used to focus and track objects in different directions. An image of WorkPartner's head in shown in Figure 5.1.2.

The laser pointer is mounted next to the camera. It can measure distances to a point with a theoretical accuracy of 1 mm. The laser pointer is in the robot's head and can be used to assist the camera in determining the distances to objects. The camera and laser pointer are in fixed positions in relation to each other. The location of the laser pointer in the camera image can be calculated on the basis of known geometry.

The laser scanner is a SICK LMS-291 type indoor device. It is mounted on the middle of the torso of the robot's manipulator. The manipulator can be tilted, and thus the direction of the scanning plane of the laser scanner can be varied. With this method the robot can build a 3D



(a) Simple finger-type gripper. (b) Gripper equipped with extensions.

Figure 5.1.3: Gripper configurations alternative to the gripper shown in Figure 5.1.1.

Link	Variable	θ	α	l	d
1			90		
2	body rotation	θ_1	90		54
3	body inclination	$\theta_2 + 90°$		563	
4		$-90°$	-90	-42	11
5	head pan	θ_{pan}	90		
6	head tilt	θ_{tilt}			

Table 5.1.1: Denavit-Hartenberg parameters between WorkPartner's coordinate origin and laser point.

model of its environment by using the scanner in different orientations.

The stereo microphones are mounted on the robot's shoulders. They are used to determine the direction of loud command words.

The ultrasonic sensors are located at the rear of the robot, facing backwards. The sensors are similar to those used in cars to assist parking and backing.

In the test case, the camera and laser pointer are used for recognizing objects and measuring their positions. Because the objects are located in different directions the pan-tilt unit needs to be used to direct the robot's perception.

The gripper of the robot is customizable to various applications. Figure 5.1.1 shows WorkPartner equipped with a spike suitable for picking up some types of pieces of litter from the ground. Other alternative is a gripper with two finger-type pieces that can be opened or closed. Yet another option is to install extensions to the fingers providing a better grip to various objects. The two latter types of gripper are shown in Figure 5.1.3.

5.1.1.1 Sensor model of laser pointer

The laser pointer measures the distance to one point. Its location can be determined by the measured distance and orientations of the pan-tilt unit (PTU) and the robot's manipulator. The kinematic transform from the robot's coordinate origin to its head is expressed with Denavit-Hartenberg parameters which take into account the angles of the joints. [Forsman, 2005][Craig, 2004] Table 5.1.1 lists the parameters. Figure 5.1.4 shows the dimensions of the robot and the locations of the coordinate origins. One row of parameters is used to build a transformation

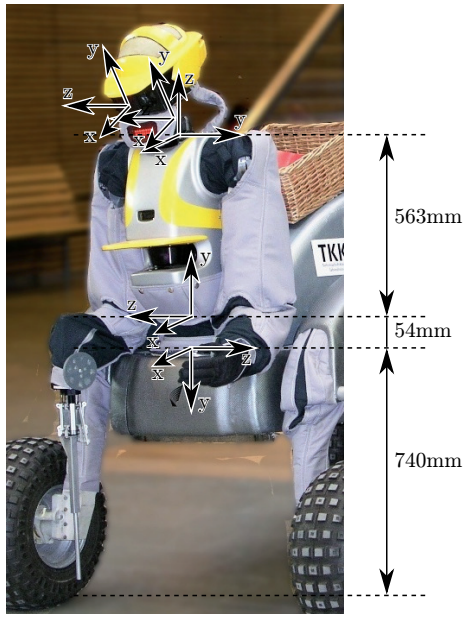


Figure 5.1.4: Coordinate origins listed in Table 5.1.1 and dimensions of WorkPartner's manipulator. The WorkPartner's coordinate origin is the one closest to the ground.

matrix

$$ {}^{n-1}\mathbf{T}_n = \begin{pmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.1.1} $$

that does the rotation and translation from the coordinate system $n-1$ to the coordinate system n. The matrices are multiplied in a chain to produce the whole transform. In addition to the Denavit-Hartenberg parameters, the transform from the head to the end of the laser point is expressed with a simple translation

$$ {}^{6}\mathbf{T}_7 = \begin{bmatrix} 1 & 0 & 0 & 0.041 + d \\ 0 & 1 & 0 & 0.090 \\ 0 & 0 & 1 & 0.067 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.1.2} $$

The 3D coordinates of the point are obtained by calculating

$$ \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \left(\prod_{k=0}^{6} {}^{k}\mathbf{T}_{k+1} \right) \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.1.3} $$

The measurement performed with the laser pointer is very accurate. On the basis of practical experience, the standard deviation of the location uncertainty is about 0.01 m. The uncertainty is assumed to be the same in every direction. On the basis of this assumption, the covariance of the

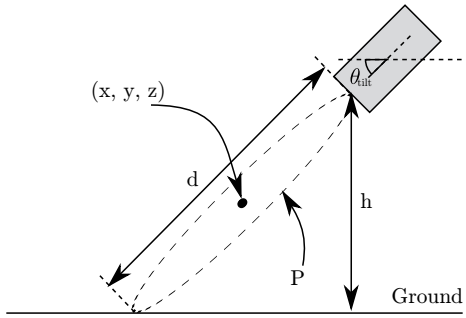


Figure 5.1.5: Geometry for estimating the locations of objects with monocular camera used in WorkPartner. θ_{tilt} is the *tilt*-angle, h is the height of the camera from the ground, d is the estimated distance, and x, y, and z are the estimated coordinates of the object. $\mathbf{P}$ is the 2σ area of the covariance of location uncertainty. The dimensions of the diagram are not drawn to scale.

measurement uncertainty for the laser pointer is simply

$$\mathbf{P} = \begin{bmatrix} 0.01^2 & 0 & 0 \\ 0 & 0.01^2 & 0 \\ 0 & 0 & 0.01^2 \end{bmatrix}. \tag{5.1.4}$$

The covariance matrix is a scaled identity matrix describing a ball-symmetric distribution. Therefore, rotation does not affect it. This is different from a case where the correlation terms are non-zero. The camera model explained below is an example of a case where the covariance matrix is not a scaled identity matrix, but an arbitrary diagonal matrix, and is therefore affected by rotation.

5.1.1.2 Sensor model of camera

WorkPartner is equipped with only one camera, and therefore it cannot determine the exact distance to the targets by using the camera only. It is assumed that the ground is sufficiently flat and located 0.74 m below the origin of the robot's coordinate system. The coordinates of a target object are set to halfway between the camera and the estimated ground, and the covariance of the location is set in such a way that the distribution covers the area between the camera and ground. This is illustrated in Figure 5.1.5.

The camera coordinates are calculated with the same transformation matrices as the laser pointer, with one exception; the last matrix is

$${}^{6}\mathbf{T}_{7} = \begin{bmatrix} 1 & 0 & 0 & 0.041 + \frac{d}{2} \\ 0 & 1 & 0 & 0.090 - y_c \\ 0 & 0 & 1 & -0.067 + x_c \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.1.5}$$

where

$$\begin{cases} d &= \frac{h}{\sin\theta_{tilt}} \\ x_c &= \frac{x_i - o_x}{f_x} d \\ y_c &= \frac{y_{im} o_y}{f_y} d. \end{cases} \tag{5.1.6}$$

Parameter	Explanation	Value
o_x	Image center, x	360.0
o_y	Image center, y	288.0
f_x	Focal length, x	785.4
f_y	Focal length, y	785.4

Table 5.1.2: Parameters of WorkPartner's camera

Variables h and θ_{tilt} are shown in Figure 5.1.5, variable h is the height of the camera from the ground, θ_{tilt} is the tilt angle of the camera, (x_c, y_c) are the image coordinates of the object in meters, (x_i, y_i) are object's image coordinates in pixels, and o_x, o_y, f_x, and f_y are camera parameters acquired with the calibration procedure. [Bouguet, 2008] Their values are listed in Table 5.1.2. Other parameters, such as lens distortion, are considered negligible, and are not included in the camera model. The estimated 3D coordinate of the object is calculated using Equation 5.1.3.

The covariance of the location is calculated with

$$\mathbf{P} = \mathbf{R} \begin{bmatrix} d & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \mathbf{R}^{-1}, \tag{5.1.7}$$

where $\mathbf{R}$ is the rotation matrix

$$\mathbf{R} = \begin{bmatrix} \cos\theta_x \cos\theta_y & -\cos\theta_x \sin\theta_y & \sin\theta_x \\ \sin\theta_y & \cos\theta_y & 0 \\ -\sin\theta_x \cos\theta_y & \sin\theta_x \sin\theta_y & \cos\theta_x \end{bmatrix}, \tag{5.1.8}$$

where

$$\begin{cases} \theta_x & = \tan^{-1} \frac{x_i - o_x}{f_x} - \theta_{pan} \\ \theta_y & = \tan^{-1} \frac{y_i - o_y}{f_y} - \theta_{tilt}. \end{cases} \tag{5.1.9}$$

This produces a covariance that characterizes a distribution whose $\sigma = \frac{d}{2}$, meaning that the $\pm 2\sigma$ range (95% of the distribution) covers the range between the camera and ground, as shown in Figure 5.1.5.

If the distance to an object is known, for example on the basis of a laser pointer measurement, it is possible to calculate the size of the object simply by subtracting the calculated the 3D locations of the boundary points of the object with

$$\begin{cases} width & = \max(\mathbf{x_c}) - \min(\mathbf{x_c}) \\ height & = \max(\mathbf{y_c}) - \min(\mathbf{y_c}) \end{cases}, \tag{5.1.10}$$

where $\mathbf{x_c}$ and $\mathbf{y_c}$ are arrays of calculated X and Y coordinates of the 3D points. It should be noted that measuring width and height in this way is not very accurate. It depend on the viewing angle and is prone to errors due to incorrectly measured boundaries and distance. Basically, the width and height measures can only be used as approximations.

In addition to transforming the camera coordinates to the 3D world, also the projection from 3D coordinates to camera image is often needed. As the transformation matrix between the world coordinates and camera coordinates is known, the projection from the world coordinates (x, y, z) to the metric image coordinates (x_c, y_c) and to image pixel coordinates (x_i, y_i) is calculated simply

Algorithm 5.1 Measuring the location of an object with laser pointer in conjunction with the camera.

This algorithm is used to determine where the laser pointer should point when using it with a camera to measure a location of a target object.

1. Calculate the 3D location of the laser point with Equations 5.1.1 and 5.1.2.
2. Calculate the projection of the laser point to the image with Equations 5.1.11, 5.1.12, and 5.1.13.
3. Calculate the horizontal and vertical angles θ_x and θ_y to the target object with Equation 5.1.9.
4. Turn the pan-tilt-unit according to the angles θ_x and θ_y.
5. Repeat from phase 1 if needed.

with

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = {}^{6}\mathbf{T}_{7} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5.1.11}$$

$$\begin{cases} x_c &= \frac{x'}{z'} \\ y_c &= \frac{y'}{z'} \end{cases} \tag{5.1.12}$$

$$\begin{cases} x_i &= f_x x_c + c_x \\ y_i &= f_y y_c + c_y. \end{cases} \tag{5.1.13}$$

5.1.1.3 Using the laser pointer in conjunction with the camera

Pointing the laser pointer to a target requires using the camera to detect the actual object and ensure that the laser pointer actually hits the target. In a bright sunlight it may not be possible to see the laser pointer light, therefore the location of the laser dot needs to be calculated by projecting the measured 3D location of the point to image coordinates. It is then possible to evaluate if the laser hits the target, or if the orientation of the laser pointer needs to be changed. Algorithm 5.1 presents the phases required for measuring the location using the laser pointer in conjunction with the camera.

5.1.2 Avant

Another machine used in the tests is an Avant-based semi-autonomous vehicle. [Saarinen et al., 2007] Avant is a product of Avant Tecno Oy. It is modified and instrumented by the Center of Excellence in Generic Intelligent Machines. The centre is a joint research group between the Department of Automation and Systems Engineering of Aalto University and the Institute of Hydraulics and Automation of Tampere University of Technology. Avant is a small machine for various tasks related to agriculture and similar tasks. The modified machine is actuated by computer-controlled digital hydraulics. An image of the Avant machine is shown in Figure 5.1.6.

Avant's environmental sensing is based on a pair of stereo cameras and a laser scanner. The stereo cameras are used in the tests. The test case consists of building a full 3D model of the

Figure 5.1.6: Avant machine based intelligent machine.

environment by using the stereo cameras. [Terho, 2010]

5.1.2.1 Sensor model of stereo cameras

The stereo cameras are used for creating a full 3D model of the scene viewed by the cameras. The wide field of view of the camera optics causes heavy distortion of the images. Therefore the camera images are corrected from geometric distortions. In addition, a perspective rectification is applied to the images to equalize the scan lines to allow more straightforward stereo matching. The image correction is done with the OpenCV imaging library. [OpenCV, 2011] Geometric distortion correction uses the model

$$\begin{cases} x_i &= f_x x_c \left(1 + k_1 r^2 + k_2 r^4\right) + o_x \\ y_i &= f_y y_c \left(1 + k_1 r^2 + k_2 r^4\right) + o_y \\ r^2 &= x_c^2 + y_c^2, \end{cases} \tag{5.1.14}$$

where (x_c, y_c) are undistorted metric coordinates in the frame of reference of the camera, (x_i, y_i) are distorted image coordinates in pixels, and o_x, o_y, f_x, f_y, k_1, and k_2 are camera parameters listed in Table 5.1.3. Perspective correction is done using projection matrices

$$\mathbf{H}_l = \begin{bmatrix} 1.00000 & 0.00000 & 0.00000 \\ 0.01945 & 1.00000 & 0.00000 \\ -0.00001 & 0.00000 & 1.00000 \end{bmatrix} \quad \mathbf{H}_r = \begin{bmatrix} 0.98461 & -0.00078 & 0.01889 \\ 0.03107 & 1.02367 & -24.66674 \\ -0.00004 & 0.00005 & 1.00000 \end{bmatrix} \tag{5.1.15}$$

for the left- and right-hand cameras.

After the image geometries have been adjusted, the images are processed with stereo matching and reconstruction algorithms that transform the image points to a 3D space. An elevation map

Parameter	Explanation	Left camera	Right camera
o_x	Image center, x	328.13	327.17
o_y	Image center, y	238.99	247.29
f_x	Focal length, x	372.18	370.01
f_y	Focal length, y	372.18	370.01
k_1	2^{nd} order distortion parameter	-0.3026	-0.2924
k_2	4^{th} order distortion parameter	0.0733	0.0680

Table 5.1.3: Parameters of Avant's stereo cameras

is calculated from the 3D points. No location uncertainty information is assigned to the elevation map.

In addition to 3D information, features are extracted from images. By combining the extracted feature vectors with the 3D data, a rich 3D data model is built. Section 5.2.1 goes into details of the building of the 3D model.

5.2 Experiments on building the model

Two test cases were used for evaluating the techniques for building the model. The cases evaluate different aspects of the proposed concept. The primary goal is to demonstrate the capabilities of the concept in collaboration between a robot and a human. In addition, features related to the object recognition and the probalistic representation are evaluated.

The first case tests the building of a shared cognition model based on the robot's sensor data and human input. In this case, the robot shows a snapshot of its sensor data to a user, who marks an object in that. This inputs the object to the robot's cognition. The location is determined on the basis of data inputted by the user and the robot's own navigation system, and the appearance is based on the robot's own sensor data. In time, the robot learns more about the physical structure of the object as it gets more sensor data on the object. This is possible because the robot is able to track the object as it changes its own location and orientation.

The second case evaluates building the cognition model on the basis of description from user. The description does not involve using robot's sensor data, but the user describes a blue box with an attached circle code that the robot has not yet seen. When the robot enters the working area it uses its sensors to find the object on the basis of the description. As the area contains similar looking objects, the robot needs to observe it from other angles to finally find the correct object.

5.2.1 Building a shared cognition model

In this test Avant uses its pair of stereo cameras to build a rich 3D model of the environment. The user defines part of the model to represent an object, a pile of sand, which corresponds to a distinctive formation of sand in real life. The data of the model is gathered to a hypothetical work task that the robot would execute. The task consists of moving the pile of sand to another location. To accomplish this, the task needs to determine the shape and location of the pile. The shape is presented as an elevation map of the ground.

The following description is divided into two parts. In the first part, the user assists the process by defining a segment from the sensor data that includes the pile of sand. The model learns from the definition and from its own sensor data, and builds a cognition model that includes the elevation

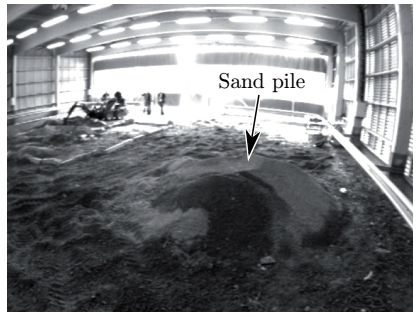


Figure 5.2.1: Photo of the scene of the first experiment. The whole sand pile can be seen in the image.

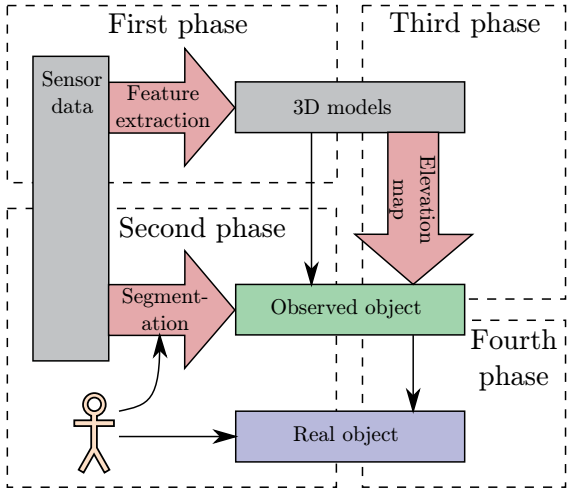


Figure 5.2.2: Phases of processing the first snapshot of sensor data.

