Indoor Vision-based Robot Navigation: a Neural Probabilistic Approach

Dissertation with the aim of achieving a doctoral degree at the Faculty of Mathematics, Informatics and Natural Sciences

Department of Informatics
of Universität Hamburg
submitted by Wenjie Yan
Hamburg, 2015

Day of oral defense: 17.03.2016
The following evaluators recommend to admit this dissertation:

Prof. Dr. Stefan Wermter
Prof. Dr. Jianwei Zhang

Abstract

Socially assistive robotics has experienced a steadily growing interest over the last decade. Mobile robots with sensors and external sensing devices, which may also be integrated synergistically, enable a human-like interaction with the environment. As a precondition for assisting a person, a robot needs to navigate to the position of the person first. To achieve this, the robot needs to localize the target person and itself; calculate an optimal trajectory; and adapt its motion in real time during navigation. Despite advances in mobile robotics, autonomous indoor robot navigation is still challenging. Due to the high complexity of the real world and possible dynamical changes, it is hard to localize a person or a robot robustly in an unconstrained environment. Sophisticated sensor systems can improve the localization, but will increase the system costs significantly and require a person with expert knowledge for setup. On the other hand, although it is a mature research field, robot autonomous navigation still relies on artificial landmarks and the accuracy of sensor signals. This limits the flexibility and the robustness of robot navigation as well. In addition to navigation in cluttered environments, a robot needs to cope with dynamic conditions while taking into account the presence of

its human companion.

This thesis presents a novel framework for vision-based person/robot localization and navigation. A hybrid neural probabilistic architecture is developed to localize a service robot and a person in a home-like environment using a ceiling-mounted camera. Considering that an integration of multiple visual cues can increase the detection reliability, multiple visual cues are employed and combined with a Sigma-Pi network. Through fast adaptation, parts of the visual cues can be learned during tracking, which increases the system robustness.

Taking into account the latest research insights into the neural mechanisms of decision-making during goal-directed navigation, we developed a neural-inspired system for robot navigation based on learning the spatial information. A spatial memory is built to represent the environment and a sensorimotor representation to learn control signals of the robot's navigation behavior. Based on learned spatial and sensorimotor information, a robot can reach arbitrary target positions by real-time motion planning. As one of the advantages of our navigation system, the motion signal is processed by a fast combination of the sensorimotor memory information instead of being calculated according to the current position and the map information. This fast combination accelerates the computation and ensures a real-time behavior during the robot's navigation. In addition, considering the fact that a person could move around in a room without colliding with objects, we have developed the system to learn the spatial information by observing the person's movements. Through an integration of reactive behaviors with the motion planning, the robot is able to adapt its spatial representation and avoid obstacles proactively.

A camera integrated into the head of the robot is used to learn the appearance of the environment during navigation. Visual features of the current camera view are extracted and associated with the state of the robot in the spatial memory, which builds up a view-based memory of the environment context. This memory, together with the cognitive map, provides the robot with the ability to solve complex cognitive tasks such as fetching an object observed during previous navigation.

Experimental evaluations of the localization and the navigation models are conducted in a home-like laboratory as well as through field trials of the KSERA project. A humanoid Nao robot is used to test the navigation, obstacle avoidance, and the learning of the appearance of the environment. Through analysis of the results, we show that through tracking a person with integration of adaptive visual cues, the localization system is able to learn the environment, which enables the robot to navigate to target positions robustly. When the environment changes, the robot can adapt its behavior autonomously and learn to avoid the obstacles in the future.

In conclusion, our research presents a novel neural probabilistic robot localization and navigation system that provides a mere concept while enabling far-reaching functionality. Through emulating several basic functionalities of the brain– for example, the spatial cognition and redundant information representation for target recognition–the system is able to achieve complex tasks such as robust target tracking, environment learning through observation, and flexible robot navigation in a home-like environment. The concept of our work is implemented and evaluated using a robot platform in a home-like environment, and the experimental results show that our

neural system helps a robot to realize different functions successfully. We believe that through the further development of artificial intelligence, robotic sensors, and robotic hardware, highly intelligent functions will be realistic and assist people in their daily lives.

Kurzfassung

Sozial-assistive Robotik hat ein stetig wachsendes Interesse im letzten Jahrzehnt erfahren. Mobile Roboter mit Sensoren und externen Erfassungsvorrichtungen, welche auch synergistisch integriert werden können, ermöglichen eine menschenähnliche Wechselwirkung mit der Umgebung. Als Voraussetzung zur Unterstützung einer Person, muss ein Roboter zuerst zu der Position der Person navigieren. Um dies zu erreichen, muss der Roboter die Zielperson sowie sich selbst lokalisieren, eine optimale Trajektorie berechnen, und seine Bewegung während der Navigation in Echtzeit anpassen. Trotz der Fortschritte im Gebiet der mobilen Robotik, ist die autonome Indoor-Navigation immer noch eine Herausforderung. Aufgrund der hohen Komplexität der realen Welt und auch aufgrund der möglich dynamischen Änderungen der Umgebung, ist es schwierig, eine Person oder einen Roboter robust in einer ungezwungen Umgebung zu lokalisieren. Ausgefeilte Sensorsystemene können die Lokalisierung verbessern, jedoch können sie die Systemkosten deutlich erhöhen und benötigen eine Person mit Expertwissen für die Einrichtung. Auf der anderen Seite, trotz ausgereifter Forschung, ist autonome Roboter-Navigation immer noch auf künstliche Landmarken und die Genauigkeit der Sensor-

signale angewiesen. Dies begrenzt insbesondere auch die Flexibilität und die Robustheit der Roboter-Navigation. Neben der Roboter-Navigation in schwierigen Umgebungen, muss ein Roboter dynamische Bedingungen bewältigen, aber auch die Anwesenheit des menschlichen Partners berücksichtigen.

Diese Arbeit präsentiert einen neuartigen Ansatz für die Bildverarbeitung-basierte Person / Roboter Lokalisation und Navigation. Eine hybride neuronale Wahrscheinlichkeitssystem wurde entwickelt, um einen Serviceroboter bzw. eine Person in einer hausähnlichen Umgebung mittels einer Deckenkamera zu lokalisieren. Damit die Integration von mehreren visuellen Signale die Detektionsrobustheit erhöhen kann, sind in unserem System mehrere visuelle Signale verwendet und mit einem Sigma-Pi-Netzwerk kombiniert. Teile der visuellen Signale können während der Verfolgung durch eine schnelle Anpassung erlernt werden, welche die Robustheit des Systems weiterhin verbesseren.

Unter Berücksichtigung der neuesten Forschungseinblicke in die neuronalen Mechanismen der Entscheidungsfindung für die zielgerichtete Navigation, haben wir ein neuroinspiriertes System für Roboternavigation basierend auf dem Erlernen der räumlichen Informationen entwickelt. In diesem System ist ein räumliches Gedächtnis eingebaut, um die Umgebungsinformation darzustellen und eine sensomotorische Repräsentation für die Steuersignale des Roboter-Navigationsverhaltens zu lernen. Basierend auf der gelernten räumlichen und sensomotorischen Informationen, kann ein Roboter beliebige Zielpositionen durch Bewegungsplanung in Echtzeit erreichen. Als einer der Vorteile unsereres Navigationssystems, wird das Bewegungssignal durch eine schnelle Kombination der sensomo-

torischen Speicherinformationen verarbeitet, statt dass die Signale gemäß der aktuellen Position und der Karteninformation berechnet werden muss. Diese schnelle Kombination beschleunigt die Berechnung, und gewährleistet ein Echtzeitverhalten während der Roboternavigation. Darüber hinaus, unter Berücksichtigung der Tatsache dass eine Person sich in einem Raum ohne Kollision mit Objekten bewegen könnte, haben wir ein System entwickelt, um die räumliche Information durch die Beobachtung der Bewegung der Person zu lernen. Durch eine Integration von reaktiven Verhaltensweisen mit der Bewegungsplanung, ist der Roboter in der Lage, die räumliche Darstellung nach der Detektion der Hindernissen anzupassen und solchen gelernten Hindernissen proaktiv auszuweichen.