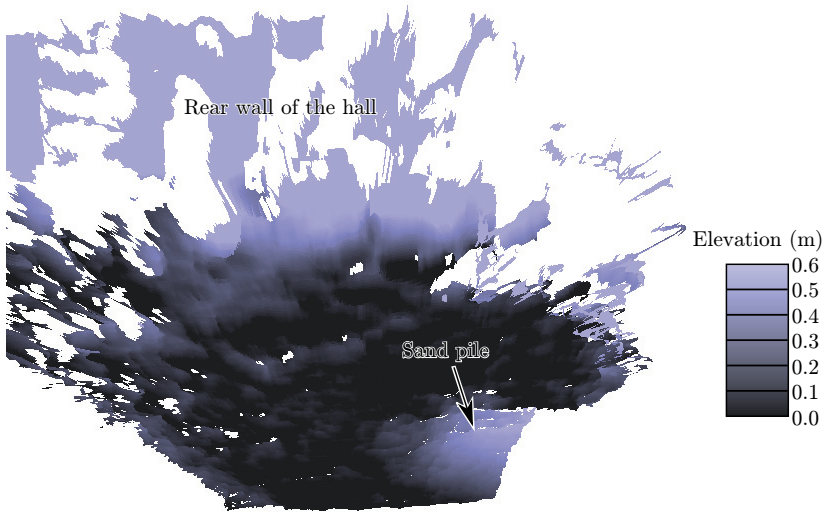
map. In the second part, autonomous operation of the robot is assumed. In the next phase the robot updates its cognition model on the basis of its current state, which is, in turn, based on the user input in the first part.

Figure 5.2.1 shows a photo of the scene. The pile of sand is annotated to the image. In the first phases of this experiment, the whole pile is not visible to the sensors.

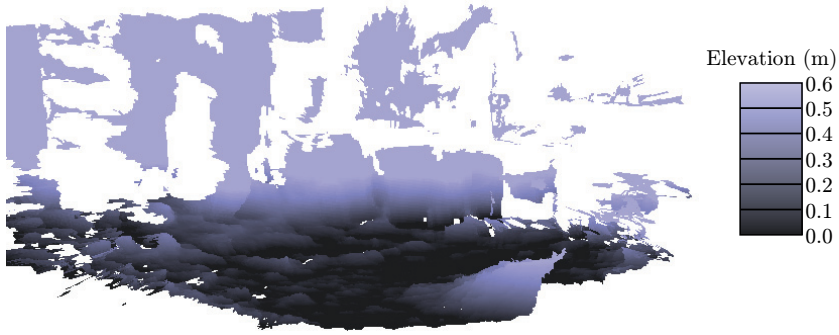
5.2.1.1 Collaboratively creating the cognition model

This subsection describes what happens initially on the basis of the first snapshot of the sensor data. Figure 5.2.2 shows a conceptual block chart of the phases.

First phase: building raw and rich 3D models The test starts with building a rich 3D model of the scene viewed by the robot. A 3D model is built from the camera images. A dense disparity map is built on the basis of correlation matching. Then the 3D points are reconstructed on the basis of the rectified geometry. The building of a 3D map based on stereo images is described in Algorithm 2.1 on page 16. The left-hand camera image is used as the reference. In addition to the



(a) Raw 3D data points viewed from above and slightly to the left from the observer.

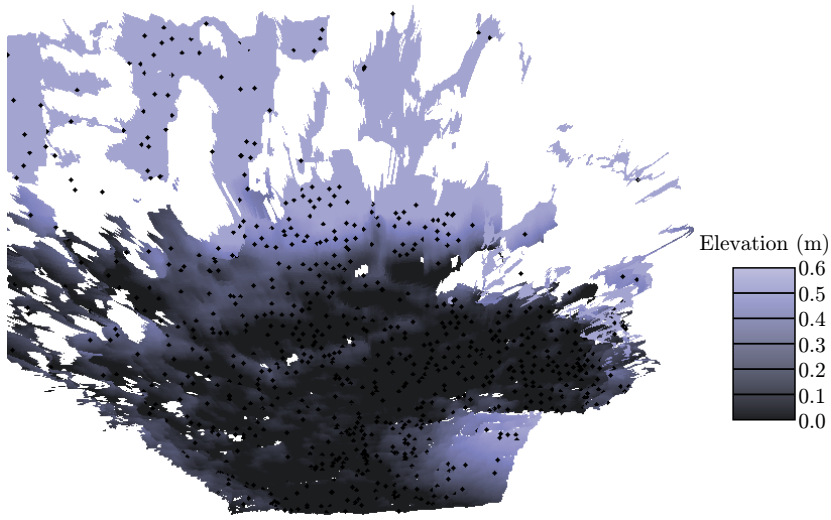


(b) Raw 3D data viewed from slightly lower position.

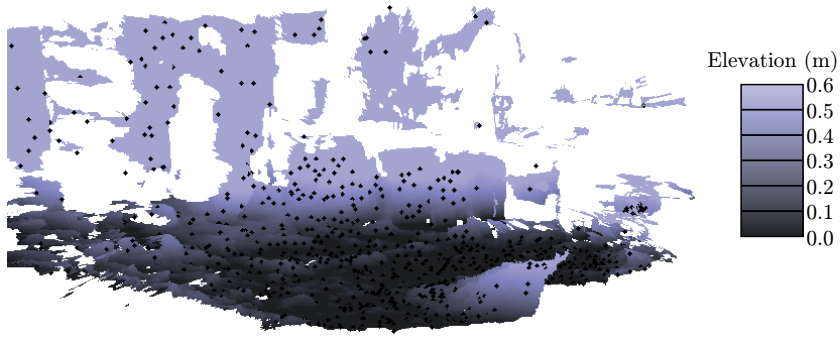
Figure 5.2.3: Raw 3D data points calculated from a stereo image pair. Colors indicate elevation. Higher areas are coded with blue color. Data is rendered from two slightly different angles for easier visualization. Part of the sand pile can be seen in the front right side corner of the data. Especially the subfigure 5.2.3b shows that the actual pile is partially outside the measured data. On the background, the wall of the hall can be seen.

building of the 3D model, the SIFT features are extracted from the left-hand camera image. Their 3D coordinates are calculated by comparing their image coordinates to the coordinate map. This procedure is described in Section 4.1.1.2. The dense 3D point map is thus the raw 3D model, and the SIFT features with their 3D coordinates represent the rich 3D data. Both types of data are needed in later phases. Figure 5.2.3 shows an illustration of the raw 3D points, and Figure 5.2.4 shows the rich 3D data. The locations of the SIFT features are marked onto the image.

The output from this phase is perception data, and thus not yet part of the actual cognition model.



(a) Rich 3D data of the scene viewed from above and slightly to the left from the observer.



(b) Rich 3D data viewed from different angle

Figure 5.2.4: Rich 3D data set calculated from the same stereo image pair as in Figure 5.2.3. Black dots represent locations of SIFT features assigned to the model. The data is shown from two different directions.

Second phase: user interaction The 3D model is shown to the user. His aim is to teach the robot where the pile of sand is and what it looks like. On the basis of his spatial understanding and interpretation of the robot's sensor data, he defines a section of the data that represents the pile of sand. The entire pile is not visible in the sensor data, and therefore the boundaries of the user-defined pile object extend to an area that has not yet been covered by the sensors. By comparing the robot-generated model and the real-world pile, the user is able to estimate where the real boundaries of the pile are in the model.

Figure 5.2.5 shows the pile and boundaries defined by the user. The figure shows a zoomed-in detail of the scene shown in Figure 5.2.3. In addition to the boundaries, the user defines that the extraction of the elevation map is required for the object.

Defining the boundaries creates an observed object. The sensor data that are inside the defined boundaries are associated with the observed object. The coordinates of the boundaries are stored

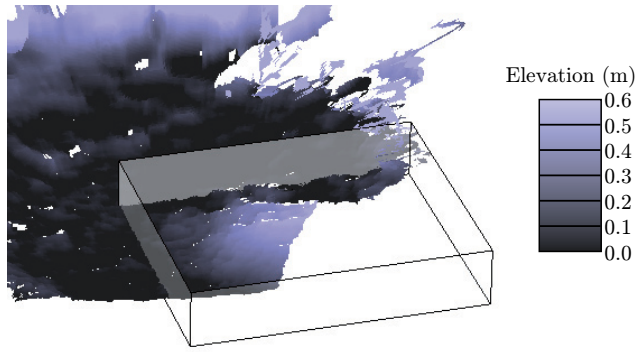


Figure 5.2.5: Boundaries of the pile defined by user.

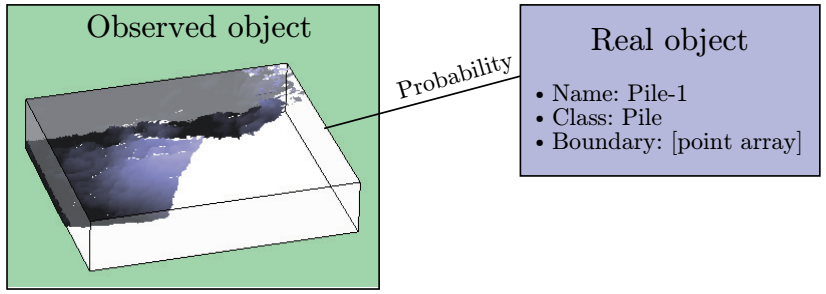


Figure 5.2.6: Real and observed objects after user has segmented and connected the data to the real object. The probability is set to 1.

to the object.

A real object is also created. Whereas the observed object represents the robot's perception of the pile, the real object represents a higher understanding of the pile. The class of the real object is "Pile"; it is given the name "Pile 1", and the boundary coordinates are also stored to the real object.

The real and observed objects are connected with a strong link with probability 1, because the recognition is based on user input and is thus considered certain.

Figure 5.2.6 shows the topology of the model after user interaction.

Third phase: measuring the elevation map The user has defined in the real object that an elevation map is needed. The observed object does not yet contain the elevation map, and it needs to be calculated. The elevation map is calculated on the basis of the raw 3D data within the boundaries of the observed object. Section 4.1.1.3 describes the calculation of the elevation map. The elevation map is stored in the observed object. Figure 5.2.7 shows the state of the elevation map.

Fourth phase: learning the appearance In this phase features are transferred from the observed object to the real object. Not all the perception data are transferred, only rich 3D features and the elevation map. The raw 3D data are not needed after this, and are not transferred to the real object.

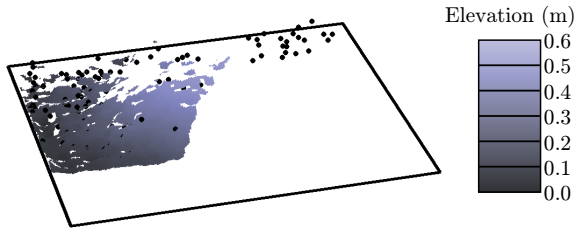


Figure 5.2.7: Elevation map and SIFT features after the first measurements

After the learning phase the real object contains all the information that can be extracted from the sensor data in one instant in time.

5.2.1.2 Higher-level planning based on data

A higher-level task can now utilize the data in the real object. The task should consist hierarchically of two levels: perception planning and manipulation. The perception planning level takes care that the robot has enough information to execute the manipulation. Perception planning as such is not part of the model, because it is heavily dependent on the actual task, and needs to be implemented as a part of the task itself. However, the perception planning can be executed entirely on the basis of the information in the cognition model. Any missing parts can easily be determined by examining the real objects. If a part of a physical entity is not seen, part of the cognition model is incomplete.

In this case it is assumed that the task examines the elevation map and detects that parts of it are missing. In the test scenario, the sensors were turned directly towards the pile to simulate a case where the robot focuses its attention on the target, and maximizes the area covered by the sensors.

5.2.1.3 Refining the model automatically

The cognition model learns as more data are gathered. The next phases describe how the model is updated as the robot moves and gathers more sensor data. These phases are repeated for each round. Figure 5.2.8 shows the phases described below.

Fifth phase: building new 3D model After the robot has changed its pose, a new stereo camera pair is analyzed. Again, rich and raw 3D models are built from the stereo camera data.

Sixth phase: automatic segmentation On the basis of information from the real object, automatic segmentation can now be carried out. The segmentation is based on the known location of the object in world coordinates. If the robot's navigation system were perfect, nothing else would be needed. However, because there is always some inaccuracy in navigation, the exact location needs to be determined with recognition.

The object is recognized from the sensor data based on matching rich 3D features of the real object and the current snapshot of the sensor data. The robot's current location is used as an *a priori* assumption for matching the features. As a result, the exact location of the known features in the current sensor data can be determined. After the location has been determined, the sensor data are segmented using the boundaries of the real object. This is illustrated in Figure 5.2.9.

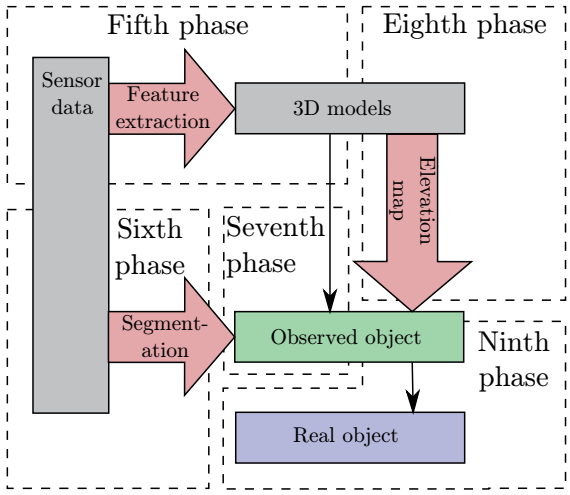


Figure 5.2.8: Phases of processing the sensor data on next time instances.

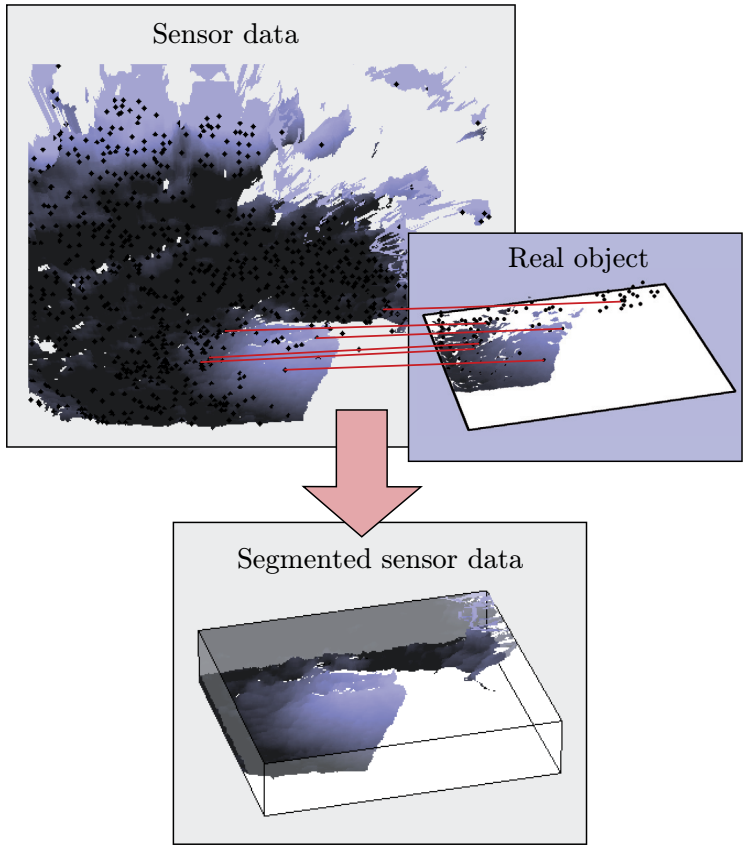
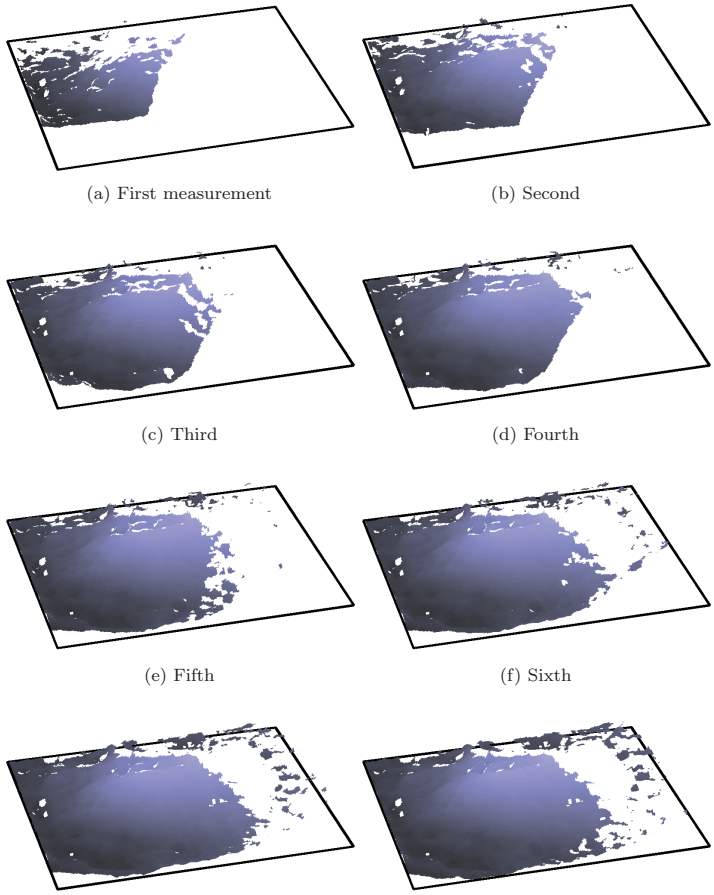


Figure 5.2.9: Automatic registration and segmentation of the pile from new sensor data. The real object information is from the previous sensor data. The sensor data shown in the figure is from the new measurements.



(a) First measurement (b) Second

(c) Third (d) Fourth

(e) Fifth (f) Sixth

(g) Seventh (h) Eighth

Figure 5.2.10: Elevation map in different phases of the task execution. In each phase, all the previous information is accumulated to the map, and more area is mapped. The area on the right hand side of the region is still occluded even after the eighth phase and therefore it remains unknown.

Seventh phase: matching to previous objects In this phase the new sensor data are either associated with a previously-existing observed object, or a new observed object is created. Because the new segmented sensor data are known to represent a certain real object that is connected to an observed object that already exists, the new information from the sensor data will be fused with the information in the observed object that already exists.

Eighth phase: measuring the elevation map As previously, the connected real object indicates that an elevation map needs to be calculated on the basis of the 3D data.

Ninth phase: learning the new appearance The extracted rich 3D features and elevation map are updated to the real object. The new rich 3D features are compared to the previously-

Figure 5.2.11: Target box of the task.

existing features. The features that exist in the observed object, but not in the real object, are copied to the real object. The elevation map is fused on the basis of choosing the lower elevation for each measured point on the map. In this way any moving objects that may have been detected in one of the rounds will be discarded, and only the ground formations are left. Previously unmapped areas get new information from the observed object.

5.2.1.4 Development of the elevation map over time

The procedures described above generate an elevation map whose state depends on the sensor data available. Figure 5.2.10 shows the state of the elevation map at different instants in time. The first map is generated with the phases described in Subsection 5.2.1.1, and the next maps are updates, generated with the phases described in Subsection 5.2.1.3.

5.2.2 Recognizing objects on the basis of a description from the user

This test evaluates the recognition of objects on the basis of human input. Robot sensor data is not used in describing the object, but the user describes an object that the robot has not yet seen. The user input is transformed into the robot's understanding of the objects' appearances and topologies.

Based on the description, the robot searches for the object and measures its location. Besides the target object, the scene contains other objects that look similar to the target. But these other objects do not require any action from the robot other than avoiding a collision with them. The object in question is a blue box of a known size with a circle code attached to one of its sides. Figure 5.2.11 shows the object. Figure 5.2.12 shows the setting with the target object along with the other objects present at the scene.

The task starts by defining the object's appearance to the robot. In the next phase, the robot observes the scene and measures the locations of the potential target objects. Then the robot starts moving. While moving, the robot updates the cognition model on the basis of the perception. The robot moves until it is sure that it has found the target object. In the task setup, a human intervenes in the middle of the task execution and changes the setting.

The topology of the model is inspected during different phases of the task execution.

Figure 5.2.12: The scene in the beginning of the task. The blue box on the left is the target box whose appearance the user described. The code is not visible from this angle. The code seen on the plastic box encodes different value than the code that the user described.

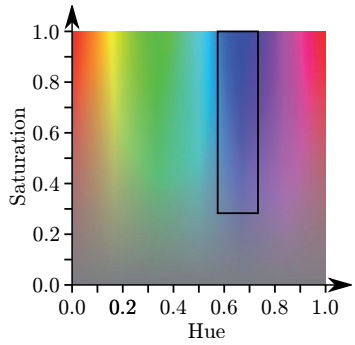


Figure 5.2.13: Hue-Saturation -chart

5.2.2.1 Describing the appearance

The user creates a real object for the box and describes its appearance by describing its color and its physical height. In addition, the user creates a real object corresponding the circle code, and defines a topological "attached to" connection between the box and the code.

Defining the color requires a common ground for the robot and the human. Basically, the robot interprets the colors by numbers, while humans usually describe the colors using verbal descriptions. A color map in the graphical user interface can be efficiently used as a common representation for humans and robots. Figure 5.2.13 shows the color map used in this experiment and the color of the box defined by the user.

The height is defined in metric units as 0.25 meters. The robot is able to measure the approximate physical size of the box, therefore it is possible to use the metric size as a description. If the robot was not equipped with such a sensor, only visual features could have been used to describe the object.

The circle code is defined by the numeric representation encoded in the circle code. As the code is generated by a computer program, the corresponding number is known by the user.

Figure 5.2.14 shows the defined real objects, their connection, and the corresponding parame-

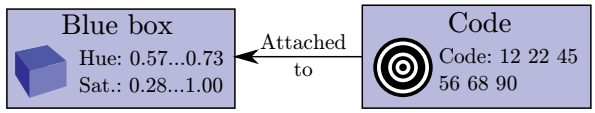


Figure 5.2.14: Real objects that define the appearance of the box and the attached circle code.

(a) Image of the scene captured with the camera of the robot

(b) Blue regions segmented from the camera image

Figure 5.2.15: Image of the scene captured with the camera of the robot.

ters.

5.2.2.2 Looking for objects at the initial location

The robot starts the task execution by looking around the working area and searching for the target object. Figure 5.2.15 shows the scene initially viewed with the robot's camera.

The image is segmented to create observed objects. The segmentation is based on the color description of the real object corresponding to the target box. The segmentation is performed in the Hue-Saturation color space. Thus, brightness differences (represented in the Value channel) are ignored because the brightness is heavily affected by illumination. Figure 5.2.15b shows the blue regions after the segmentation. After the segmentation, regions that are too small are filtered out, and convex hulls of the regions are calculated. Observed objects representing the remaining regions are created. In addition to the blue areas, circle codes are located in the image. One code is found, and a corresponding observed object is created. The value encoded in the circle code is read and stored in the observed object. The initial estimates of the locations of the observed objects are calculated with Equation 5.1.10. The uncertainties of the locations are calculated using Equation 5.1.7. Figure 5.2.16 shows observed objects annotated to the image and labels indicating the parameters of the objects.

After estimating the initial locations, the more accurate locations are measured using the laser pointer of the robot. The robot points to each object in turn and measures their locations. Algorithm 5.1 describes how the pointing is done in practice. Using the laser pointer results in much more accurate measurements than when using the camera alone, greatly reducing the uncertainty of the locations.

After measuring the locations, the approximate physical sizes of the objects can be calculated

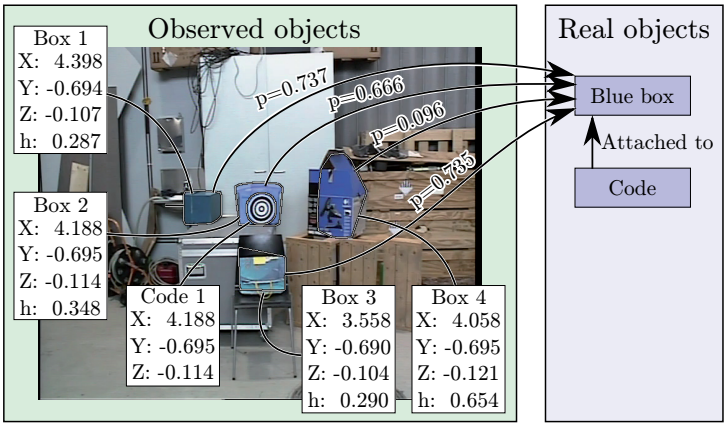


Figure 5.2.16: Observed objects, their measured parameters, the connections to corresponding real objects and the probabilities of the connections.

using Equation 5.1.10. The physical size is only approximate because only the size of the projection is calculated. The projection is affected by the perspective, and therefore the measured height may be higher than the actual height. The final measured locations and sizes of the objects are also shown in Figure 5.2.16.

The next phase is to match each observed object to the real objects. Each blue observed object is matched to the real object representing the box. The match probabilities between the real object and each observed object are determined.

The color matching probability is equal for each of the objects. The difference between the measured and reference colors could have been used as a measure of reliability. However, the colors may differ for various reasons, such as an inaccurate color description in the first place, variations in lighting, or an improperly adjusted white balance in the robot's camera. That's why the "exactness" of the color is not used to measure the certainty of the match.

The size of the object, on the other hand, is used to measure the probabilities of the matches. The measured sizes are compared to the reference inputted by the user. Probability is assumed to be normally distributed around the reference size with standard deviation $\sigma = 0.1$m. Thus the probability of the match is calculated with

$$P = ke^{-\frac{\left(s_{ref} - s_{meas}\right)^2}{2\sigma^2}}, \tag{5.2.1}$$

where s_{ref} is the reference size, s_{meas} is the measured size, and k is the scaling factor. The maximum probability is assumed to be 0.75 if the size of the object matches exactly the reference size. Therefore $k = 0.75$ is used.

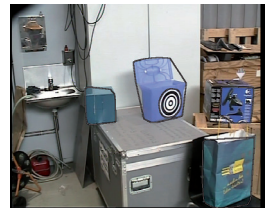
The detected code object is compared to the real object corresponding to the circle code. Because the codes differ, the observed object is not matched to the real object. There are no other real objects describing circle codes, and therefore the observed object is not matched to any existing object. A new anonymous object is created for the observed code object.

The probabilities of the matches are also shown in Figure 5.2.16. None of the objects is clearly more probable to be the target box than the others. Therefore, it cannot be concluded that the object has been found.

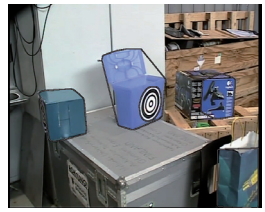
(a) After moving 0.43 meters from start

(b) 1.13 m from start

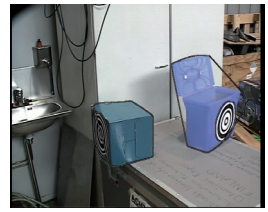
(c) 1.91 m from start



(d) 2.31 m from start, user has changed the setting



(e) 2.93 m from start



(f) 3.36 m from start

Figure 5.2.17: Snapshot images from the camera of the robot during the execution of the task.

5.2.2.3 Tracking the objects while moving

In the next phase, the robot starts to move. The decision to start moving is done on a higher level task. While moving, the robot looks at the objects, providing a view from different directions. Figure 5.2.17 shows the snapshots of the robot's camera image during the task execution. Table 5.2.1 shows the task parameters corresponding to each of the objects during the task execution.

Measuring the location using the laser pointer is a slow procedure. Therefore the robot does not measure the locations with the laser, but rather estimates the locations and sizes on the basis of the camera image and the robot's self-localization (that is, the locations are estimated from the image), and an offset from the robot's own pose is added. If the locations and sizes are consistent with the previous measurements, the robot assumes that nothing has changed. Some of the objects are not always visible. They remain in the memory even if the robot does not see them.

The changing viewpoint also affects the match probabilities between the observed objects and the real object. When viewing from different directions, the projections of 3D objects onto a camera image vary. Because of this, the observed heights of the objects vary over time. In addition, different viewing angles change which parts of the objects are visible.

In the middle of the task execution, a human enters the working area. He opens the big box, closes the plastic box with the attached code, and changes the shape of the paper bag. This causes

Sub-figure	Heights of the observed box objects (m)				Probabilities of the matches to real object			
	1	2	3	4	1	2	3	4
(a)	0.279	0.344	0.279	0.657	0.742	0.672	0.742	0.094
(b)	0.297	0.359	0.255	0.666	0.730	0.646	0.750	0.086
(c)	0.323	0.394	0.301	0.593	0.701	0.579	0.726	0.172
(d)	0.349	0.602	0.541	0.365	0.664	0.160	0.260	0.635
(e)	0.420	0.676	0.428	0.425	0.523	0.077	0.505	0.512
(f)	0.464	0.762	-	-	0.424	0.028	-	-

Table 5.2.1: Measured heights of the objects and the probabilities of the matches to the real object corresponding the blue box. The numbers of the objects correspond to the observed objects in Figure 5.2.16. The Subfigure-column indicates which instances presented in Figure 5.2.17 the rows correspond.

Figure 5.2.18: A human changing the setup.

the observed sizes of the corresponding objects to change. Now the sizes need to be calculated again. The locations of the objects whose sizes have changed now need to be measured again using the laser pointer. It is not enough to use the previously measured locations. It is not certain that the objects are the same ones as before because their sizes differ. The robot needs to stop and measure the locations of the changed objects. Each location measurement is compared to the previously observed objects. Because the locations are consistent with the previous measurements, no new observed objects are created, but the old ones are merely updated. The change in the shapes of the objects did not change the interpretation of the identities of the objects. Figure 5.2.18 shows the human taking something from the paper bag, thus changing the configuration of the objects.

The human is not detected by the robot. The robot only detects the objects that it needs to detect. A human-type object is not actively sought (and not detected) because the task definition only includes real objects that correspond to the blue box and the circle code.

When the robot has moved forward about 2.8 meters, the other circle code is revealed behind the blue box. As soon as the robot is able to read the code, the corresponding observed object is created. Now the numeric code matches the one defined in the real object, and the observed object and real object are matched with high probability. This probability is much higher than any of the matches already present. The location of the code is measured with the laser pointer. Because

the distance between the code and the corresponding blue box is small (0.132 meters), the code is considered to be attached to the box. Because of this, the match between the box observed object and the real object is updated to the same probability as the match between the observed and real objects corresponding to the code. This means that the target object is now found.

5.2.2.4 Utilizing the information in a higher level task

The above experiment assumes that the decisions to move the robot or its sensors to gather more information are done on a higher level task. The task makes the decisions on the basis of the state of the cognition model. Viewing the initial scene is not sufficient to make a conclusion about which object is the correct one. To make such a conclusion, the object must be viewed from different angles. The robot planned its actions to gather more information; that is, the robot executed some perception planning.

When the human intervened on the scene and changed the configuration of the objects, the state of the cognition model changed. This resulted in increased uncertainty in the model. The robot executed actions to reduce the uncertainty by measuring the accurate locations of the objects whose configuration had changed.

The final conclusion that the object was found was made based on the probability of a match between the real object and the corresponding observed object.

5.3 Experiments on applying the model in task execution

The following two experiments expand the evaluation of the model. In addition to the model itself, these experiments also evaluate its applicability to concrete interactive tasks performed in cooperation with a human operator. The first case evaluates the picking up of ambiguously defined discrete objects – pieces of litter from the ground. In the second case, the robot unloads a pile of boxes whose appearance is taught to the robot, but the order of the task execution is taught by a human operator using real-world pointing methods.

Both of the experiments were done with WorkPartner robot. The robot's actions were performed manually by the author. The actions tried to simulate the behavior of an autonomous robot as accurately as possible. The task described here requires a relatively high level of intelligence. The various exceptions encountered during the execution and the related handling mechanisms are described. The tasks are not described in a detailed form (such as a flowchart), but the essential phases are described verbally.

In the following two experiments, the human involvement and the cognition model were evaluated. This is an essential part of the performance of the shared cognition model, as one of its main purposes is to function as an intermediator between the human and robot in sharing the spatial cognitive information.

5.3.1 Picking up litter

This test evaluates object recognition assisted by a human operator. The test compares two different approaches for human assistance. In both cases, WorkPartner inspects an area and locates possible pieces of litter on the ground. In the first case, the human looks through the objects and decides which suspected objects are actually pieces of litter. Then the robot starts collecting the pieces from the ground. In the second case, the robot starts collecting the pieces

Figure 5.3.1: Photograph of the area where the pieces of litter are collected.

immediately without showing them to the user. In both cases, user assistance is requested when problems occur.

The purpose of this test is to evaluate how the model is used in the presence of uncertainty, and how the user can interact with the model to increase certainty. In particular, the user rejects false matches. This enables the utilization of human cognition in executing the task.

Figure 5.3.1 shows a photo of the scene. The pieces of litter are three foam plastic cubes, one coffee cup, two small juice cartons, and one snack container made of plastic.

The following description is divided into five subsections. The first subsection describes the task at hand. The second subsection describes the detection and localization of possible pieces. The third subsection describes the first case with immediate user interaction. The fourth subsection describes the second case with user interaction only when problems occur. The fifth subsection compares the two cases.

5.3.1.1 Task description

The goal of the task is not to pick up specific pieces of litter, but to collect any number of pieces whose exact properties are not known beforehand. Meta-objects typically carry this kind of information. Real objects would tie the objects to specific physical entities, which is not true in this case. The target objects are thus described using meta-objects.

The appearance of the objects is defined through how they differ from the background. The ground is assumed to be reasonably even, and the pieces of litter look distinctive. The ground and the pieces of litter are assumed to be separated by visible edges. Therefore the appearance of the objects is defined to be anything surrounded by edges. In addition, the objects can only be found in a certain area. Figure 5.3.2 shows the area defined in the world coordinate system.

5.3.1.2 Detection of potential pieces of litter

It is assumed that the robot has just arrived at the scene. Thus the area is initially clear of observed and real objects.