Eine in den Kopf des Roboters integrierte Kamera wird verwendet, um das Aussehen der Umgebung während der Navigation zu lernen. Visuelle Merkmale des aktuellen Kamerabildes werden extrahiert und mit dem Zustand des Roboters im räumlichen Gedächtniss verbunden, welches einen Sichtbasierten Speicher für den Umwelt-Kontext aufbaut. Dieser Speicher, zusammen mit der kognitiven Karte, stattet den Roboter mit der Fähigkeit aus, komplexe kognitive Aufgaben wie das Abrufen eines in früheren Navigations beobachteten Objekts, zu lösen.

Experimente der Lokalisierung sowie der Navigationsmodelle werden sowohl in einem heimähnlichen Labor als auch durch Feldversuche des KSERA Projektes durchgeführt. Ein humanoider Nao Roboter wird verwendet, um die Navigation, die Hindernisvermeidung und das Lernen des Erscheinung der Umgebung zu testen. Durch Analyse der Versuchsergebnisse wird gezeigt, dass durch die Verfolgung einer Person mittels

Integration von adaptiven visuellen Signale, unser System in der Lage ist, die Umgebung zu lernen. Dies ermöglicht dem Roboter zur Zielposition robust zu navigieren. Wenn die Umgebung sich ändert, kann der Roboter sein Verhalten selbständig anpassen und lernen, die Hindernisse in der Zukunft zu vermeiden.

Zusammenfassend, zeigt unsere Forschung ein neuartiges Neuro-Wahrscheinlichkeitssystem für die Roboter-Lokalisation und -Navigation. Durch die Emulation mehrere Basisfunktionalitäten des Gehirns - zum Beispiel die Raumkognition und redundante Informationsdarstellung für Zielerkennung - ist das System in der Lage, komplexe Aufgaben wie beispielsweise robuste Zielverfolgung, Lernen der Umwelt durch Beobachtung, und flexible Roboternavigation in einer heim-ähnlichen Umgebung zu erreichen. Das Konzept unserer Arbeit wird mit Hilfe einer Roboter-Platform in einer heimähnlichen Umgebung durchgeführt und evaluiert, und die experimentellen Ergebnisse zeigen, dass unser neuronales System helfen kann, um verschiedene Roboter-Funktionen erfolgreich zu realisieren. Wir glauben, dass durch die Weiterentwicklung der künstlichen Intelligenz, der Roboter-Sensoren, und der Roboter-Hardware, hoch intelligente Funktionen realistisch sein werden, welche das tägliche Leben der Menschen unterstützen.

Acknowledgement

The presented work in this thesis would never have been possible without the support and guidance of many people. I would like to use this opportunity to express my deepest gratitude to them.

First and foremost, I want to thank my Ph.D. supervisors, Professor Stefan Wermter and Dr. Cornelius Weber, for their sincere support, assistance, and encouragement. Cornelius, you taught me what computer science means and encouraged me to grow as a research scientist. Stefan, you provided me with the vision, motivation, and inspiration that I need to go through my doctoral journey. I cannot image better mentors than you both.

Special thanks also go to Professor Jianwei Zhang and Professor Frank Steinicke for serving as my committee members. I really appreciate your insights and valuable suggestions.

I would like to acknowledge all the people involved in the KSERA project for giving me such an unforgettable experience. I especially want to recognize Elena Torta and Marco Bazzani, with whom I worked hard to solve countless problems and make the impossible possible. It was really a joy working with you. Many thanks also go to the EU FP7 project

KSERA under grant agreement n°2010-248085 for the financial support.

Members of the Knowledge Technology group deserve my heartfelt thanks for their friendship, assistance, and inspiration. In particular, many thanks go to Erik Strahl for supporting my work and Katja Kösters for spending so much time improving my language. Also many thanks to Stefan Heinrich for the warm hearted help as always. I would also like to express my gratitude to Heidi Oskarsson and Dieter Jessen for supporting my life at the beginning of my doctoral study.

Finally, a very special word of thanks goes to my parents, Yuefang Yang and Ronghai Yan, and my wife, Jing Tian, for their love, trust, encouragement, and unconditional support over these years. Without you, I am nothing. Also, thanks to Chenxin Tina Yan, my baby girl. You have become my morning star and have motivated me to persevere and get things done. This thesis is dedicated to her.

~ For Jing and Tina ~

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the enormous developments in the field of robotics and computer science, robots have become more intelligent and are capable of accomplishing more complex tasks. They are not restricted to industrial usage anymore, which is well-constructed, and could be applied in the field of medical care and artificial assistants. In particular, socially assistive robotics has received more attention in the last decades. Considering that the world-wide population aging problem that will become more severe over time, socially assistive robots could be companions to assist the elderly and improve their quality of life eliminating the need for caretakers. According to statistical analysis (Hootman and Helmick, 2006; Steg et al., 2006), the proportion of the elderly is growing quickly and there will be a large demand for working-age persons to support persons over 65 years of age in the future, even through the manpower of those of working age is decreasing steadily.

This challenge motivates researchers to find new solutions

to assist elderly persons. Intelligent home-care systems, which could monitor the health statuses of elderly persons and react when needed, have received more attention during the last decade. Progress in artificial intelligence (AI) (Wermter et al., 2005, 2014) shows that these new technologies will extend the functionalities of traditional domestic automation devices, which will benefit the elderly to improve their quality of life (Steg et al., 2006). Based on state of the art AI, such a system can monitor and diagnose the health state of users and provide active assistance with a domestic automated system. For example, through integration of ubiquitous monitoring and intelligent actuation systems in a home-care system, an ambient assisted living system (AAL) can support the user's daily activity and increase his life independence significantly (O'Grady et al., 2010). The AAL system can adjust the environment condition based on the sensor measurement. For example, it can control the room temperature, and monitor the user's health status (e.g., blood pressure and heart beat ratio) by analyzing signals from sensors to provide corresponding medical advice. In case of emergency, such as if the user falls, the system will detect the danger, monitor the user's status, assist the user and eventually contact the health organization when needed.

However, since these ubiquitous domestic systems are usually fixed in the home, they cannot provide active assistance, for example, bringing a medicament to the user. Compared with a fixed system, a mobile robotic system provides an attractive alternative based on its high mobility and multiple functionalities. Recent advancements of robotics show that robots will not only be used in industry but also become a com-

panion of humans in their daily lives in the near future (Broxvall et al., 2006; Louloudi et al., 2010). A robot could perceive the environment, realize cognitive behavior, and accomplish complex tasks using state-of-the-art artificial intelligence. A robot would be capable of performing in an environment made for humans, providing assistance, and interacting with the user. It could assist the elderly effectively in daily tasks like communicating with the external world, providing medicine and health check reminders in a proactive fashion, and demonstrating physical exercises. Because a robot, in particular a humanoid robot, would behave similarly to human beings and could interact with users through verbal or nonverbal communication, users might accept these new robotic technologies more readily (Torta et al., 2011b).

The combination of a service robotic system and an AAL system shows a promising way to realize a more intelligent service robot. As a common robot perceives information with integrated sensors, which have a limited sensor range, it may not detect objects when they are outside of the sensor range. Therefore, it is hard for a robot to learn the surrounding environment properly. Sensors of an AAL system distributed in the home environment could help the robot, in this case, to perceive the environment more precisely and quickly. With the new information acquired from these sensors, the robot could detect events outside of its sensor range and react correspondingly. On the other side, the mobile robot platform can act as an active actuator of the AAL system. The ambient environment will not only respond passively to a change of a situation but can also provide active help with a service robot. According to the request to the AAL system, the robot can record the

health status of the user, check the environment status, and send real-time data to medical organizations (e.g., hospitals or medical teams that take care of the patient), which helps the doctors in diagnosing the patient and adjusting the environment. When an accident happens, the robot can send an alarm signal to the medical team on time and act like the first aid for rescue. As this combination could improve the robot's service quality significantly and benefit the elderly's life, many research projects towards this new combination have been conducted. A summary of the recent relevant research projects will be given in the following section.