Creation of observed objects The first phase of detection is the segmentation of the sensor data and creation of the corresponding observed objects. The segmentation is done based on what

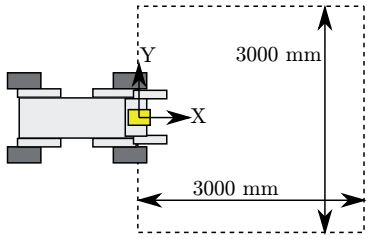
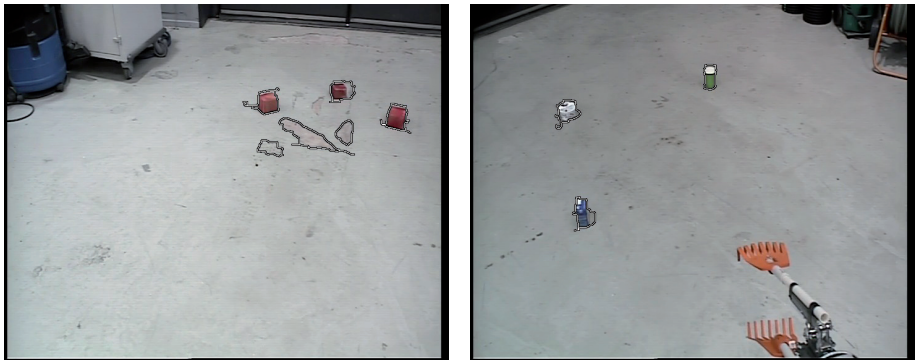


Figure 5.3.2: Diagram of the working area viewed from above



(a) The first camera image (b) The second camera image

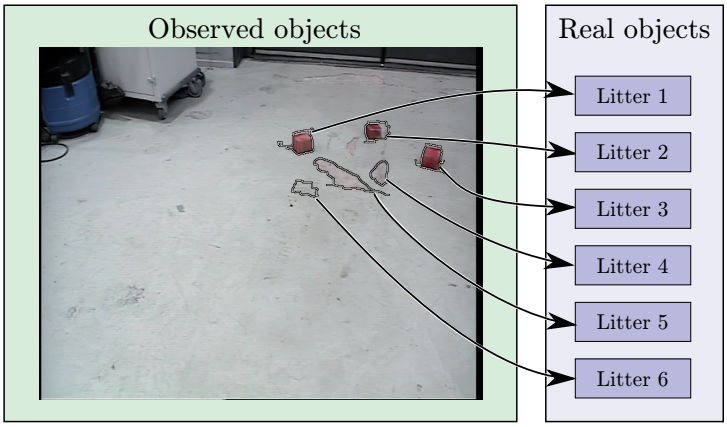
Figure 5.3.3: Detected potential pieces of litter in two camera images taken to different directions

objects are being sought. As the task description only defines the "piece of litter" meta-object as an object of interest, the corresponding edge-based segmentation method is used. In practice, the camera image is divided into regions separated by edges, and each region is considered a separate entity. The edges are extracted with the Canny edge detector, and separate segments are connected with morphological closing.

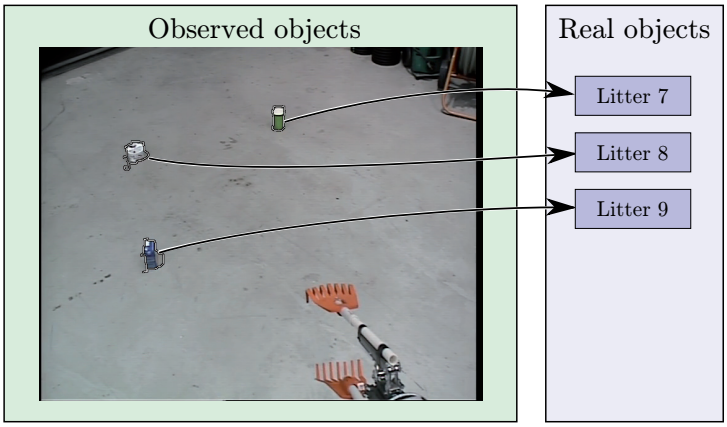
To determine which part of the camera image represents the predefined working area, the projection from the ground to the camera coordinates is calculated with Equations 5.1.11, 5.1.12, and 5.1.13. The region is projected in the image, and only the corresponding part of the image is processed when detecting objects. Figure 5.3.3 shows the detected regions and the working area in two images taken from different directions.

The appearance features used for object identification are extracted after the segmentation. The objects are identified by color. The color histograms are extracted from the segmented regions. The histograms are three-dimensional arrays representing the hue, saturation, and value channels of the regions. A histogram is quantized to 50 units in each dimension, thus leading to 125,000 bins for each object. Another type of appearance feature used for visualization is an extract from the camera image. Therefore, for each object, the image inside the bounding box of the corresponding region is stored in the observed object.

Creation of real objects In the second phase, the observed objects are compared to the meta-objects. If the features of an observed object match a meta-object, a new real object of the corresponding object class is created. In this case, a new real object is created for each observed



(a) The first camera image



(b) The second camera image

Figure 5.3.4: Observed objects extracted from the two camera images, and the corresponding real objects.

object. Because the detection is based on a relatively uncertain method, the match certainty between the real and observed is set to a low value of 0.5. This informs the system in the later phases that the match is not yet completely certain. Figure 5.3.4 shows the observed objects, which regions of the image they correspond to, and the respective real objects.

When a new real object is created on the basis of a meta-object, the appearance of the object needs to be determined. Therefore all the appearance information from an observed object is copied to the real object.

Localization of the objects In the third phase, the objects are located using the available ranging sensors. In this case, the laser pointer is the most applicable sensor for this purpose. The location of each region is measured separately. The robot recognizes the objects by comparing the color histograms of the real and observed objects, and by measuring their distances. Localization based on a laser pointer is described in Algorithm 5.1. Table 5.3.1 presents the locations of the objects and an approximate description of the color histograms used for the identification.

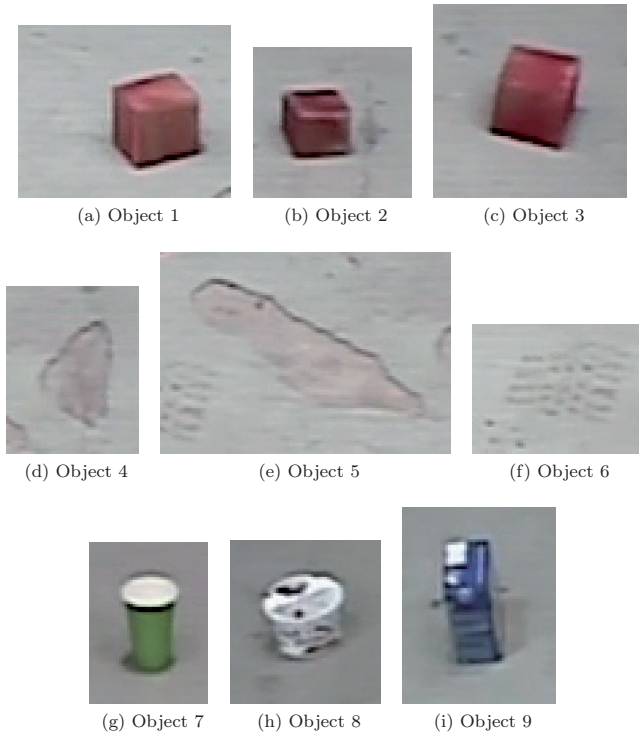
	Location of the object in meters			Approximate verbal description of the color
Object	X	Y	Z	
1	2.321	0.730	0.962	Red
2	2.601	0.721	0.627	Red
3	2.466	0.709	0.289	Red
4	2.273	0.825	0.516	Gray
5	2.184	0.832	0.691	Gray
6	2.009	0.835	0.846	Gray
7	2.243	0.719	-1.347	Green-white
8	2.464	0.738	-0.594	White-red
9	1.438	0.676	-0.876	Blue-white

Table 5.3.1: Measured locations of the objects and verbal description of the measured color histogram.

5.3.1.3 Utilizing human cognition in picking up the objects

Asking the user for confirmation After the robot has found the possible pieces of litter, it needs to determine which ones are true matches and which ones are false matches caused by noise, an irregular background, or uneven illumination. The uncertainty is presented by the match probability. The system looks through the matches and asks the user for confirmation for objects whose match is not certain. In practice, the snapshots stored in the real objects are shown in the user interface. The user then chooses which ones are actual pieces of litter. Based on the user's selection, incorrect matches are discarded, and the corresponding objects are deleted. The probability of the matches confirmed by the user is set to 1.0. Figure 5.3.5 shows the candidate object images shown to the user.

Picking up the objects After the confirmation phase, the robot starts to pick up the pieces of litter. The robot collects the pieces one by one. It starts by approaching the measured location of the first object. When it is near the object, it moves the manipulator to an orientation that allows the gripper to reach the object. Figure 5.3.6 shows the orientation of the manipulator when the robot has approached the first piece of litter. The orientation of the manipulator is calculated with the kinematic transform of the robot. When picking up the piece of litter, the manipulator is controlled so that the gripper reaches the measured coordinates of the object. However, the self-localization of the robot introduces considerable error in the location. Updating the previously measured location with the robot's measured movement is not enough for controlling the manipulator. Therefore, before picking it up, the robot needs to measure the location of the object again. To be able to point the laser to a correct object, the robot now needs to identify the object again. The recognition is done on the basis of the appearance description stored in the real object. The object is located again as described in Algorithm 5.1. New observed objects are created for each identified objects, and they are matched to the real objects. The probability of a match is now set to 1.0. Even though the recognition is not 100% reliable, the procedure described in this experiment does not validate the recognition, and therefore this recognition is considered as certain and is directly used to control the picking subtask. Figures 5.3.7 and 5.3.8 show the observed and real objects related to each piece of litter. As can be seen in Figure 5.3.8a, the mug is not identified correctly. This is discussed below.



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

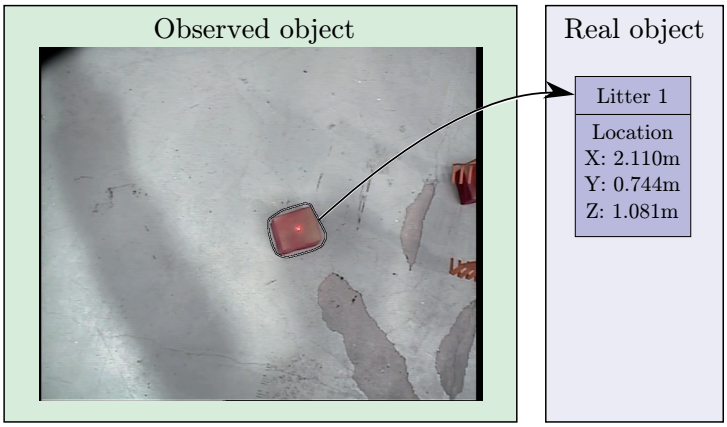
(d) Object 4 (e) Object 5 (f) Object 6

(g) Object 7 (h) Object 8 (i) Object 9

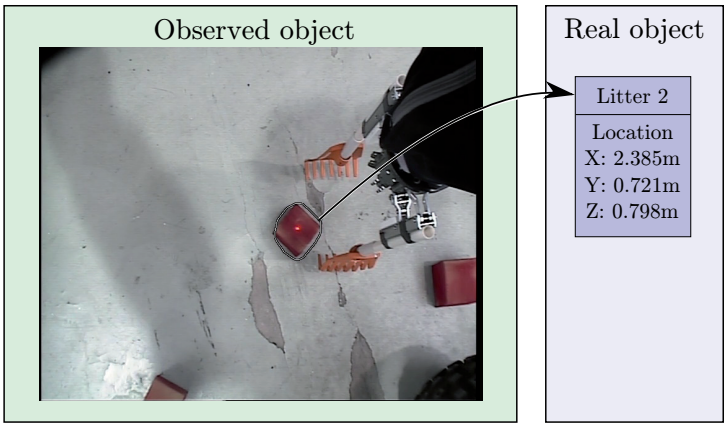
Figure 5.3.5: Candidate objects shown to the user

(a) Approaching the area (b) Bending down and measuring the location (c) Picking up the object

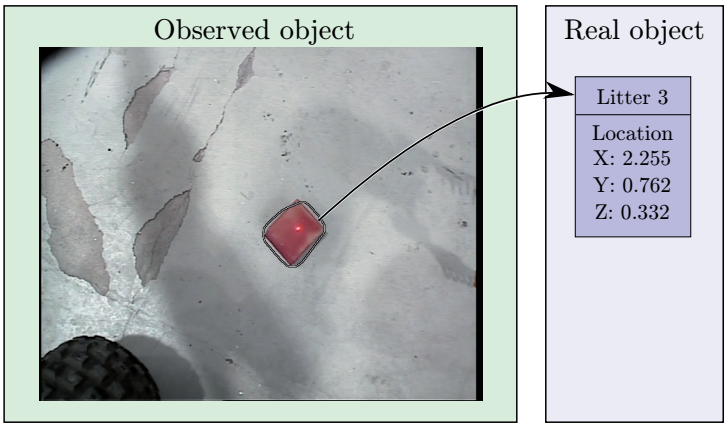
Figure 5.3.6: Phases of picking up a piece of litter from the ground.



(a) Object 1

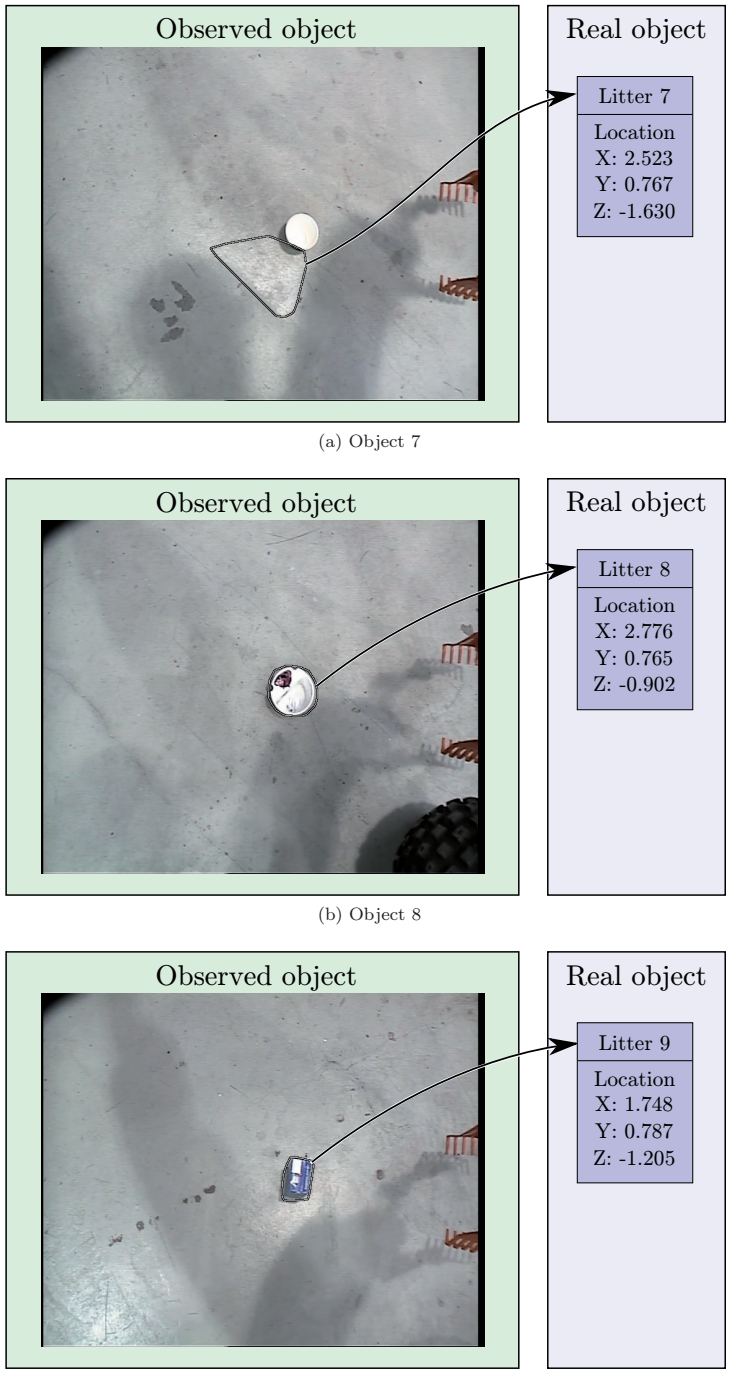


(b) Object 2



(c) Object 3

Figure 5.3.7: First three pieces of litter viewed with camera when the robot is ready to pick up the objects. The object names refer to those used in Figure 5.3.4. Figure 5.3.8 shows the next three pieces.



(a) Object 7

(b) Object 8

(c) Object 9

Figure 5.3.8: Another three pieces of litter viewed with camera when the robot is ready to pick up the objects. The object names refer to those used in Figure 5.3.4. Figure 5.3.7 shows the first three pieces.

After picking up the object, the robot first tries to determine if it can reach other objects without moving itself. This is examined by comparing the robot's current location, the locations of the other objects, and using the robot's kinematic calculations to determine if the objects can be reached. If the robot can reach an object without moving the wheels, it adjusts the manipulator accordingly, and repeats the aforementioned procedure. If there are no more objects within reach, the robot backs up to the starting position, and then approaches the next object and collects it. When there are no more objects, the robot returns to the starting position.

Problems during the task As the color of the object was originally determined from a long distance, the color histogram may not identify the object perfectly when viewed from close range. The difference may be caused by a different viewpoint, variation in the illumination, or noise. In particular, this causes a problem with a mug. When viewed from a distance, the camera saw the green side of the mug and a bit of white inside the mug. But when viewed from above, the robot sees only the white interior. Furthermore, when viewed from a distance, the interior looked gray, but from above, it is completely white. Therefore, the robot fails to correctly identify the mug and moves the manipulator to a completely wrong position. As a result, the robot is not able to collect the object.

This kind of problem again needs help from a human. The robot is not able to know if the detection was correct before actually trying to pick up the piece. If the object still seems to be there after picking it up, the robot is able to detect that the action failed. Now the robot can ask for the human operator to help in recognition. This is done by rejecting the earlier appearance information and segmenting the camera image on the basis of the edge extraction only. The human is asked to choose the correct object from the segmented areas. After this selection, the appearance information of the real object is updated and the robot is able to resume the work.

In many cases, it would be possible to detect the presence of an object by measuring its distance from the ground. If the object is flat, it is probably not a real discrete object, but just an illusion. The problem is, however, that in this case, the challenging object was a mug that when measured with a laser pointer may appear to be flat because the laser pointer probably hits the bottom of the mug. This could be solved by scanning a larger area with the laser pointer or using a different kind of sensor, such as a stereo camera pair. In this setup, these options were not applicable.

Except for the mug, all the other objects were successfully recognized and collected in this experiment. There were also three objects that did not correspond to real pieces of litter. These are discussed below.

5.3.1.4 Postponing the user interaction

In the other version of the test case, the robot was not assisted by a human before problems occurred. The robot assumed that all the detected objects were pieces of litter, or at least it did no harm trying to collect them. Therefore, the robot started collecting the pieces one by one, as described above. If the robot encountered an object that did not seem to disappear even though it was collected, the robot showed the camera image and asked the human to verify the object's existence and mark its location if it really existed.

Problems during the task There were three false matches that looked like pieces of litter, but were in fact just cracks in the floor surface. These areas are named Objects 4, 5, and 6 in Figure 5.3.4. Figure 5.3.9 shows the robot trying to grip an object that is falsely classified as a piece of

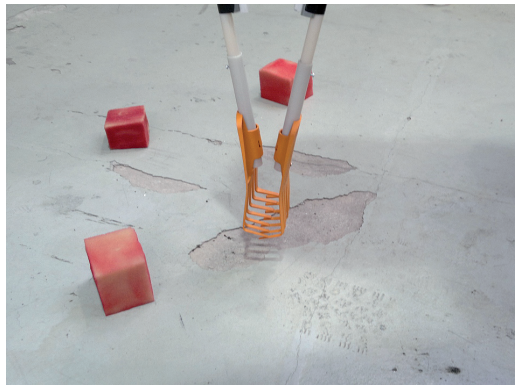


Figure 5.3.9: Robot trying to grip an object that it thinks is a piece of litter. After the pickup procedure, the robot observes the object again intact and asks for user assistance.

litter. Figure 5.3.10 shows all the problematic objects viewed by the robot's camera at the pickup position. These images were shown to the user, who then discarded all the cracks and marked the correct location of the mug.

5.3.1.5 Comparison of the cases

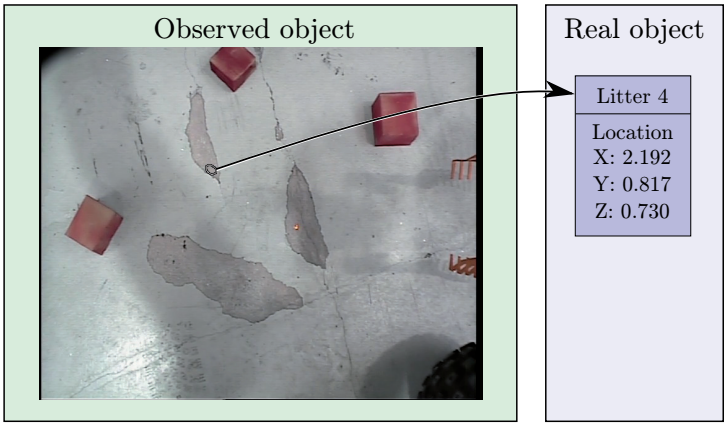
There were two different approaches to the task execution: (1) utilizing human cognition in the beginning of the task execution, and (2) postponing the user interaction until problems occurred.

The cognition model was an integral part of the task execution because information about the identities and locations of the objects was exchanged through it. Color information was a relatively effective way to store appearance information for object recognition. And most of the objects were recognized correctly when viewed from a close distance, even when the appearance was taught to the model from a longer distance and a completely different viewpoint.

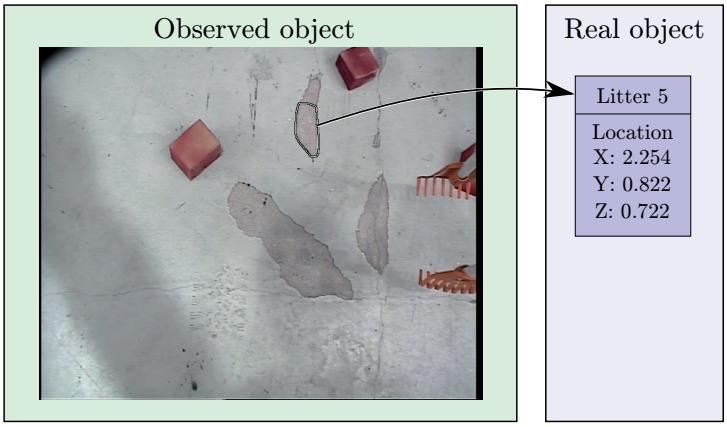
The behavior was different in the case of false matches. When the human assisted the robot in the beginning, the human had to go through all the candidate objects, but the robot did not need to approach the false matches. In the other case, the human only had to concentrate on the cases that were problematic. As it was not possible to unambiguously define the appearance of the pieces of litter, it was necessary to assume that the recognition may fail and human assistance was needed. It was not possible to assume that the robot would have been able to execute the whole task without a human. The comparative effectiveness of these two approaches depends on the application. In this case, the fact that the robot was able to avoid necessary moving made the entire execution time shorter when the user had already discarded the false matches in the beginning. In addition, every time the human had to assist the robot, he or she needed to interrupt the current task. Therefore, it's usually most effective to assist the robot as much as possible in the beginning, even if that means giving a little more assistance.

5.3.2 Carrying boxes

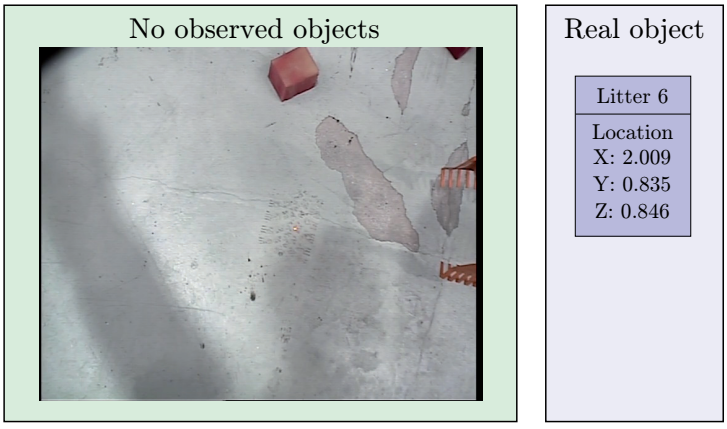
This test presents a case where the user actively guides the task execution. The test compares different real-world pointing methods and how they are handled in the model. The test case involves unloading a pile of boxes in a certain order. The order is determined by the user. Two



(a) Object 4



(b) Object 5



(c) Object 6

Figure 5.3.10: Incorrectly detected pieces of litter viewed with camera when the robot is ready to pick up the objects. The object names refer to those used in Figure 5.3.4.

Figure 5.3.11: Picture of the working area in the beginning of the box carrying -experiment.

cases present the different modalities for determining the order. In the first case, the user points to the boxes one by one before the execution starts. Two different pointing methods are used – using a flashlight or a pointer stick. With a pointer stick, the user points to the boxes only after the previous box has been collected. The same pointing methods are used in this case. The boxes are recognized in various ways. Some of the boxes are identified with SIFT features and some of the boxes use color- and code-recognition modalities.

The purpose of this test is to evaluate how the model works when using physical pointing methods without a graphical user interface. In addition, the initial configuration of the pile has some of the boxes overlapped by others, thus causing additional uncertainty in the model. This test evaluates how the model works in this case.

Figure 5.3.11 shows a photo of the scene before the task starts.

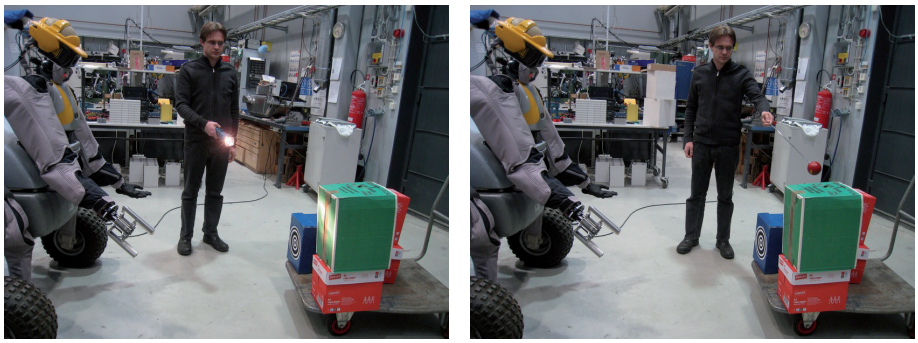
The following description is divided into four subsections. The first subsection describes the pointing methods, how the model is used to detect them, and how the pointing action affects the overall topology of the model. The second subsection describes the first case where the user determines the order of the boxes in the beginning. The second subsection describes the case where the user determines the order as the task proceeds. The fourth subsection compares the approaches.

5.3.2.1 Pointing methods explained

The methods for pointing were illuminating the target object with a flashlight and placing a distinctive-looking object on top of the target object. Figure 5.3.12 shows examples of the two pointing methods. These methods differ from each other conceptually. Pointing with a flashlight changes the appearance of the target object, whereas the pointing stick is a separate object, and the correspondence to the target object needs to be defined with a topological connection.

This case differs from the previous case in that the actual indication is also detected with the same camera as the objects themselves. The pointing means were described in the same way as the box objects themselves.

Detecting the pointer stick The pointer stick was defined as a real object. The object had an "on top of" -type topological connection to the target object. In practice this means that the target



(a) Pointing with a flashlight (b) Pointing with a pointer stick

Figure 5.3.12: The pointing methods used in this experiment.

object is assumed to be under the pointer stick when detected. If no object is detected under the pointer, no target object is associated with the pointer. The location of the pointer is not measured with a laser pointer, but the relative locations in the camera image are used. The appearance of the pointer stick is defined by two parameters: its color and its motion characteristics. The color is in general distinctive, except when compared to the red boxes also present at the scene. The motion characteristics represent knowledge that the pointer is a moving object in an otherwise static scene. The pointer can be identified using color recognition in conjunction with background subtraction. The actual indication occurs when the pointer is held relatively static for one second.

Detecting the flashlight Although the beam of light is not a physical object, it was possible to define it as a real object. The characteristic appearance of the object was the change that was detected with background subtraction. The object had an "on the side of" type of topological connection to the target object. When a change in illumination was detected, the target object was the one that was "behind" the detected light object. In practice, the light object was the side of the actual box, and the target object was therefore the box itself.

5.3.2.2 Recognition of the boxes

The boxes were well-defined objects whose appearance was known in great detail. The walls of the boxes had been photographed; therefore the appearance was described using the images on the sides of the boxes. SIFT features and colors were extracted from the images. In addition, a circle code was attached to one of the boxes. For each box, a real object was created. In addition to the pure appearance description, the size and some additional metadata related to the contents were associated with the boxes. This did not affect the recognition itself but it instructed the robot how it should handle the boxes and what it should do with them. This issue is discussed more deeply in Subsection 5.3.2.3 below.

Creation of observed objects The observed objects were created using recognition-based segmentation. The recognition of the boxes was done combining the SIFT feature matches and the color-based segmentation. When matching the SIFT features in the image to the features in the box objects, the projective transform between the reference image and the camera image was

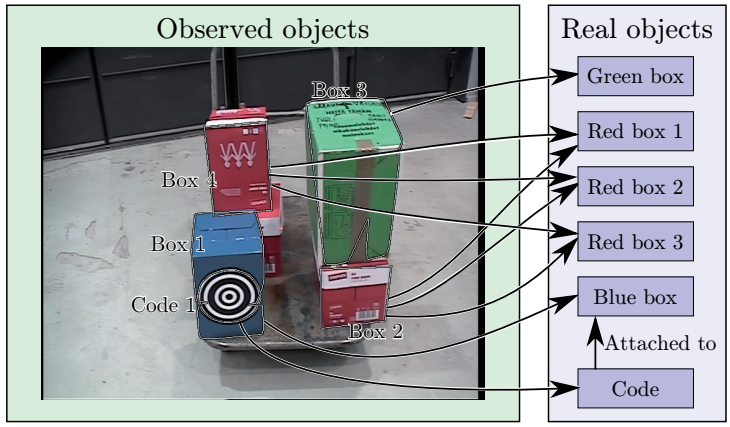


Figure 5.3.13: View of the pile of the boxes. The measured locations of the boxes and the probabilities of the matches are shown in Table 5.3.2.

	Location of the object in meters			Match probability to real objects					
Object	X	Y	Z	Green box	Red box 1	Red box 2	Red box 3	Blue box	Code
Box1	1.442	0.372	0.138	0.0	0.0	0.0	0.0	0.8	0.0
Box2	1.455	0.432	-0.291	0.0	0.3	0.3	0.3	0.0	0.0
Box3	1.454	0.101	-0.273	0.8	0.0	0.0	0.0	0.0	0.0
Box4	1.831	0.149	0.100	0.0	0.3	0.3	0.3	0.0	0.0
Code1	1.442	0.372	0.138	0.0	0.0	0.0	0.0	0.0	1.0

Table 5.3.2: Measured locations of the observed objects and match probabilities between the observed and real objects shown in Figure 5.3.13.

determined. In addition to the appearances of the sides of the boxes, the circle code detection algorithm was used to detect the code. The locations of the objects were not limited to a specific area, and they could have appeared anywhere in the image. The 3D locations of the boxes were not yet determined in this phase.

Matching to real objects As the observed objects were already created on the basis of recognition, the corresponding real objects were found using the existing information on the matches. The match probabilities were calculated on the basis of the match score of the SIFT and color matches. Figure 5.3.13 shows the initial view, the observed objects extracted from it, and the corresponding real objects. As can be seen from the figure, the red box under the other red box was not recognized at all in the initial configuration. In addition, the two visible red boxes were ambiguously matched to three real objects; and therefore, from both of observed objects, there are links to both of the corresponding real objects. This introduces a many-to-many relationship between the red observed objects and the red real objects. The match probabilities are shown in Table 5.3.2.

Localization of the objects The locations of the boxes were again determined using the laser pointer of the robot. Because the boxes are relatively large objects, it is not enough to assume

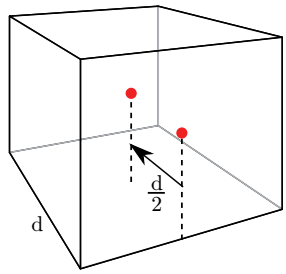


Figure 5.3.14: Projection of the measured 3D point inside the object whose dimensions are known.

that the measured 3D point describes the location of the box completely. Therefore, the geometry of the box is used in conjunction with the measured location. The goal, however, is not to recover the complete location and orientation of the box, but just to gather enough information for lifting the box. The location of the measured point is projected to the middle of the box as depicted in Figure 5.3.14. The location information is passed to the real objects, except in the case where there are links to multiple real objects. This information can be passed as soon as the ambiguity has been solved. The locations of the observed objects are also shown in Table 5.3.2.

Eliminating multiple matches In this case, there are no visual features that could distinguish the two boxes from each other. It is not possible for the model to determine the exact identities of the boxes. If they are equivalent, having similar contents, it is possible to randomly assign the unique links to the objects. If it is not possible to handle the two similar looking boxes as equivalent, then the user needs to inform the system about which observed object corresponds to which real object.

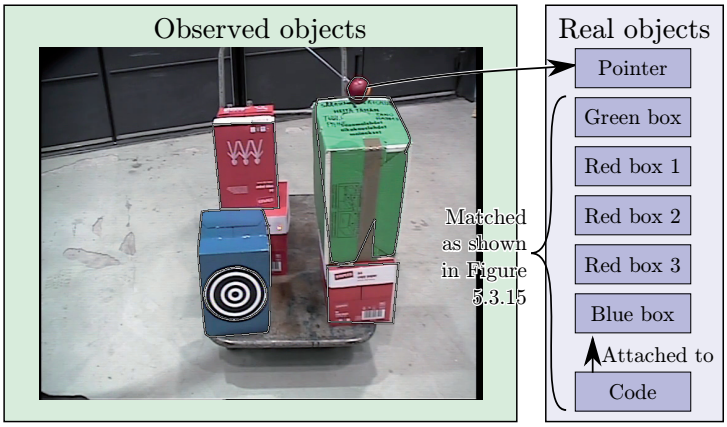
5.3.2.3 Functional features

In addition to the appearance description, some additional data was associated with the boxes. This data described the contents and functional features of the boxes, but not their appearance. With this information, the robot was able to know how to handle the boxes. This data could include information on the weight of the box, whether the contents was fragile, whether the contents fall or spill when accelerating too fast, and what kind of action needs to be performed on the box. This information can be provided to the robot in the task definition, or it could be later provided through additional codes attached to the boxes. In this case, it was assumed that the user provided all the necessary information in the task description.