The concept of combining a robot with an AAL system arouses great interest in research communities due to the enormous potential social impact and benefit for the elderly. As more advanced functions can be achieved by such a networked system, research has been conducted to develop service assistive robots in the last decade. For example, within the research project Companionable[1] (Integrated Cognitive Assistive & Domestic Companion Robotic Systems for Ability & Security), an innovative architecture was developed that integrates robotics and ambient intelligence technologies synergistically to support elderly in their daily lives. A mobile companion robot platform (Hector) was designed to assist the user with an innovative way of interaction. The Hector robot could move automatically in the home environment based on its wheeled platform and respond to the user based on voice inputs. As a part of the smart home, the robot was equipped with a large touchscreen that could display environment information and control the home system based on the touch commands. The

[1] `http://www.companionable.net/`

robot also assisted the user to improve his social activities. It could easily establish communication, using the touchscreen and microphones, with friends, health carers, etc.

Similar research projects, such as Florence[2], and project ROBO M.D.,[3] developed different robot platforms for person assistance. Through semantic cooperation with a smart-home system, a robot could provide a reliable and efficient health monitor of users and improve their quality of life. A robot could recognize critical situations (e.g., fall detection) based on its sensor inputs and assist a medical institute in diagnosing the user at an early stage through remote communication.

Another research project which leads to the results of this thesis is the KSERA project[4]. International partners worked together to build up a novel system for serving the elderly, in particular patients suffering from Chronic Obstructive Pulmonary Disease. A humanoid NAO robot was employed to support the activities of daily life, as well as the healthcare needs of an elderly person. Through the integration of the robot platform with an AAL system, an innovative architecture was built which improves the functionalities of both. For the AAL system, the NAO robot acted as a unique interface between the user and the AAL system. On the other hand, the ubiquitous monitoring of the AAL extended the view of the robot, which helped it to extend its functionalities. For example, a ceiling-mounted camera of the AAL system could be helpful for the robot to localize the user's location without using any artificial landmarks enabling the robot to navigate

[2] http://www.florence-project.eu/

[3] http://www.innovation4welfare.eu/307/subprojects/robo-m-d.html

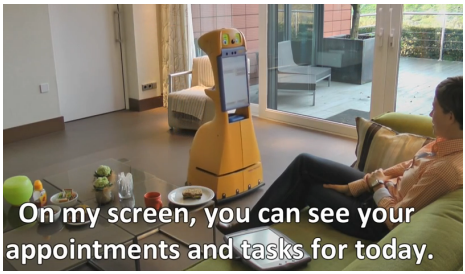
[4] http://ksera.ieis.tue.nl/

fully automatically towards the user.

As one of the most important research topics in KSERA, the psychological impact of a service robot to the user's daily life has been studied deeply (Johnson et al., 2014). Through investigation of users' feedback, researchers had found that users could accept service robotics more easily using verbal and non-verbal interaction with the robot. For instance, users could adopt medical advice and exercise physical activities by following the demonstration of a friendly humanoid robot. Compared with traditional methods, e.g., playing videos, the interaction with a human-like robot could encourage users to improve their lifestyles in a more active manner.

1.1 Challenges for Applying Robots in Real Environments

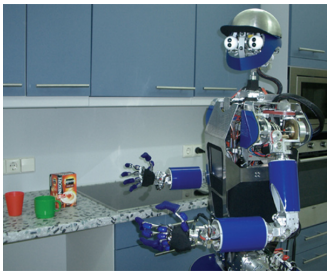
As we have described in the last section, research on integrating robots into AAL systems is drawing more attention from society, and robots are now providing a promising vision that they can be companions (for humans) to assist us and improve our lives in the future. However, the challenges of developing robotic hardware (e.g., battery) and software still prevent their application in our lives. While traditional machines need to be controlled by users, a robot should reach a high degree of autonomy in decision-making, interaction with users and behavior adaptation when the environment changes. In this case, artificial intelligence is crucial because it provides robots with a human-like ability to learn and to react to unknown conditions.



(a)

(b)

(c)



(d)

Figure 1.1: Service robots.

1.1(a): A service robot is providing information to the user[1].

1.1(b): A DarwinOP humanoid robot is stacking a towel[2].

1.1(c): A roboter is assisting a doctor in an emergency. (Parisi et al., 2014)

1.1(d): An ARMAR III robot is helping dish washing. (Bauer et al., 2010)[3]

[1]Source: a snapshot from a video of the European FP7 project "Companionable" led by Professor Atta Badii.

[2]Source: an image from the project "Teaching With Interactive Reinforcement Learning" led by Professor Stefan Wermter, WTM, UHH.

[3]Source: an image from the research project "SFB 588: Humanoid Robots - Learning and Cooperating Multimodal Robots", KIT.

A large number of research topics have been addressed to develop different human-like robotic functionalities. Among them, robotic mobile behavior (including localization and navigation) is one of the most favorite topics because it addresses the fundamental ability of humans for spatial cognition. Furthermore, from an application point of view, the mobile behavior is a precondition for realizing many other functions. When a robot is requested to deliver medication, the robot needs to determine the position of the target user and the medication, and maneuver itself to them without collision. In case of emergency, the robot should move itself to the user to assist a medical helper, for instance, to diagnose the user's health situation remotely.

A significant difference between an industrial robot and a socially assistive robot is that while the traditional industrial robots work in a well-constructed laboratory environment, a socially assistive robot has to work in a real, homelike environment. This environment increases the complexity of robot navigation significantly because the home-like environment is usually unknown and cluttered, and the robot has to share space with users. Although many models for map-building and planning exist, normal hand-defined methods may not cover all possible situations and cause problems while running because of the high complexity of a real environment. Robotic approaches with sophisticated sensors could overcome the problem of map building and localization by improving the data precision and detection range. For instance, a laser scanner could measure a large distance reliably with high accuracy, which would help the robot to create a precise map. However, the high cost of these sensors may be unaffordable for the

end user. In contrast, simple sensors could reduce the system price, but the limited functions of them may cause the problem of their signals being too rough and unreliable. Furthermore, due to the unpredictable situation in the real environment, the robot needs to adapt its behavior according to the changes in the environment. All these problems together obstruct the usage of assistive robots in our daily life.

1.2 Contributions of this Work

Considering that humans and animals can achieve far more complex tasks than robots, an intelligent system based on human-like behavior and learning functionalities could accomplish more complicated cognitive tasks in home-like environments than industrial fields. In particular, the understanding of how information is processed in the brain could be a key to realizing a robust self-positioning and navigation. How the brain works has been investigated deeply in the last decades. Although many parts of the brain remain unknown, more knowledge about the structure and the way of information processing has been obtained. Because artificial neural networks emulate the structure and function of the brain, they could be a method to reproduce some functionalities of the brain such as feature learning and recognition, spatial recognition, information fusion and decision making. In this work, we focus on the investigation of the neural mechanisms of spatial cognition and develop an automatic intelligent navigation system based on these insights.

1.2.1 Indoor Localization based on a Hybrid Neural Probabilistic Model

Indoor person and robot localization is crucial for socially assistive robots. A robot needs to know the position of the user and navigate to him to provide service. Usually, since no wearable sensor is carried by the user for localization, the user or target person needs to be detected by sensors, for instance using cameras, identified when multiple persons are in the room, and must continue to be tracked without disturbance. However, a robust localization of the user is hard to obtain due to the complexity of the real environment. For example, a vision-based person localization could fail due to the image noise, the dynamic of the light intensity, and/or the change of the target person's features. Considering that humans and animals recognize objects based on different features (e.g., color, shape, size, etc.), a reliable information-fusion could help the system improve the robustness of target detection. Hence, a hybrid neural-probabilistic model is presented that integrates multiple visual cues obtained from the ceiling-mounted camera to detect and track the target person.

Traditional vision-based methods, such as motion detection and feature matching rely on a single type of information and work well only in a constrained environment. For example, as a motion detector is sensitive to objects that move, a user may be missed when (s)he does not move. Feature matching based on an object's shape (e.g., SIFT) works well only when the object is rigid, but due to the distortion of the lens, the person's shape may differ strongly when his position and pose are changed. The clothes' color pattern could be a strong evidence for a person's detection, but this pattern needs to be trained

before use. Moreover, the learned color pattern may not work and needs to be regenerated when users change their clothes (which could frequently happen). Compared with these, while some visual cues cannot provide useful information (e.g., no motion is detected when the person sits on a chair), the redundant target detection (e.g., from the shape cue and the color cue) could help the system to detect the target further. Therefore, as one of the key features of our model, our neural model combines diverse information obtained from the ceiling-mounted camera and is capable of compensating weaknesses of visual features in different situations, which results in an overall reliable detection rate.

In order to realize a real-time localization performance, a particle filter is employed to detect the person based on outputs of visual cues. The particle filter here resembles the mechanism of visual attention: instead of processing the entire image information, the particle filter first processes the visual input with a uniform random sampling and then focuses on image areas that contain relevant information. With this novel method, the computational effort of using multiple methods decreases significantly, enabling the system to detect the target person in real-time. Moreover, an adaptive learning scheme is developed to learn visual patterns (i.e., the color and the shape cue) and to adjust the reliabilities of visual cues, which improves the tracking quality online.

1.2.2 Indoor Robot Navigation based on Learning of Sensorimotor Representation

The navigation system uses the localization information and controls the robot to walk to the target person. As the target

person may walk to different places, no fixed target position can be defined, and the robot should adapt its behavior in real time when the person moves. Therefore, the navigation system needs to have a high flexibility to respond to an environment change instantly while keeping the robot safe. Sophisticated path planning methods may be unsuitable for our requirements because of the high computational complexity. In addition, the well-planned trajectory may not fit the real situation because the environment may change during planning. A human-like behavior is desired, in which an abstractive planning (e.g., direction to the target) should be computed quickly, while the reaction to the details of the environment (e.g., obstacles) should be calculated in real time during navigation. Study of the spatial cognition ability of humans and animals is helpful, in this case, to develop an intelligent navigation system.

Consistent with the suggestion that for humans and animals, a spatial environment can be represented with sensorimotor features (Zetzsche et al., 2009), we present a neural probabilistic model for the robot navigation based on learning the sensorimotor representations of an indoor environment. The neural model represents the environment at a high level of abstraction–i.e., in a sensorimotor map that stores the spatial information in a self-adaptive structure. The sensorimotor map consists of a topological network, in our model a growing when required network, to represent spatial features and relations of the environment. Based on the information stored in the topological network, a strategic navigation plan is built, which is computed efficiently and can be adapted based on the change in the environment. By combining this strategic plan with a reactive obstacle avoidance behavior, this model is ca-

pable of navigating a robot in a real environment with high complexity and dynamic changes.

The motor information of the state of transition is represented as connection weights in the network, which are used to interpolate the control signal while navigating. During map exploration, detailed actions of the sensorimotor are learned through associating spatial status with the motor information of the robot. This motor information is then reused during robot navigation to generate corresponding motion signals. For each navigation step, the corresponding motor information is triggered and combined without extra geometry calculation. With the help of this novel method, the robot can handle complex situations while keeping high planning efficiency using learned a-priori motion information. Compared with methods that regenerate motion information in real time during navigation, this combination of learned knowledge accelerates the computation and ensures real-time planning.

The sensorimotor map can be learned based on the robot's exploration or through observation of the motion of others. Considering that usually a person is moving in a room without colliding with objects, a new training mechanism is presented which learns the spatial knowledge by observing the person's movement using a ceiling-mounted camera. With the help of this method, the learning phase of the environment can be accelerated, and possible collisions during the robot's explorative movements are avoided. This novel method resembles the principle of "latent learning" (Thorpe, 1956), i.e., learning the environment without an expressive response at that time. When a target is given, the collected experience together with the reward signals compose a model-based reinforcement

learning, which in our case guides the robot to maneuver to the target position.

Moreover, because of the different sizes of a person and a robot, the sensorimotor map learned from a person's movements may not be perfectly suited to the robot. Therefore, an online adaptive learning mechanism is developed, which adjusts the observed spatial knowledge (stored as connection weights of the topological network) to the robot while maneuvering. A reflective behavior is defined for robot-environment interaction during navigation, which quickly controls the robot to avoid possible collisions based on the sensor signals of the robot. Based on the adaptive learning and the reflective behavior, a robot can remember the obstacles in its sensorimotor map and avoid them pro-actively in the future.

As one unique feature of our approach, during the robot's exploration in the room, the visual information extracted from the robot's head camera is associated with the location memory, which anchors the appearance features of the environment with the states in the sensorimotor map. This technique allows the robot to achieve complex tasks like fetching an object by being shown an image of it. It shows another advantage of our model that multiple sources of abstract information can be used for environment representation, which is also a fundamental ability of human's spatial cognition.

The localization method and the navigation method together build up a highly integrated system comprised of the robot and the AAL system. It provides an effective way of environment learning, target person detection, and autonomous navigation. With the help of the AAL system's sensors, the robot is capable of learning the environment without using

sophisticated sensors, e.g., laser scanners, which reduces the system cost and extends the capability of the robot to assist a person. A socially assistive robot can be a unique interface between the AAL system and the user which helps the medical team to support the patient.

1.3 Structure of the Thesis

This thesis is organized as follows: in chapter 2 we review the state of the art of indoor localization and robot navigation. We also discuss the relevant computational models of neural representations and spatial cognition, which are helpful for understanding the humans' and animals' navigation ability. Then, in chapter 3, we present the hybrid probabilistic model for indoor person/robot tracking using a ceiling-mounted camera. The method of flexible robot navigation using a cognitive map is described in chapter 4, and a model of visual anchoring of the environment is explained in chapter 4.3, which helps a robot to learn and to localize a specific object in the environment. Methods for adaptive learning of the environment dynamics are described in chapter 5. In the end, we summarize this thesis with a discussion in chapter 6.

Chapter 2

Related Work

Socially assistive robotics aims to move robots from well controlled environments (such as laboratories or factories, where robots operate under limited conditions), into real social environments (e.g., schools, hospitals and homes) to assist persons in their daily lives. Among the many challenges this field present, how a robot has to adapt its mobile behavior in a cluttered, unknown environment, possibly with dynamic changes within a space, is one of the most crucial problems, and it is closely related to the successful performance of the robot. Fully programmed behaviors used for industry robots are not realistic in this case. First, as the environment in which the robot moves is unknown at the beginning, the navigation plan cannot be computed without the a priori knowledge. Second, the target position could be dynamic in a real-life scenario (e.g., the target person could move), and the robot needs to adjust its plan in real time instead of following the one programmed in advance. Therefore, a robot needs to explore its

environment using its sensors, understand the situation, and adjust the navigation towards the target position based on its estimation. Computational intelligence plays an important role in this field of research, particularly important is its work on artificial neural networks based on studies on the learning mechanisms of the central nervous system of animals and humans.

Recent research in robotics and computational intelligence shows that socially assistive robots have made considerable progress in the last decades. In this chapter, we will give an overview of the state-of-the-art techniques for the robot localization, navigation, and neural mechanisms of humans' and animals' spatial knowledge. In section 2.1, we will review the methods developed for localization, in particular, in an indoor environment based on different sensors. Then, in section 2.2, we will describe the current progress of autonomous robot navigation, including mapping, planning, and interaction with the environment. We will also review how a robot interacts with the environment in section 2.3, and the biological findings of spatial cognition as well as the related computational models developed during recent years in section 2.4.