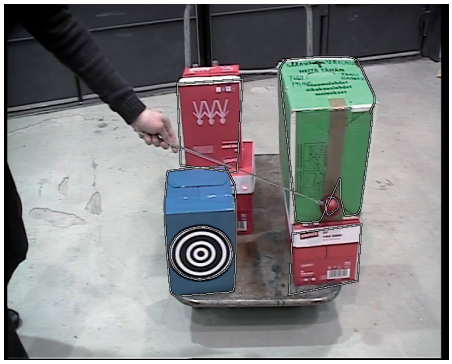
5.3.2.4 Indicating the order in the beginning

In the first case, the user showed the robot the complete picking order of the boxes before the robot started to perform the task. The user indicated the boxes one by one in the correct order. An ordered list of target objects was formed on the basis of the pointing. The two alternative pointing methods are discussed below.

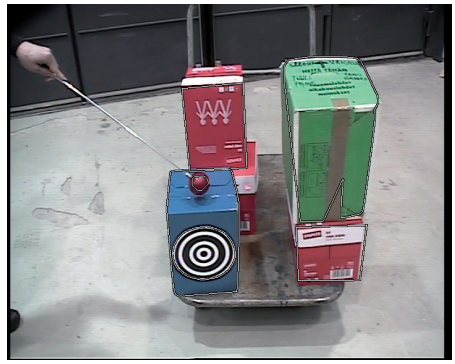
Pointing with the pointer stick The boxes are indicated in the following order: green, blue, the red box under the green box, the topmost red box behind the blue box, and the red box under the other red box. The pointer could be found in every case, even though the color of the red boxes



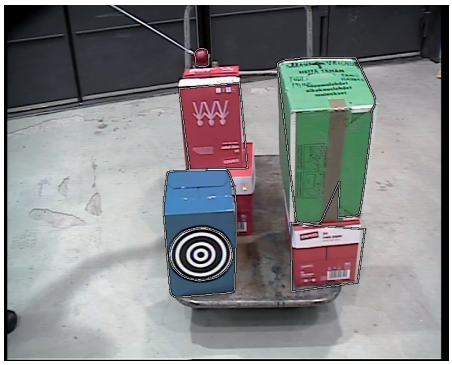
(a) Pointing the green box



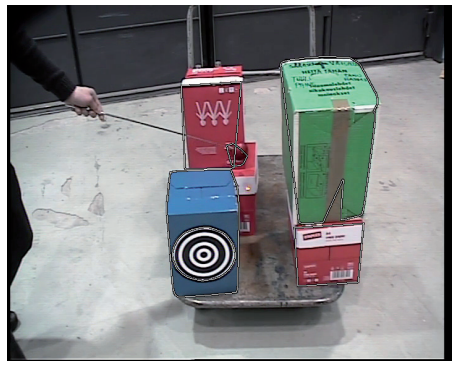
(b) Pointing one of the red boxes



(c) Pointing the blue box



(d) Pointing another red box.

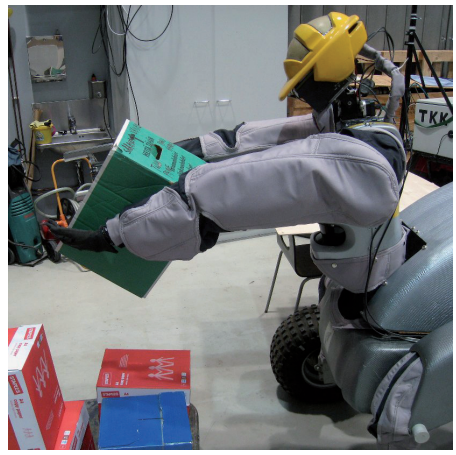


(e) Pointing the red box behind the blue box.

Figure 5.3.15: User pointing different boxes using the pointer stick.

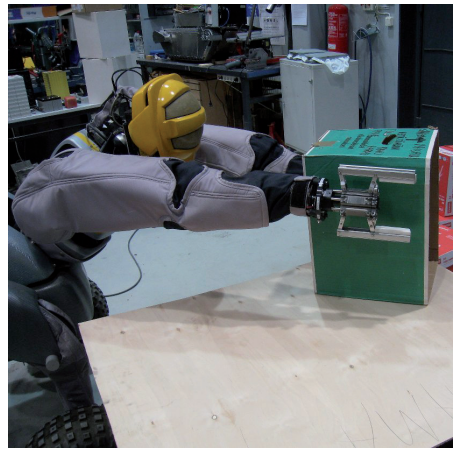
was close to the color of the pointer. Figure 5.3.15 shows the extracted observed objects when pointing to the boxes. The results were good for the first four boxes because the target object could be uniquely determined. When the pointer was in front of an object, only the object under the pointer was considered. The fifth box was more problematic because it was occluded behind

(a) Reaching the box and grabbing.



(b) Lifting the box up.

(c) After backing up, placing the box on the platform.



(d) Another view of the platform.

Figure 5.3.16: WorkPartner lifting one of the boxes from the pile to the platform.

the other box. Therefore it was not clear which box was pointed to when the user held the pointer stick as shown in Figure 5.3.15e. This pointed to the blue box, but because it had already been pointed to, this action was ignored. Because the model did not include the red box on the bottom, the task description only included the four first boxes at the start.

The actual lifting procedure was performed in the following way. First, the robot approached the box to a distance where the box was reachable by the manipulator. When the robot was at a close distance, the location of the box was measured again. Then, if needed, the manipulator was tilted down for a better reach. After that, the box was grabbed between the hands, and the manipulator returned to the original position. Then the robot backed up a little and reached the manipulator far to the side to lay the box down onto a platform. Figure 5.3.16 shows the phases of lifting a box to the platform.

The lifting of the first three boxes was straightforward. All the boxes were initially visible and

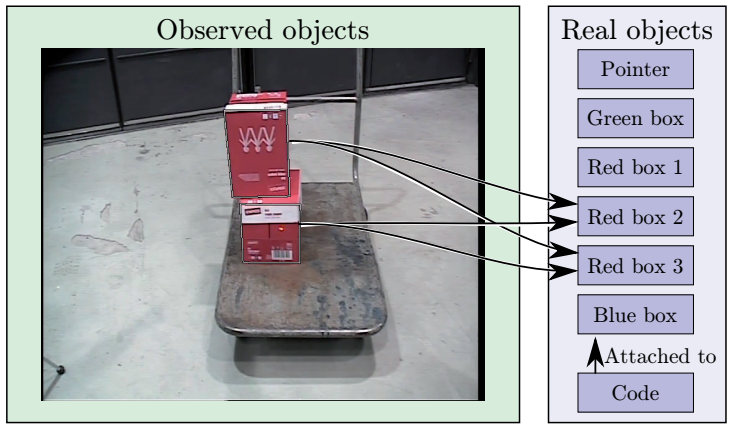


Figure 5.3.17: Bottom-most red box revealed behind the blue box.

identified. After the blue box was removed, the bottom-most red box was revealed (as shown in Figure 5.3.17) and then recognized. Because the robot did not have information on what to do with the red box at this point, it had to ask the user for assistance. Basically, the user needs to show the new picking order of the remaining boxes to the robot. Therefore the robot discards the order information that it has, and the user has to indicate the picking order of the two boxes at this point. The order can now be unambiguously indicated, and the robot is able to determine the correct picking order and deal correctly with the remaining boxes.

Pointing with the flashlight In the alternative scenario, the order was indicated with a flashlight. The order was the same as in the previous case. The wide beam of the flashlight sometimes illuminated an area larger than the one object. However, this did not cause problems because only the center of the illuminated area was compared to the locations of the box objects, and the intersecting object was chosen. Figure 5.3.18 shows the indicating of the order with a flashlight.

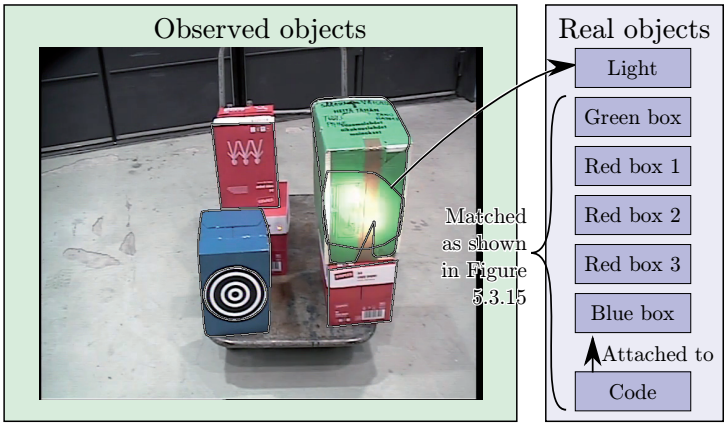
The first four objects were also successfully indicated in this case, but the fifth object remained unseen. The illumination of the fifth object is shown in Figure 5.3.18e. The target of this procedure was unknown because it did not match any of the known real objects.

The task was executed as described in the previous case. The boxes were picked up one by one until the lower red box was revealed. Again, the user was asked for assistance. The user then indicated the remaining boxes in the desired order, and the robot was able to carry on the task.

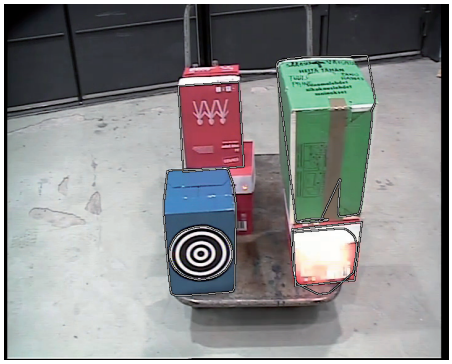
5.3.2.5 Indicating the order as the task proceeds

In this case, the human does not indicate the picking order of the boxes in the beginning, but just determines the first box. The task description defines that this is enough for starting the procedure; therefore the robot immediately starts to move. When the robot has removed the first box, the user indicates the second one. This is continued until the task is finished. Figure 5.3.19 shows pointing the boxes with the pointer stick one by one only after the previous one has been removed. Figure 5.3.20 shows the same procedure, but now using a flashlight for indicating the picking order.

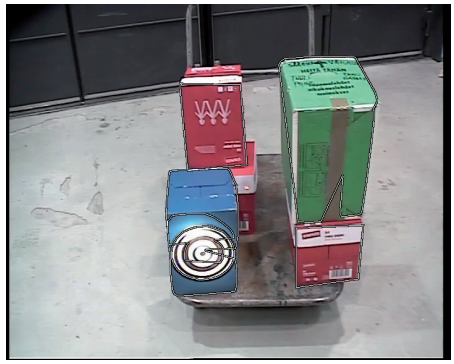
The ambiguity that was present in the previous case is now avoided. The indicated target box



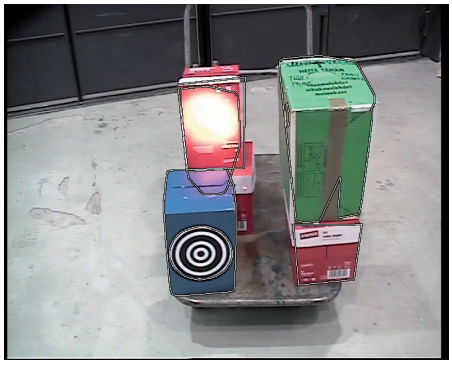
(a) Pointing at the green box.



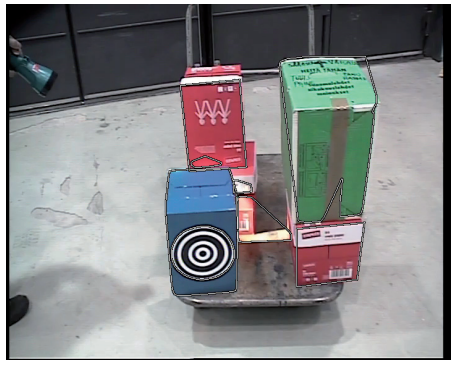
(b) Pointing at the first red box.



(c) Pointing at the blue box.

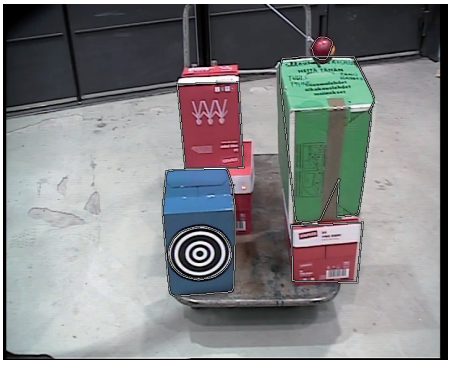


(d) Pointing at the second red box.

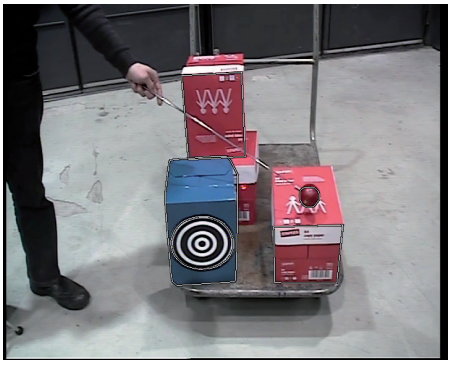


(e) Pointing at third red box.

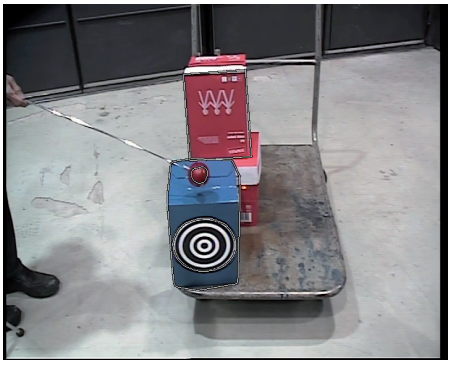
Figure 5.3.18: Indicating the objects with a flashlight.



(a) Pointing the green box.



(b) Pointing the first red box.



(c) Pointing the blue box.

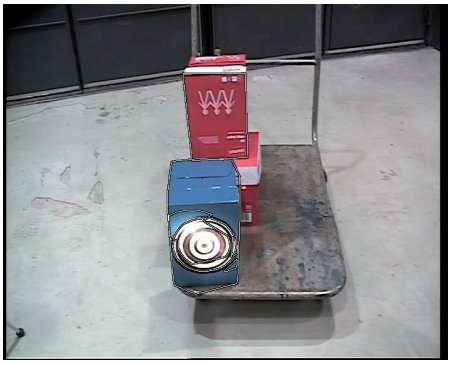
(d) Pointing the second red box.

(e) Pointing the third red box.

Figure 5.3.19: Pointing the order of the objects during the execution. The figures show the boundaries of the observed objects including the pointer itself.

(a) Pointing the green box with the flashlight.

(b) Pointing the first red box.



(c) Pointing the blue box.

(d) Pointing the second red box.

(e) Pointing the third red box.

Figure 5.3.20: Pointing the order of the objects with the flashlight during the execution. The figures show the boundaries of the observed objects including the beam of the flashlight.

was always completely visible. Therefore the robot was able to correctly recognize which object was indicated with the pointer stick or the flashlight.

5.3.2.6 Comparison of the approaches

The two methods were similar in how they were handled in the cognition model. The only difference was how the model was actually utilized in the task execution.

The first method, showing the whole order in the beginning, introduced problems because the partial occlusion of one box in the initial configuration. Therefore the robot needed to ask for assistance during the task execution. However, the user needed to assist the robot only once during the task. The main potential problem arising from this approach is that because the human does not necessarily realize that the robot does not see all the boxes correctly, this may lead to incorrect interpretations by the robot, and the task may be executed in a wrong way.

The second method, showing the order during the task execution, required more involvement from the user, but the result was more reliable. The user was better able to realize what the robot actually saw, and the robot was able to get correct information.

The pointing methods were very similar in how they were used. In practice, using a flashlight may be more challenging in bright light because the illumination used for indicating should overcome the ambient light. In bright sunlight, this may be impossible without an extremely bright light, which can be dangerous to the eyes. The pointer stick, on the other hand, may be difficult to detect if there are similar shapes or colors in the target objects. In both cases, the described motion detection mechanism only works if the scene is completely static. If there are other moving objects, or if the target objects themselves are moving, other methods need to be used.

5.4 Conclusions from the experiments

The experiments evaluated several aspects of the shared cognition concept. The first experiment presented utilization of human cognition in the segmentation of the sensor data. The second experiment presented a simple example that demonstrated building the model. The third experiment evaluated autonomously and interactively recognizing objects whose exact appearance and number were not known before starting the task. Finally, the fourth experiment presented a case where a relatively complex task was executed relying on the cognition model, and where the human pointed to the target objects using a pointing method without a graphical user interface.

In the first experiment, the segmentation done by the human allowed the robot to understand what the boundaries of the sand pile really were. A sand pile is hard to define formally, because it can look virtually like anything, therefore human cognition is valuable in this task. After the human has marked the corresponding area in the robot's sensor data, the robot is able to use this information in its task execution. The robot needs to identify landmarks in the sensor data so that it knows which part of the sand pile corresponds to which part in the sensor data, and therefore it knows where the marked boundaries are in the real world. It is often not enough to rely just on the navigation capabilities of the robot because the accuracy may not be good enough to know the exact location of the marked boundary after the robot has moved.