2.1 Localization

Localization refers to the techniques that determine the position of an object in space, which is one of the fundamental problems in robotics. For a socially assistive robot, a robot needs to identify its own position based on the internal or external sensor inputs and use this information to plan its trajectory to the target. The localization as well as the navigation

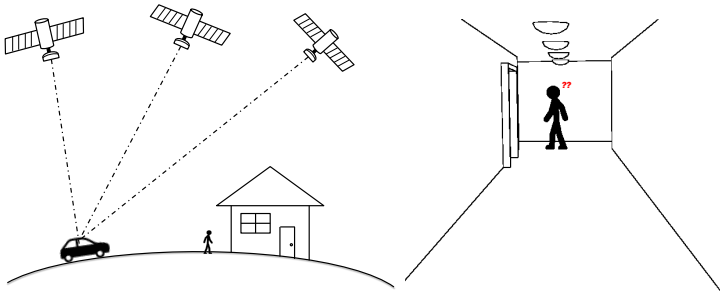


Figure 2.1: Schema of outdoor and indoor localization

are preconditions to many other tasks, such as human-robot interaction and providing medical assistance, because the robot needs to reach the person first before providing any service.

2.1.1 Non-vision-based Localization

In general, the localization problem can be categorized as *outdoor* and *indoor* localization (see Figure 2.1). While outdoor localization is well addressed using advanced signal processing and satellite techniques, for example, the Global Positioning System (GPS), the indoor localization remains challenging for several reasons. First, as the satellite signals cannot penetrate the building structure, the GPS localization does not work in an indoor environment. Second, the indoor localization requires a much higher accuracy than the outdoor localization because of the relatively small environment. Due to the cluttered structure of the indoor environment, signals may be distorted or reflected, which leads to a high signal-to-noise ratio. The environment dynamics is another challenge for the localization, which may cause a mislocalization due to the mis-

matching of the current sensor data with any landmarks stored in the map.

Many methods for indoor robot localization have been proposed to overcome these challenges in the last decades. Non-vision-based localization techniques, for example, radio-based localization measurements (e.g., RFID tags (Koch et al., 2007; Raoui et al., 2009), GSM networks (Varshavsky et al., 2006) and wireless sensor networks (Mao et al., 2007)) have been developed successfully. Within these methods, we can further distinguish active models, where the target object collaborates with the localization, for example, by transmitting a radio signal, and passive models, where objects are independent and the system works based on the reflected signals (for example, with a passive RFID). One drawback of radio-based localization is the low accuracy of the calculation to determine position because no direct distance measurement is conducted. Redundant base stations such as wireless sensor networks have to be built in order to obtain a relatively good precision.

Another localization strategy uses distance information directly measured by sensors. For instance, distance information can be calculated with laser and ultrasonic sensors by estimating the difference of the time between the emission of a signal and its reception after being reflected by objects (Adams et al., 2004; Drumheller, 1987; Hahnel et al., 2003; Kleeman and Kuc, 1995). However, this measurement (usually called time-of-flight technique) is very complicated. Laser scanners provide great accuracy of distance measurement and a wide detection range, but their price is so high that they are not suitable for consumer applications. Ultrasonic sensors are relatively cheap compared to laser scanners, but they provide a

lower resolution measurement (i.e., a sonar sensor only measures the closest distance to obstacles in its detection range).

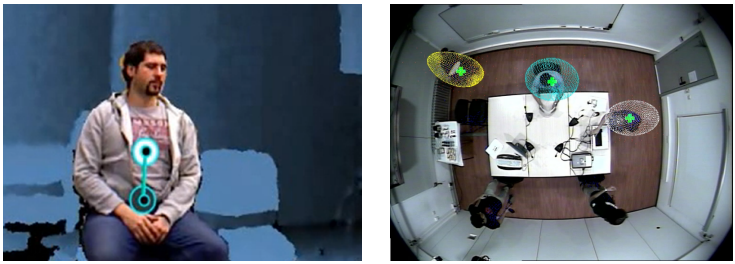
Since a robot needs to approach the target person before providing assistance, the position information of the target person is necessary for robot navigation planning. Although the principle of person localization is the same as that of robot localization, methods developed for mobile robots may not be suitable for person localization. For example, unlike the robot localization that can be obtained from the robot's internal sensors, which provide rich information based on a robot-centered environment, this information is missing for person localization, because these sensors cannot be fixed on the person's body. Although wearable sensors could be an alternative for providing this information, and many methods based on them have been proposed successfully in the last decades (e.g., using infrared (Want et al., 1992) and radio frequency signals (Bahl and Padmanabhan, 2000)), an extra sensor device has to be carried by the user permanently, which is inconvenient for lifelong person tracking or not allowed for safety reasons. Furthermore, since wearable sensors need to stay active while tracking, a power supplier is required to keep the sensor always active.

Another strategy of addressing the person localization problem is to use external sensors of the robot (e.g., a robot camera that detects the person based on the feature matching) or the AAL system to identify and track the target person, for example, using a ceiling-mounted camera. These methods do not require the user to carry any sensor, which is convenient for the daily usage. However, since the localization with these methods is based on feature detection and position estimation rather than accurate measurement, challenges such as detec-

tion robustness, detection range, and detection safety have to be solved. For example, motion sensors (Barger et al., 2005; Yang et al., 2009) can detect a person entering or leaving a room, but they cannot provide the precise location information due to their low resolution and short range. In addition, these sensors cannot distinguish a moving person from other objects. Laser sensors are capable of scanning the environment in a very detailed way and detecting a person at a long distance. However, laser emitters are usually expensive and may cause potential safety issues for a person. Moreover, as laser sensors only provide distance information, it is hard to identify a target person with them.

2.1.2 Vision-based Localization

Compared with non-vision-based localization models, a vision system promises to provide reliable performance and a broad functionality at a reasonable cost. Vision-based localization determines the position of the object by analyzing the image(s) from a camera, including person detection, segmentation, image registration, and reconstruction in real-time. As a major branch of artificial intelligence, computer vision has become one of the most important topics in computer science, and is being widely applied in smart home systems, robotics, etc. Vision systems use passive sensors, i.e., cameras to localize the target person through person detection and tracking. As stated before, these systems have a relatively low cost and do not need the user to carry any sensors. The camera stream provides far more information than the other kinds of sensors: it can be assessed whether the person is standing, sitting or moving, or whether it is an emergency situation such as a



(a) (b)

Figure 2.2: Person tracking based on different sensors

Left: The gesture of the person is estimated based on a neural network using a Kinect-like sensor[1].
Right: View of a camera from a camera array. Multiple cameras are used to track the person by information fusion[2].

fall (Nait-Charif and McKenna, 2004). The target person/object can be detected by texture, color, or motion information extracted from images and reconstructed in the 3D environment based on different technologies such as the structure of motion, epipolar geometry, etc. Privacy concerns of camera surveillance can be addressed by not storing image information if the person's location is needed only for a short time.

There are many vision techniques developed for localization based on different camera setups. For instance, stereo vi-

[1]Source: a snapshot from a video related to the article "Hierarchical som-based detection of novel behavior for 3d human tracking", (Parisi and Wermter, 2013)

[2]Source: a snapshot from a video linked with the article "Multi-person tracking strategies based on voxel analysis", (Canton-Ferrer et al., 2008)

sion systems (Bahadori et al., 2007; Muñoz-Salinas et al., 2007) are used to reconstruct the 3D information based on the epipolar geometry of two cameras and can detect persons based on the shape segmentation using depth information. Similarly, RGB-D sensors (e.g., Kinect and Xtion) (Luber et al., 2011) use a combination of an infrared signal emitter (which creates a known scattered point pattern) and a camera to calculate the depth based on triangular geometry. With these cost-effective sensors, a 3-D reconstruction can be achieved with a decent resolution and an excellent real-time capability. The drawbacks of these methods are the small detection range and a narrow angular field of view, which obstructs the usage of them. A Time-of-Flight (ToF) camera (Guomundsson et al., 2008; Knoop et al., 2006) measures the time between emission and reception of a signal to calculate the distance, which provides a high space resolution and can yield a very accurate 3D reconstruction. A disadvantage of the ToF camera is the complexity of the manufacture, which leads to high system costs. An infrared camera (Kemper and Linde, 2008) is a particular kind of camera used to localize the person by detecting a heat source, which is natural for identifying a person or a robot in the room because they usually emit heat. However, since multiple heat sources could exist (e.g., television, microwave, etc.), the sensor images may have a high signal-noise ratio.

Considering that a single camera may not cover the entire observation space or may not provide accurate position information, the fusion of multiple information may be helpful to improve the tracking quality. For example, a higher tracking resolution than that available when using traditional camera systems can be reached by registering motion informa-

tion from an accelerator sensor (Bauer and Lukowicz, 2008) or using a multiple-camera system. Person tracking with multiple sensors (Demiroz et al., 2012; Fleuret et al., 2008; Kobilarov et al., 2006; Nickel et al., 2005; Salah et al., 2008) can obtain extra information, which can offset drawbacks like the narrow field-of-view of a single camera (Lanz and Brunelli, 2008) and overcome shadowing and occlusions (Kemmotsu et al., 2008). However, these systems are rather complex and require expert knowledge to be set up. For example, a camera system has to be carefully calibrated using sophisticated methods not only to eliminate the distortion effect of the lenses, but also to construct accurately a 3D coordinate system based on different camera views.

Compared with these methods, a single ceiling-mounted camera provides a much simpler solution to person localization. West et al. (2005) have developed a ceiling-mounted camera model in a kitchen scenario to monitor the interaction between a person and cooking devices. The single ceiling-mounted camera can be calibrated easily or even be used without calibration at all. With a wide-angle lens, for example, a fish-eye lens, the ceiling-mounted camera could monitor an entire room. In particular, a ceiling-mounted camera has the advantage, being able to observe any position in the room and to track the target person without obstacles' occluding its view. The main issues of the single ceiling-mounted camera setup are (1) the limited raw information extracted from the camera and (2) the diverse body shapes of the person in the image (e.g., Figure 2.3). Moreover, the need for detecting a person in a realistic home environment without any modification makes this task harder. Sophisticated algorithms are essential, in this

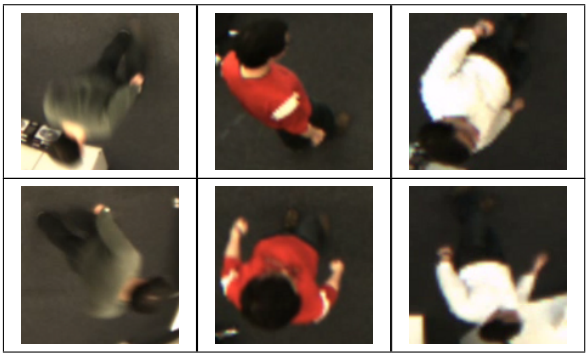


Figure 2.3: Person images from a ceiling-mounted camera
It is hard to recognize a person by a fixed shape pattern. Since the person may change his clothes in daily life, it would be hard to assign a color pattern for tracking.

case, to compensate for the drawback and to obtain a reliable tracking performance.

In general, single-camera-based person localization consists of two steps: detection, which finds the person using image processing methods, and tracking, which keeps the localization as good as possible by filtering out noise information (sometimes even when the detection is interrupted). Among the various person detection methods, the most common technique for detecting a moving person is background subtraction (Piccardi, 2004), which finds the person based on the difference between an input and a reference image. A moving object can be found perfectly well with this simple technique; however, there is a strict constraint in that the background has to be constant, because any change in the background (also

the light condition) will result in a difference and disturb the person tracking. Appearance-based models have been intensively researched in recent years; these detect target objects by analyzing their visual features. For example, principal component analysis (PCA) (Jolliffe, 2005) and independent component analysis (ICA) (Hyvärinen and Oja, 2000) represent the original data in a low-dimensional space by keeping major features. Through learning the major features, a pattern of the target object (e.g., face) is built that can detect new inputs whose features match the pattern. Some other methods such as scale-invariant feature transformation (SIFT) (Lowe, 2004) or speeded up robust features (SURF) (Bay et al., 2006) detect interest points (for example using Harris corners (Harris and Stephens, 1988)) for object detection, which have the advantage of scale- and rotation invariance and are able to detect similarities in different images. However, the computation complexity of these methods is high, and they perform poorly with non-rigid objects.

Vision-based methods using body part analysis (Frintrop et al., 2010; Hecht et al., 2009; Ramanan et al., 2007) can detect a person precisely by detecting the structure of the skeleton and the motion of a body, but these methods require a precise body shape captured from a front view that is not reliable in the real scenario. In order to improve the tracking quality, a multiple-camera system could be installed in a room environment to get the body shape correctly. However, the complexity of such a distributed detection system will be increased significantly. Color-based methods use color information obtained from clothes and skin as reliable tracking features (Comaniciu et al., 2000; Muñoz-Salinas et al., 2007;

Zivkovic and Krose, 2004). Other than shape detection that requires the form of the target to be constant, a color-based method has good generalization ability and can track a target person with an arbitrary pose. However, since the cloth color of the user cannot be predicted due to the daily change of the clothes, no color pattern exists at the beginning of tracking as the system does not know the cloth color, and it needs to be adapted when the color changes.

Probabilistic models are widely used to improve the tracking robustness and to fuse multiple sensor inputs or visual cues. As the person detection could be disturbed due to the image noise, the current state of him could be estimated based on the previous state and a dynamic model, and then corrected using the new observation. The probabilistic models could be used for different purposes, for example, localization, feature estimation, etc. For example, as a kind of discrete probabilistic model, particle filters are employed in a number of models (Canton-Ferrer et al., 2008; Isard and MacCormick, 2001; Klein et al., 2010; Nummiaro et al., 2003) for tracking single or multiple persons, and handling occlusion problem. It has been shown that a reliable tracking performance can be reached by combining probabilistic state estimation with vision detection techniques. Knoop et al. (2006) present a model for person detection and pose estimation. A human body model is represented as cylinders connecting with different types of joints. The pose of the person is estimated by the state that matches the visual inputs with the highest likelihood.

2.2 Robot Navigation

As a fundamental research topic, autonomous indoor robot navigation continues to be a challenge in unconstrained real-world environments. A robot needs to (1) represent the spatial knowledge of the environment, (2) determine its own position (and orientation), (3) find the target, and (4) navigate to the goal without injuring itself. This entire sequence of processes for reaching the target position is for a human also a complex cognitive task, because it involves various abilities such as perception, recognition, and behavior control. Although many models for map building and planning exist, it is still hard to integrate them synergistically due to the strong sensor noise, environment dynamics, and high complexity.

In general, robot navigation consists of three components: *Localization*, *Map building*, and *Path planning*. The localization determines the robot's position based on the map information, while the map building uses the robot's localization as well as sensor data to build an internal spatial representation. Because localization and map building are tightly related, these two parts usually are examined together with a special name as simultaneous localization and automatic mapping (SLAM). The sensor information for map building can be further categorized into two types: *idiothetic* information and *allothetic* information. Idiothetic information refers to the information from self-positioning, which estimates the change of the robot's state via internal representations such as speed, acceleration, or orientation of the robot. Allothetic information refers to the state estimation of the robot based on external information, for example, the visual perception of the environment, the sound information, etc. Although the idiothetic

information is easy to obtain, it is based on a robot-centered space, and the error of the sensor signal may be accumulated because of the lack of an absolute reference. For this reason, it cannot be used to represent the robot's position in a large-scale environment. In contrast, the allothetic information is extracted based on the environment perception, which does not only rely on the previous state but also the external information. One problem of the allothetic-based localization is that the quality of performance is influenced by the *perceptual aliasing* and *image variability* problems (Kuipers and Beeson, 2002), which refer to the fact that different positions may look the same, or the same position may look different. Consequently, both kinds of information are complementary for their disadvantages, and the fusion of both sources of information could improve the localization and map building significantly (Cox, 1991).

The path planning refers to the task of computing a sequence of actions to guide the robot to the goal based on the learned map, which is also called path-based navigation. Based on the map information, a trajectory is calculated where collisions with obstacles are considered and prevented. Typical methods are widely used for indoor and outdoor navigation, such as Voronoi diagrams (Aurenhammer, 1991) and visibility graph (Lozano-Pérez and Wesley, 1979). The drawback of the path-based navigation is that the robot can handle dynamic environments hardly based on the map information. The planned trajectory may be blocked due to a change of obstacles, and the robot will not be able to achieve the navigation when no more free path is available based on the old map information. Compared with this, a behavior-based navi-

gation controls the robot's motion in a more flexible way based on sensor feedbacks. In particular, when the environment frequently changes (for example with other moving objects), the robot can adapt its motion quickly to avoid collisions. The problem of behavior-based methods is that the performance of obstacle avoidance depends on the quality of sensors and the speed of the motion response. Particularly, as behavior-based methods do not calculate a global plan, the robot may not be able to reach any goal in a complex environment. For example, the frequent triggering of the reactive behavior for obstacle avoidance in a cluttered environment could disturb the normal navigation behavior.

Considering that humans and animals can learn their spatial environments and plan their path toward destinations easily, how information is processed in the brain is key to creating an intelligent robot (Burgess et al., 2002a). Therefore, the understanding of the biological principle of navigation is very appealing to researchers involved in the development of navigation systems for robots. For example, the discovery of place cells in the hippocampus of rats has been seen as essential for internal spatial representation. Place cells were discovered in 1971 by O'Keefe and Dostrovsky (1971), who saw that the activity rate of these neurons is strongly related to rats' location in the environment. A model based on spatial representation has a high robustness to sensor noise, good adaptation capabilities, and is highly efficient. With the help of these features, a robot could overcome challenges of indoor navigation such as high complexity of environment and possible dynamic changes. Because of these advantages, biologically-inspired navigation methods are attractive to service robotics, which support the

development of an assisted robot in an ambient assistant living (AAL) setup.

2.2.1 Map Building

Before a robot reaches a target, a representation of its environment is needed to determine its location and that of its goal. Although an accurate map is required for the robot localization, such a map is hard to obtain when the robot is in an unknown environment. Due to the lack of the a priori knowledge, the robot's location, as well as the structure of the environment can only be estimated. To build an accurate map becomes a challenging task, because the coordinate information of the environment can be partially computed based on the robot's current location estimation. The noise of the environment information will be added up during navigation, and the global mapping could be messed up easily. As accumulative error of the path integration (also known as dead reckoning (Reinstein and Hoffmann, 2013) that estimates the current position based on the previous position and the new observation) is hard to avoid, external information, e.g., from laser scanner (Guivant et al., 2002), sonar (Diosi et al., 2005), and vision (Karlsson et al., 2005) are often used to generate landmarks for environment representation. These landmarks will be stored and associated with the corresponding states on the map, which helps the robot to estimate its position when these landmarks are observed during the navigation of the robot.

The map representations of the environment can be split roughly into two groups: *metric maps* and *topological maps* (see Figure 2.4). The metric map represents the environment

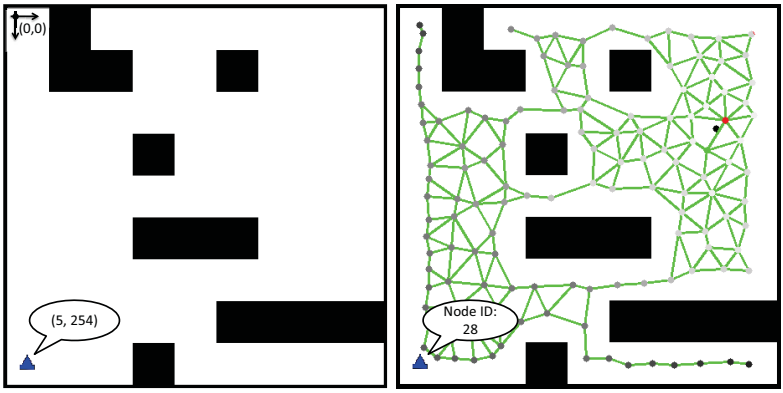


Figure 2.4: Schema of a metric map and topological map

Left: Metric map. The position of the robot is represented based on the coordinate information.
Right: Topological map. The position of the robot is represented by nodes of the map.

with coordinate information based on a reference system. In order to reduce the map complexity, the environment is usually discretized, and then such a resulting map is also called a grid map (Thrun, 2003). With the help of allothetic information from the internal sensors, the metric map can be built easily, and it can compute the optimized navigation path based on the map geometry. However, methods based on the metric map are inefficient for planning, and the required accurate positioning of the robot is hard to obtain.

In contrast, the topological map represents the environment as a set of nodes, which contain features of distinctive places, and a set of connections, which show the relations be-

tween different nodes (Thrun, 1998). Since the position of a robot can be determined based on the features stored in the nodes rather than using allothetic information, accumulative errors of the distance sensing will not disturb the environment representation. Therefore, robust robot localization can be achieved in a large-scale environment using the topological map. In addition, since the environment is discretized as a set of nodes and connections, the topology-based methods are efficient for planning and can be potentially combined with other cognitive tasks. The disadvantages of topological map-based methods are that they are difficult to use for representing details of a large-scale environment and localizing the robot exactly on the map due to the limit of the map resolution. Furthermore, since details (i.e., distance information) are missing in the topological map, the accurate maneuver behavior cannot be computed based on the topological map information alone.

Methods for Learning Metric Maps

Many methods have been proposed in recent years for solving the metric mapping problem (more precisely called the simultaneous localization and mapping (SLAM) problem) using the range sensors, and the most popular methods are based on probabilistic modeling (Bailey et al., 2006; Durrant-Whyte and Bailey, 2006; Montemerlo et al., 2002). Probabilistic SLAM represents the states of the robot via a probability distribution, as the state of the robot cannot be determined accurately. The location of the robot is not represented by a single position, but a set of states with corresponding probabilities. This representation is helpful to increase the system robustness, which

can avoid mislocalization due to defective sensor input. Filter techniques such as Kalman Filter (Bailey et al., 2006) and Particle Filter (Fox et al., 2001) are often integrated to estimate the states of the robot based on allothetic and idiothetic information. During the exploration, the state of the robot will be predicted via a transition model and corrected with the new observation from the robot sensors. Methods based on other techniques have been developed, for example, using Fuzzy Logic (Oriolo et al., 1998), which represents the uncertainty of the states of the robot with a fuzzy set. Moreover, artificial neural networks have been used to learn the environment. For example, Krose and Eecen (1994) presented a method that represents the environment using a self-organizing map, which learns the sensor input and identifies the robot's position based on the activity of the self-organizing map. Duckett and Nehmzow (1999) used a pre-trained multilayer perceptron network to detect unknown space based on a set of sensor signals and learn this area automatically. In order to compensate for the accumulative errors of the metric map-based methods during exploration, loop closing methods are used, which recover the map when a place is visited twice (Newman and Ho, 2005; Williams et al., 2009). Allothetic information is usually used for the relocalization because it is based on the environment's space and is independent of the robot's position representation.

Methods for Learning Topological Maps

Because the topological map represents the environment abstractly, landmarks rather than detailed position information are stored. Within these landmarks, visual features are com-

monly used for representing the observation of the environment where the corresponding nodes are built in the topological map (Gaspar et al., 2000; Matsumoto et al., 1996). A robot can determine its position on the map by finding the node whose visual features match the current visual input best. While visual landmarks have the advantage that they are based on the environment without having the accumulative error problem, they may not be detected well due to the different view angle and sensor noise. Hence, redundant landmark detection is often employed to improve the robustness. Various strategies can be used for robot positioning based on the topological map, such as winner-takes-all, which estimates the robot position by the best matching feature. Other methods use a probabilistic distribution of the possible robot position based on the feature matching. When features are observed from different positions, the robot's location will not be determined uniquely but be represented by multiple states in the topological map. This distributed representation is useful to improve the localization in the reality by estimating the robot's state. Moreover, as a topological map is a discrete map with nodes, the redundant state detection helps the robot to determine its position precisely between two states. However, since the position cannot be determined uniquely, multiple states have to be considered and a sophisticated planning method (e.g., a partially observable Markov decision model (Kaelbling et al., 1998)) is needed to integrate different states and make an efficient decision.