In the second experiment, the human described the target object in a way that allowed the robot to recognize it when it was seen. In the test case, the object did not uniquely define the object, but there were several alternatives for the object. The model was able to handle the ambiguous

situation by retaining several hypotheses for the recognition. Different viewpoints affected the reliability of the recognition because of changing perspective. Finally, the topological connection between the code object and the actual box object allowed the robot to reliably identify which of the objects was the target one. In this case, the robot was moving around the working area. The experiment only evaluated how the cognition model worked; it did not evaluate how the higher level reasoning (that is, perception planning) was done. The robot was able to keep the state of the model up-to-date during the execution.

The third experiment presented a case with a real robotic manipulation task, where the robot collected pieces of litter from the ground. The objects to be collected were defined with vague definitions, and therefore human verification was required. The human assistance was performed either in the beginning or only as problems occurred. The robot was able to cope with a major part of the case even without human assistance, but in some cases the robot needed to waste a lot of time when trying to collect pieces that were not really interesting objects and were just false matches. Basically, it was beneficial to preprocess the data automatically and to use human cognition as much as possible in the classification task.

In the fourth experiment, the user interacted with the robot using concrete real-world pointing methods instead of a graphical user interface for controlling the robot. The pointing methods were presented using the modalities of the proposed shared cognition concept. In this way, the pointing method and the perception of the actual task were modeled with the shared cognition model. Recognition of the pointing methods worked relatively well, and the topological connections provided a good way to define the target for the pointing. In practice, this modality requires an advanced higher-level task, because it needs to figure out when the user is actually pointing to the objects and when the actions in the scene are something else.

An actual real-time implementation was not used in any of the experiments. The manually controlled actions were a realistic simulation of the robot's actual actions. Implementation of the tasks would have required a complete task execution framework in addition to the cognition model. This was beyond the focus of this thesis, and therefore the manual approach was chosen. In addition, the manual step-by-step execution enabled full control of the task and repeating the algorithms to the recorded sensor data later on.

Chapter 6

Discussion

The proposed shared cognition concept has been tested with various experiments. These have covered most of the features described in the earlier chapters. However, the proposed shared cognition concept also contains some features that were not evaluated through practical experiments. This chapter goes through the model and analyzes the applicability of its different aspects, either on the basis of the actual experiments or a discussion.

6.1 Structure of the proposed model

This section discusses the overall structure of the proposed shared cognition approach as described in Chapter 3.

6.1.1 Abstraction of robot's perception and cognition

The proposed concept divides the way the robot understands its environment into three separate concepts: perception, cognition, and knowledge. The perception of the robot is represented with observed objects, its cognition with real objects, and its knowledge with meta-objects.

The division formalizes the whole process of building a cognition model. This way the appearances and identities of the objects can be described in a unified way that does not heavily depend on the robot platform used. The same modalities can be used for autonomous object recognition by the robot and for shared cognition.

As an alternative to the approach with separate observed and real objects, the usual way of recognizing objects is to match the sensor data directly to the objects representing the actual physical entities. In practice, this would mean leaving out the whole concept of observed objects and using the real objects for dealing with the sensor data directly. One of the reasons for the chosen approach is the probabilistic approach of the model, that is, to allow uncertainty in the recognition. When the observed objects have been created from the sensor data, the system does not yet assume that they necessarily represent some specific real objects. The connection is formed if there is a match between them. In this way, even multiple candidate observations for one real object are allowed.

In practice, this model is idealizing. because it assumes that the sensor date can always be segmented. This is not always possible, and the chosen approach may cause additional (sometimes unnecessary) overhead to object recognition. In addition, the multiple matching observed objects

for one real object causes ambiguity problems that are not necessarily addressed strongly enough in the model. A concrete example was the experiment with the boxes, where multiple matches were formed because the model could not distinguish the red boxes from each other. However, how the information is exchanged between the observed and real objects is a partially unanswered question. In the example case, excess matches were discarded by the task to allow an unambiguous determination of real object's location.

The knowledge level was covered in the litter-picking experiment. A piece of litter was defined as a meta-object that was used for detecting possible pieces on the ground. The meta-object represented the robot's knowledge of a piece of litter. According to the experiment, the meta-objects fit well with the proposed model.

Meta-objects and real objects include functional features that represent all the information related to the physical entities that is not related to the perception process. This can include, for example, the material of the entities, the contents and weight of containers, or instructions on how they should be handled.

6.1.2 Shared cognition

Sharing the cognition between the robot and the human was the primary motivation for developing the proposed model. Abstracting the robot's understanding to different levels allows the user to affect different aspects of building the robot's cognitive understanding. Sensor data segmentation by the user results in observed objects whose segmentation is usually superior when compared to segmentation performed automatically by the robot. Forming the matches between observed and real objects allows the user to perform the actual object recognition on behalf of the robot, which again often leads to a better result. In addition, the user can instruct the robot by inputting location data to the real objects. After the user has performed the segmentation, recognition, or both of these, or has inputted other data, the robot is able to utilize this information afterwards and autonomously proceed with the task execution.

In addition to assisting the robot in the recognition task, the meta-objects and real objects can be used as means of expressing the task-related information to the robot. The user can create real objects or meta-objects that represent the targets of the actual task. In this way, the task itself does not need to worry about how the appearance of the objects is represented, but it is handled by the model.

The experiments cover various modalities for cognition sharing. In the sand pile experiment, the user performs the sensor data segmentation using a graphical user interface. In the first box recognition experiment, the user formally described the appearances of the objects. In the litter-picking experiment, the user again used a graphical user interface for classifying the objects on behalf of the robot. In the box-carrying experiment, the user did not have to use a user interface on a computer but was able to use a physical pointer stick and a flashlight to indicate the target. The graphical user interface was not covered more deeply than just mentioning which tasks it is required for. It is needed for viewing and segmenting the robot's sensor data, creating and editing real and meta-objects, and observing their states.

6.2 Building the cognition model

This section focuses on how the cognition model is actually built. This section refers to the description presented in Chapter 4.

6.2.1 Using the robot's sensors to build the model

In a shared cognition concept, most of the aspects can either be performed automatically by a robot or collaboratively with a human. The automatic approaches are based on algorithms and strict definitions, whereas the collaborative methods utilize the human's understanding. The following subsections cover both of the approaches when applicable.

6.2.1.1 Segmentation

Several automatic segmentation modalities were presented: recognition-based segmentation, depth-based segmentation, edge-based segmentation, motion-based segmentation, and location-based segmentation. In addition to the automatic modalities, the human-assisted segmentation was covered.

Recognition-based segmentation is performed on the basis of the known appearance features of the object. In practice, this segmentation modality does the sensor data segmentation and also the matching of the observed and real objects simultaneously. If this was the only modality, the rationale of the whole concept could be questioned, because the segmentation and recognition would no longer be separate actions. However, this is just one of the several segmentation methods, and the other methods can be used separately from the recognition methods. The recognition-based segmentation was extensively used in the experiments because it is the most applicable for most situations. According to the tests, it works relatively well, but its problem is that it does not necessarily describe the boundaries of the segmented area accurately. This is caused by the fact that, for example in SIFT-based recognition, the features are usually inside the object's boundaries, not on the edge, and therefore it is not known where the edge actually is.

Depth-based segmentation is based on discontinuities in the depth data. This usually requires accurate depth information from stereo cameras, from a laser scanner, or a similar sensor producing depth information. This kind of segmentation was not evaluated in the experiments; therefore its applicability was not extensively evaluated in practice. However, the principle of the segmentation is simple, and it is known to work relatively well in various applications. There are two kinds of problems in depth-based segmentation. If the physical entities are very close to each other, they may look like a single object. On the other hand, entities whose physical form is not compact may look like several separate objects.

Edge-based segmentation relies on an assumption that a visible edge separates the objects from the rest of the sensor data, particularly camera image data. This segmentation method was evaluated in the litter-picking experiment. The method was working very well in the case where false matches were accepted. The relatively large number of false matches was caused by other edges that were not part of the actual discrete physical entities. In practice, this segmentation method can never be very reliable, and human assistance is usually required.

Motion-based segmentation assumes that the physical entity in question can move. This method was used in the box-carrying experiment to recognize the pointer stick and the flashlight beam. In an otherwise static scene, this was a very effective segmentation method. However, if there were several moving entities, the method would not have been as effective because all the moving

objects would have been detected. In addition, if the robot moves, then the whole world seems to be moving. This can be overcome by compensating for the robot's own movement.

Location-based segmentation is used in cases where the location of the physical entity in question is known to be in a certain area. If the robot is able to accurately relate the sensor data to global coordinates, this method may be very effective. This method was used in the sand pile recognition experiment. The boundaries of the sand pile were defined as a bounding box in world coordinates, and the robot was able to perform the segmentation on the basis of this information.

Human-assisted segmentation is a means of shared cognition. It utilizes the human's ability to interpret the sensor data. It may often be difficult for any algorithm to reliably determine the boundaries of an object, but it can be more easily performed by the human. This was evaluated in the sand pile recognition experiment. It was difficult to formally define how the segmentation would have been performed algorithmically, but by utilizing the human's understanding, the segmentation was performed easily. This method can also be used in various other domains where it is difficult to automatically perform the segmentation. After the human-assisted segmentation, it is usually necessary to use some other means for segmenting the data as the task continues. In the sand pile experiment, this was done on the basis of the location of the pile. The shared cognition between the human and the robot is further discussed below in Subsection 6.2.2.

6.2.1.2 Extracting information

After segmentation, the features are extracted from the sensor data. Appearance features describe how the object looks and structural features describe its shape.

These two can often be used interchangeably. The shape of the object may be used for recognition, for example, when recognizing an object with sharp visual features that directly relate to its shape. On the other hand, visual appearance can sometimes include essential information about the mechanical state of the object. An example of such a case is a pot of coffee made of glass. The level of the liquid can be determined visually, while laser scanners may only be able to measure the shape of the pot.

Two types of appearance features were covered: sparse local image features and features extracted from the entire region of an object. Using sparse local image features, such as SIFT and SURF features, makes it possible to recognize an object without complete segmentation. It is enough to see only part of the object, which makes these types of features robust against occlusion. However, these kinds of feature descriptors usually require clearly distinctive landmarks, and they may not be the best alternative for large objects with few distinctive features. Features extracted from the entire object region enable color-based recognition modalities. The color histogram of the segmented area can be compared to a reference histogram, and the differences between the histograms can be used as a metric for recognition.

Learning the appearance can be based on either of the mentioned feature types. In the litter-picking experiment, the color features extracted from the entire area were used. However, this kind of approach requires that the segmentation has been successfully performed. This requires that the segmentation is based on some reliable measure. In the experiment, the potential objects were known to be distinguishable by the edge between them and the background. If the background had more than one color, this would not have been the case, and the segmentation should have been based on segmenting 3D data acquired by sweeping the laser scanner.

Structural features describe the shape and spatial state of a physical entity. These features are

typically used to measure the properties needed for manipulative tasks of the robot. In the case of a sand pile, the robot can plan its actions on the basis of the state of the pile. It can plan the trajectory to approach the pile from the best possible direction to allow efficient scooping of the sand. The state of a sand pile is effectively represented as an elevation map.

Sometimes the structure of the physical entity is not significant. In the box recognition experiment, some of the boxes changed their shape because the user opened and closed the boxes. The object recognition was based only on the color and height of the objects. The shape affected the height, but the actual orientations of the box covers were not measured.

6.2.1.3 Recognition

Two main types of object recognition were described: object class recognition and object identification. In object class recognition, the identities of the objects are not known beforehand, but objects are recognized based on the descriptions of typical objects of a certain class. These descriptions are stored in meta-objects. Object identification, on the other hand, utilizes known identities of the objects to recognize them. The objects are considered as unique entities that can be identified on the basis of their appearance and structural state.

The object class recognition was evaluated in the litter-picking experiment. A typical piece of litter was described using meta-objects. The description mentioned a visible edge between the background and the object itself. In practice, every part of the image surrounded with visible edges was considered as possible pieces of litter. This kind of description is not very unifying, and a number of false matches were found even in a limited area. The description contained in a meta-object can be more unifying. For example, the appearance of a typical human could include a description of a typical shape, skin color, and motion characteristics.

Object identification was evaluated in all four experiments. Even in the litter-picking experiment, the objects that were first recognized using object class recognition were later identified using object identification. The difference between object class recognition and object identification is mainly conceptual. The same or at least similar algorithms can often be used for both types of object recognition. However, object identification can usually be made more reliable if the appearance of a specific object is clearly distinctive. In addition to the possibility to describe the appearance in a more unified manner, object identification can also utilize the existing information on the previously measured location of the object. If there are several similar looking objects, it may still be possible to identify one object among these if the location information is utilized. However, if the locations change while the robot is not observing them, it may not be possible to identify a certain object among other similar looking objects again.

6.2.1.4 Handling the uncertainty

The probabilistic approach of the shared cognition concept enables having multiple possible matches to the object with variable probabilities for the matches. This way the robot could cope with situations where it was not possible to uniquely identify an object on the basis of the sensor data, but several options remained. However, usually when the robot needs to make decisions about which object to manipulate, the number of candidates should be reduced to one. If the selection cannot be based on the probabilities, user assistance may be required.

The main challenge of the probabilistic approach is determining realistic probabilities. This problem was encountered in various parts of the experiments. The probabilities used were some-

times only based on approximations rather than on a deep statistical analysis of the situation. This kind of analysis is often impossible.

The probability of the match does not necessarily describe the actual probability per se. Its purpose is to inform the robot about the reliability of the recognition. The higher level tasks can utilize this information in planning and executing the movements. If there is no information on reliability, the match probability should be 1.0 in those cases.

6.2.1.5 Representing the location and orientation

The location of an object can be represented either with spatial coordinates or using a topological representation.

Spatial coordinates use metric representation. The coordinate origin can be tied to a global origin, or a local origin can be used. The choice depends on the application. Workspace origins were used in each of the experiments. This is usually the most applicable choice for practical tasks. It is usually not necessary to know the exact location of the robot with relation to a global GPS origin because the typical tasks of a service robot are usually executed in a constrained area. However, there may be cases where a more global presentation is required. These include vehicle-type robots that may as well utilize the proposed model as a representation of the environment.

The exact location of an object is often not required. It may be enough to just know an approximate location of the object and perhaps measure the exact location just before starting the manipulation task. On the other hand, in some tasks, exact positioning of the objects is necessary. If an object is described using only its location as a unifying feature, the location needs to be somewhat accurate; otherwise, the object definition is too vague to be practical. In the sand pile recognition task, the segmentation of the pile was based on its location.

A topological representation of the object location can be used in cases where it is not necessary to know the exact metric location, but some kind of relative location information is still applicable. An arrow pointing to a certain direction or an object attached to another one are examples of such cases. The location may depend on the location of another object. This kind of location information is solely based on information inputted by a user. It requires very advanced artificial intelligence for a robot to understand the topological relationships of separate objects if no constraints are known. Therefore, in this research, it is assumed that the user always defines the topological locations of the objects.

Uncertainty about the location is introduced when the measurement is not accurate. This occurs when measuring the location of an object using only a monocular camera. This measurement results in an elongated probability distribution because the distance cannot be measured from a single measurement. It is possible to perform an accurate location measurement on objects if motion is present. This technique, known as 'structure-from-motion', was not part of this research.

Uncertainty about location is useful when the robot needs to plan its actions to ensure that it has the most accurate data possible. If the location contains large uncertainty, a different viewpoint may provide more accurate information. The covariance matrix representing the uncertainty can thus be utilized on a higher level task to plan the movements.

6.2.1.6 Dealing with moving targets

All the experiments in this research were done with static objects. The only moving target was a human that changed the setting in the box recognition experiment. Then again, the human was

not recognized because the human-type object was not defined as required information in the task execution, and a human detection algorithm was not run.

With moving targets, the timestamps play an important role. If an object is seen in some location, there is no certainty that the object will be there after some time. Therefore it is important to store the timestamps of moving objects. Comparing the timestamp of the measured location to current time, it is possible to evaluate whether it is probable that the object is still there or not. The probability distribution of the location becomes larger when time passes. This indicates that the object may be anywhere within the uncertainty region.

6.2.2 Sharing the cognition

This section discusses the techniques used to utilize the shared cognition in human-robot collaboration. This section refers to techniques presented in Section 4.2.

6.2.2.1 Referring to robot's sensor data

A key technique in shared cognition is marking objects in a robot's sensor data. This utilizes the accuracy of the robot's sensor information and the cognitive capabilities of a human. This technique was evaluated in the sand pile recognition experiment. It would have been very difficult for a robot to reliably recognize the pile, but this task was much easier for a human. After marking the sand pile, the robot was able to use this information later for segmentation of the pile.

In addition to using a graphical user interface in marking the sensor data, different kinds of pointing techniques can be used. The target objects can be indicated with a pointer stick or by illuminating them, as in the box lifting experiment. With these techniques the user does not need to use a computer for communicating with the robot, but the objects can be pointed to in the real physical world. In addition to hand-held pointers, static signs can be used. An area can be marked by surrounding it with signs that denote the boundaries of the area. The robot can then use this information to limit the task execution to the specific area.