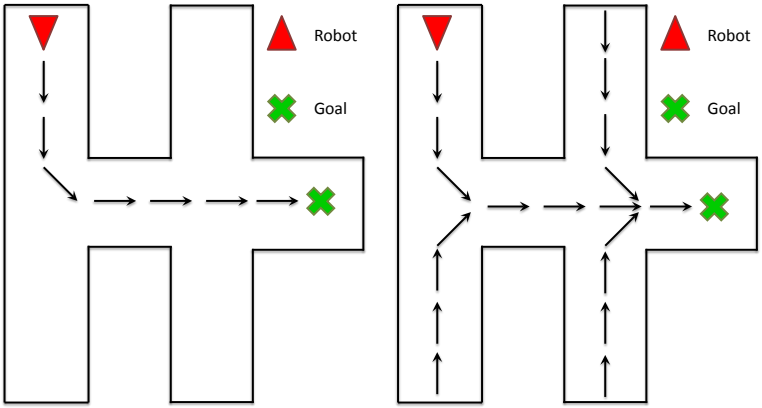


Figure 2.5: Schemata of path planning

Left: Single path planning. Only the trajectory from the current position to the goal position is considered.
Right: Universal path planning. All possible positions are considered.

2.2.2 Path Planning

Path planning refers to techniques of trajectory planning that enable robots to reach targets from a starting position based on an environment representation. A set of discrete movements is calculated, which guide a robot step by step in moving in its environment and helps it avoid obstacles. The planned trajectory is usually optimized, which allows the robot to navigate to the target using the shortest path. Choosing the right path is a fundamental ability for service robots to assist persons.

In general, robot path planning can be classified into two categories: *single path planning* and *universal path planning*

(Meyer and Filliat, 2003). For single path planning, a robot computes a path from its current position to the target, which consists of a sequence of movements to be executed. Once the route is planned, the robot will follow the path to reach its goal. As all the obstacles registered on the map have been considered and excluded from the path, the robot does not need to calculate a reaction to them, which makes navigation efficient. However, single path planning has the drawback that new computation is required when a change of the environment is detected. In particular, when the robot is in a dynamic environment (e.g., with other moving objects), the continuous regeneration of a path may lead to inefficient behavior or even to a navigation problem.

Compared with this, universal path planning is a more complex method, which builds up a motion plan for each possible position on the map to reach the goal. When a robot is located at one position, the motion associated with this position will be executed, which leads the robot to move to another position. Through continuous interaction with the environment, the robot repeats the localization and action phase until it reaches the goal position. Because multiple paths leading to the goal will be planned, the robot's motion can be not only deterministic, i.e., based on the best motion as done by the single path planning, but also through combining different motions. When the environment is changed, the corresponding motion plan can be adapted as well. This method could overcome the challenge of a dynamic environment that allows the robot to work with other moving objects (e.g., another robot) together.

Navigation planning has a high computational complex-

ity, particularly when the robot is in a large-scale environment. Therefore, methods to reduce the computational effort are important to ensure a real-time navigation capability. Many methods have been developed for this reason. For example, the free space in an environment can be represented as a topological map to realize the path-planning easily using graph-searching methods (Barr et al., 1981). The free space in an environment can be separated into subregions, which allow the system to extract a topological map from a metric map by labeling the subregions and their relations with topological nodes and connections. The free space can be decomposed exactly to cover it entirely, for example, using convex polygons (Latombe, 1990), or be represented in an approximate manner that covers it roughly with simple geometries (i.e., a grid map) to have a better planning efficiency (Thrun, 1998). Since the discretization error may influence the motion accuracy, methods for meshing the subregions with variable sizes have been presented to improve the map accuracy (Arleo et al., 1999). Furthermore, meshing with a hierarchical structure (Estrada et al., 2005; Lisien et al., 2003) has been proposed to build up accurate metric maps of the environment.

Another group of widely-used discretization techniques, which is known as roadmap methods (Latombe, 1990), compute a list of possible local paths on the map based on a set of keypoints of the environment. A global path can be generated by combining possible local paths to reach the goal from the start position using these methods. The path planning, in this case, is very efficient because all local paths have been pre-calculated, and no extra moving planning is needed here. Different concepts can be used to construct keypoints. For in-

stance, a visibility graph chooses the obstacle corners as keypoints and connects these keypoints with straight lines when they are visible to each other. Based on the target position as well as the robot's current position, it will calculate the trajectory by searching the graph. The advantage of the visibility graph is that the robot can plan and adjust its navigation effectively using an efficient graph search algorithm. However, the paths generated by this method are line-based and not smooth, which may cause turning difficulty when the orientation towards the next node changes a lot. Another well-known method, the Voronoi diagrams (Aurenhammer, 1991) tackles this problem by maximizing the distance between paths from obstacles to avoid possible collisions.

An efficient path searching algorithm plays another vital role to achieve real-time navigation planning. For example, using an A^* algorithm (Barr et al., 1981), a robot can calculate a path from its current position to the goal quickly by filtering out bad options using a knowledge-plus-heuristic evaluation. Reward spreading (Murphy, 2000; Yan et al., 2012b) can be used for universal planning that spreads out a reward value from the goal position iteratively with a discount factor. A robot always moves towards the neighborhood node with the highest reward value during navigation until reaching the goal position. Dynamic programming (Bertsekas et al., 1995) is a useful strategy for robot navigation in a large-scale environment. A large environment is split into several subregions with this method to reduce the planning complexity. Local optimal paths of subregions are calculated and combined to generate the global path of navigation.

Due to the discretization error of the space representation,

actions calculated between two adjacent states may not match the environment well. If the robot is controlled deterministically by following the action signal accurately, this discontinuity of action signals may lead to collision with obstacles or difficulty of maneuver, because the robot may not be able to respond so quickly due to the motion inertia. Additionally, as the robot may not perceive the environment precisely due to the sensor noise, a robot may behave incorrectly based on the wrong localization. For these reasons, deterministic action planning may not be suitable for navigation in a real environment, and a probabilistic method (Smith and Simmons, 2012) could help a robot to improve the robustness of navigation by estimating its state with a redundant representation. The state of the robot will not be represented by a single node, but with a probabilistic distribution over the map based on the current and the previous position. Then, based on the current state distribution, multiple actions will be triggered, and smooth motion will be computed by combining corresponding actions. However, the main disadvantage of the probabilistic model is its high computational complexity, which requires a costly hardware.

2.3 Interaction with the Environment

As a service robot aims at assisting the daily life of persons, it needs to work in reality and face the challenge of the real world. One typical problem is that rather than in a simulated environment where the robot can plan its route purely based on the map, a robot shares its environment with persons, pets or possibly other robots. The obstacle avoidance in cluttered

environments, in particular, with other moving objects, becomes a crucial topic for robot navigation.

Interaction with the environment is a tough task for a robot. As the environment changes are unpredictable, the robot needs to detect them through interaction with the environment during navigation and reorganize its path according to the change of the environment. This detection-and-reaction loop increases the complexity of navigation planning significantly. On the other hand, the recalculation must be fast, especially with other moving objects, to ensure that the robot can always respond correctly to the change in real time to avoid injuring persons or damaging itself. The interaction also needs to be intelligent. Since the interaction with the environment needed by the robot to detect and response to changes is time-consuming, a robot should be able to update its map through detecting a temporary obstacle and a structure change of the environment.

In general, there are two strategies of navigation around obstacles (see Figure 2.6): *path-based* approaches, in which collisions are prevented through global navigation planning based on the known environment, and *behavior-based* approaches, where a robot avoids obstacles based on reactive behavior using sensor inputs. During behavior-based navigation, a rough path (e.g., using the topological map) without detailed maneuver control is planned that guides the robot to move towards the target position. Then, the robot interacts with the environment based on its real-time environment perception and adjusts its motion steadily to avoid obstacles. As this interaction-based control is similar to human and animal behavior in that people do not calculate their motions exactly,