Using the robot's sensor data for teaching the appearance of an object is very effective. The same sensors and location information that is used for teaching the properties of the object are also used for recognizing it. In addition to the appearance, the robot already knows the location of the object in its own coordinate system. It is then easier to track the object. This provides much greater reliability than when the same information is inputted externally.

6.2.2.2 Describing the objects

It is also possible to describe the objects using numeric and verbal descriptions of the objects. This can be a simple description such as the hue of the color or the size, or it can be a more complex description such as a collection of SIFT features extracted from a photograph. A target object can be described through real objects or meta-objects. Both of the experiments involving boxes used description using real objects. They described specific objects and their appearance in different ways. The litter-picking experiment, as mentioned earlier, used describing the objects using meta-objects.

This kind of object description is a more traditional approach to object recognition than referring to a robot's own sensor data. The sensors used for teaching are different, and often the teaching is not actually based on any sensor data but just on an approximate description from the user. In addition, the description does not necessarily involve location information, therefore the

robot needs to first find the object and possibly confirm from the user whether it has found the right object.

6.2.2.3 Getting the information from the model

The information can also be transferred in the other direction, from the robot to the human. This is useful when the robot's accurate sensing ability is used to acquire information about the environment. The robot may, for example, perform a survey task and create a map of an area and then show the map to the user. The graphical user interface can generally be used for this type of information exchange.

Another way for the robot to provide information is to use its own mobility to indicate objects. For example, if a user interface was not available in the litter-picking task, the robot could have been indicated the potential pieces of litter using its laser pointer. The user would have seen the bright red dot and understood which physical entity the robot was referring to when asking whether it is a correctly recognized piece of litter or a false match.

Many features of the cognition model can also be summarized. For example, the robot could just use its speech synthesizer to inform the human about how many objects it has found. The user can then draw conclusions about whether the robot has found a specific object or has found multiple candidates and would require assistance.

6.3 Using the model for task execution

Ultimately, the utility of the shared cognition model depends on its applicability to real tasks. Many real task scenarios have already been discussed in this chapter. All of the experiments introduced some kind of robotic movement, and the two latter experiments introduced real manipulative tasks. The shared cognition model was successfully used as part of the execution of these tasks.

There are various applications to the proposed concept in real tasks. The simplest example is path planning. An occupancy grid can be populated on the basis of the state of the cognition model. The obstacles are placed in the corresponding locations in the grid, and the path planning is then performed.

More challenging tasks involve the manipulation of objects. These tasks often require exact spatial information about the objects when performing the manipulation, but coarser information is enough when the robot is moving from one place to another. Manipulation of objects may also require information about how the objects should be handled. If a box is full of glass vases, it should be carried with more care than a bag filled with wool. The requirement of extra care can be taken into account when moving the robot. In special cases, the robot is moved more slowly to avoid sudden movements.

The experiments presented in this thesis did not involve real-time computation and manipulation. A real-time case has various constraints that were not considered in this research. Probably the most important constraint is the computation capacity. The algorithms used in perception are generally very resource intensive and require powerful computers. Still, it is usually only feasible to concentrate on the most important algorithms and not run all the possible algorithms at once. In a way, this was demonstrated in the box recognition task. The human entered the working area, but the robot did not recognize him because it was not executing a human detection algorithm. In this task it was acceptable not to detect a human but it was enough to detect the consequences

of the actions of the human, that is, the changed configuration of the boxes.

One possible approach to provide rich environment information while still retaining the real-time performance is to run the algorithms at different priorities. For some algorithms it is completely fine that they are run only every few seconds, while some algorithms need to be run several times a second. For example, an algorithm that observes the state of a sand pile does not run constantly because it is not likely that the state of the pile would suddenly change. On the other hand, a localization algorithm should usually be run at a very high speed to ensure controllability of a moving platform.

The experiments did not evaluate the accuracy requirements of the manipulation tasks. Especially tasks involving movement and manipulation may introduce large errors in positioning of the target object and the robot itself. Therefore, practical manipulation tasks of service robots usually require visual servoing techniques that adjust the robot's manipulator according to the error measured between the desired target location and the actual location of the manipulator.

Chapter 7

Conclusions and future work

This work has presented a novel concept for building a cognition model of the environment. As the concept enables utilizing cognition along with the sensory data of a robot, it is called a shared cognition concept. The concept does not limit the use of the model to cases involving human-robot collaboration; it can also be used in autonomous operations of the robot. As the concept allows various representations for recognizing objects and building geometric models, it is a versatile representation of robot perception. In addition to being a tool for perception, it is also a tool for interaction. Many types of human-robot interactions can be easily handled with the proposed concept.

The three-level abstraction of a robot's mind is inspired by human reasoning. However, the proposed model does not try to imitate a human. Therefore human psychology was not studied when creating the robot cognition model. Different levels may bring additional overhead to the system, but at the same time it makes the whole problem of robot cognition more concrete and easier to handle.

The results from the experiments show that the proposed concept works very well. Object recognition seems to work as expected, and the user input in segmentation, recognition, and pointing provided additional information. Using both topological and metric locations interchangeably made the approach more versatile. However, the lack of real-time implementation probably caused some potential problems to remain undetected.

As the shared cognition concept is a holistic approach to robot environment understanding, the whole perception of the robot could have been modeled with the shared cognition model. In practice, the recognition process of the shared cognition model is not strictly defined and limited, but rather depends on custom algorithms. A strict formalization of the recognition process would make it possible to define the appearances of the objects without writing a single line of code. However, because a "silver bullet" solving the whole object recognition problem is still missing, it is not possible to rely on one method for all possible recognition cases. But it is necessary to allow the model to use custom algorithms that suit every application.

7.1 Future work

The components of the model were tested separately, but the whole shared cognition concept has not yet been implemented in real-time software. Research in the field continues at the Generic Intelligent Machines research center. In the future, the model will be implemented in a way that

enables it to be used in different types of robots equipped with different sensors. Implementing the proposed shared cognition concept as a complete real-time system would allow the proposed model to be used as the basis of a new approach to human-robot interaction.

The research did not focus on the real-time aspects of the model. Object recognition algorithms are often heavy, and it is not always possible to run multiple algorithms continuously. The model needs to be able to determine which algorithms are needed and which features need to be matched. This requires the recognition algorithms to be linked to higher-level tasks to allow the exchange of information on resource allocation.

This research studied the exchange of information between one human and one robot. Extending the cognition model to enable more parties to participate would offer interesting opportunities. Multiple robots sharing cognitive information would be able to use the cognitive information for task planning and execution. However, this leads to many questions requiring extensive research. Presumably, the biggest question is: what information is transferred between the agents and what information is stored only locally? In addition, an inevitable ambiguity arises when multiple agents interpret the real world with their sensors. If two parties disagree on the recognition result, it may be difficult to determine whose opinion is correct.

The service robots on the market today are mainly lawnmowers, vacuum cleaners, and similar robots that are able to work with a limited knowledge of their environments. Developing more sophisticated service robots requires a better understanding of the environment. The shared cognition concept presented in this research answers many questions related to understanding the environment and sharing this information between a robot and human. In addition to offering solutions to collaborative robot task execution, the model offers tools for further research aiming towards truly cognitive robots.

Bibliography

Kazuhiko Akachi, Kenji Kaneko, Noriyuki Kanehira, Shigehiko Ota, Go Miyamori, Masaru Hirata, Shuuji Kajita, and Fumio Kanehiro. Development of humanoid robot HRP-3P. pages 50–55, 2005.

P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310–310, January 1989.

Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, chapter 32, pages 404–417. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006.

Jean-Yves Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. 1999.

Jean-Yves Bouguet. Camera calibration toolbox for matlab, 2008.

John F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, November 1986.

Chin S. Chua and Ray Jarvis. Point signatures: A new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85, October 1997.

John J. Craig. *Introduction to Robotics: Mechanics and Control (3rd Edition)*. Prentice Hall, 3 edition, August 2004.

Encyclopædia Britannica. Cognition. `http://search.eb.com/eb/article-9024661`, Referenced 26.8.2010, 2010.

Fadi Fayad and Veronique Cherfaoui. Tracking objects using a laser scanner in driving situation based on modeling target shape. In *2007 IEEE Intelligent Vehicles Symposium*, pages 44–49. IEEE, June 2007.

Pekka Forsman. Kinematic chain of WorkPartner's laser pointer measurement. December 2005.

Pascal Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. *Machine Vision and Applications*, 6(1):35–49–49, December 1993.

Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22–22, July 2000.

Cipriano Galindo, Juan-Antonio Fernández-Madrigal, Javier González, and Alessandro Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, November 2008.

Iryna Gordon and David Lowe. What and where: 3D object recognition with accurate pose. In Jean Ponce, Martial Hebert, Cordelia Schmid, and Andrew Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, chapter 4, pages 67–82. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006.

Aarne Halme, Ilkka Leppanen, Jussi Suomela, Sami Ylonen, and Ilkka Kettunen. WorkPartner: Interactive Human-Like service robot for outdoor applications. *The International Journal of Robotics Research*, 22(7-8):627–640, July 2003.

Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, February 1968.

Richard Hartley and Peter Sturm. Triangulation. In VÃ¡clav HlavÃ¡c and Radim Å Ã¡ra, editors, *Computer Analysis of Images and Patterns*, volume 970 of *Lecture Notes in Computer Science*, pages 190–197–197. Springer Berlin / Heidelberg, 1995.

Mikko Heikkilä. *Configuration of Skilled Tasks for Execution in Multipurpose and Collaborative Service Robots*. PhD thesis, HELSINKI UNIVERSITY OF TECHNOLOGY, 2009.

Mikko Heikkilä, Sami Terho, Minna Hirsi, Aarne Halme, and Pekka Forsman. Using signs for configuring work tasks of service robots. *Industrial Robot: An International Journal*, 33(4): 308–311, 2006.

Derek Hoiem, Alexei Efros, and Martial Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15–15, October 2008.

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2):179–187, February 1962.

Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5): 433–449, May 1999.

Kenji Kaneko, Fumio Kanehiro, Shuuji Kajita, Hirohisa Hirukawa, Toshikazu Kawasaki, Masaru Hirata, Kazuhiko Akachi, and Takakatsu Isozumi. Humanoid robot HRP-2. pages 1083–1090 Vol.2, 2004.

Tobias Kaupp. *Probabilistic Human-Robot Information Fusion*. PhD thesis, July 2008.

Tobias Kaupp, Bertrand Douillard, Fabio Ramos, Alexei Makarenko, and Ben Upcroft. Shared environment representation for a human-robot team performing information fusion. *Journal of Field Robotics*, 24(11-12):911–942, 2007.

Raphaël Labayrade, Didier Aubert, and Jean-Philippe Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. pages 646–651, 2002.

Ilkka Leppänen. *Automatic locomotion mode control of wheel-legged robots*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2007.

Rainer Lienhart and Jochen Maydt. An extended set of Haar-Like features for rapid object detection. In *IEEE ICIP 2002*, volume 1, pages 900–903, 2002.

David G. Lowe. Distinctive image features from Scale-Invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision, 1981.

Yi Ma, Stefano Soatto, Jana Kosecka, and Shankar S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

Jiří Matas. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, September 2004.

Oscar M. Mozos, Axel Rottmann, Rudolph Triebel, Patric Jensfelt, and Wolfram Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55:391–402, 2007.

Kazuyuki Nagata, Takashi Miyasaka, Dragomir N. Nenchev, Natsuki Yamanobe, Kenichi Maruyama, Satoshi Kawabata, and Yoshihiro Kawai. Picking up an indicated object in a complex environment. pages 2109–2116, October 2010.

Victor Ng-Thow-Hing, Evan Drumwright, Kris Hauser, Qingquan Wu, and Joel Wormer. Expanding task functionality in established humanoid robots. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 136–142. IEEE, November 2007.

Victor Ng-Thow-Hing, Jongwoo Lim, Joel Wormer, Ravi K. Sarvadevabhatla, Carlos Rocha, Kikuo Fujimura, and Yoshiaki Sakagami. The memory game: Creating a human-robot interactive scenario for ASIMO. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 779–786. IEEE, September 2008.

Victor Ng-Thow-Hing, Kristinn Thorisson, Ravi Sarvadevabhatla, Joel Wormer, and Thor List. Cognitive map architecture. *IEEE Robotics & Automation Magazine*, 16(1):55–66, March 2009.

Curtis W. Nielsen and David J. Bruemmer. Hiding the system from the user: Moving from complex mental models to elegant metaphors. pages 756–761, 2007.

Andreas Nuchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, November 2008.

Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Gray scale and rotation invariant texture classification with local binary patterns. In *Computer Vision - ECCV 2000*, volume 1842 of *Lecture Notes in Computer Science*, chapter 27, pages 404–420. Springer Berlin / Heidelberg, Berlin, Heidelberg, April 2000.

Florin Oniga, Sergiu Nedevschi, Marc M. Meinecke, and Thanh B. To. Road surface and obstacle detection based on elevation maps from dense stereo. In *2007 IEEE Intelligent Transportation Systems Conference*, pages 859–865. IEEE, September 2007.

OpenCV. Opencv wiki. `http://opencv.willowgarage.com`, Referenced 14.6.2010, 2011.

Julia Peltason, Frederic H. K. Siepmann, Thorsten P. Spexard, Britta Wrede, Marc Hanheide, and Elin A. Topp. Mixed-initiative in human augmented mapping. pages 2146–2153, May 2009.

Ananth Ranganathan and Frank Dellaert. Semantic modeling of places using objects. In Wolfram Burgard, Oliver Brock, and Cyrill Stachniss, editors, *Robotics: Science and Systems III*, Atlanta, GA, USA, June 2008.

Jari Saarinen, Mika Hyvönen, Jussi Suomela, Jani Vilenius, Aarne Halme, and Kalevi Huhtala. Development of multi-machine remote control platform. In *Proceedings of the 10th International Conference on Fluid Power 2007*, 2007.

Stephen Se and Michael Brady. Ground plane estimation, error analysis and applications. *Robotics and Autonomous Systems*, 39(2):59–71, May 2002.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision - ECCV 2006*, volume 3951, chapter 1, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

Neo E. Sian, Takeshi Sakaguchi, and Kazuhito Yokoi. Operating humanoid robots in human environments. In *In Proceedings of the Robotics, Science & Systems Workshop on Manipulation for Human Environments, Philedelphia*, 2006.

Anthony Stentz and Is C. Mellon. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10:89–100, 1993.

Yasushi Sumi, Yoshihiro Kawai, Takashi Yoshimi, and Fumiaki Tomita. 3D object recognition in cluttered environments by Segment-Based stereo vision. *International Journal of Computer Vision*, 46(1):5–23, January 2002.

Jussi Suomela, Jari Saarinen, and Aarne Halme. Creating common presence for a multientity rescue team. In *Proceedings of the 16th IFAC World Conference 2005*, 2005.

Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991.

Fayez Tarsha-Kurdi, Tania Landes, and Pierre Grussenmeyer. Hough-Transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from lidar data. 2007.

Sami Terho. Using stereo cameras in MaCI. In *1st International Symposium for GIMNet/MaCI Developers*, May 2010.

Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1), 1998.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press, August 2005.

Elin Topp and Henrik Christensen. Topological modelling for human augmented mapping. pages 2257–2263, October 2006.

Elin A. Topp, Helge Huettenrauch, Henrik I. Christensen, and Kerstin Severinson Eklundh. Bringing together human and robotic environment representations - a pilot study. pages 4946–4952, October 2006.

Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, March 1998.

Markus Ulrich, Christian Wiedemann, and Carsten Steger. CAD-based recognition of 3D objects in monocular images. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2090–2097, Piscataway, NJ, USA, 2009. IEEE Press.

James P. Underwood, Andrew Hill, Thierry Peynot, and Steve J. Scheding. Error modeling and calibration of exteroceptive sensors for accurate mapping applications. *J. Field Robotics*, 27(1): 2–20, 2010.

Shrihari Vasudevan. *Spatial Cognition for Mobile Robots: A Hierarchical Probabilistic Concept-Oriented Representation of Space*. PhD thesis, March 2008.

Shrihari Vasudevan, Stefan Gachter, Viet Nguyen, and Roland Siegwart. Cognitive maps for mobile robots - an object based approach. *Robotics and Autonomous Systems*, 55(5):359–371, May 2007.

George Vosselman, Er Dijkman, Key Words Building Reconstruction, Laser Altimetry, and Hough Transform. 3D building model reconstruction from point clouds and ground plans. *Int. Arch. of Photogrammetry and Remote Sensing*, pages 37–43, 2001.

VTT. Alvar – augmented reality software. `http://www.vtt.fi/multimedia/alvar.html`, Referenced 14.6.2011, 2011.

Joviša Žunić, Kaoru Hirota, and Paul L. Rosin. A hu moment invariant as a shape circularity measure. *Pattern Recognition*, 43(1):47–57, January 2010.

Sami Ylönen. *Modularity in service robotics*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2006.

Kazuhiko Yokoyama, Hiroyuki Handa, Takakatsu Isozumi, Yutaro Fukase, Kenji Kaneko, Fumio Kanehiro, Yoshihiro Kawai, Fumiaki Tomita, and Hirohisa Hirukawa. Cooperative works by a human and a humanoid robot. pages 2985–2991, 2003.

Aalto-DD 104/2011

ISBN 978-952-60-4332-6 (pdf)
ISBN 978-952-60-4331-9
ISSN-L 1799-4934
ISSN 1799-4942 (pdf)
ISSN 1799-4934

Aalto University
School of Electrical Engineering
Department of Automation and Systems Technology
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS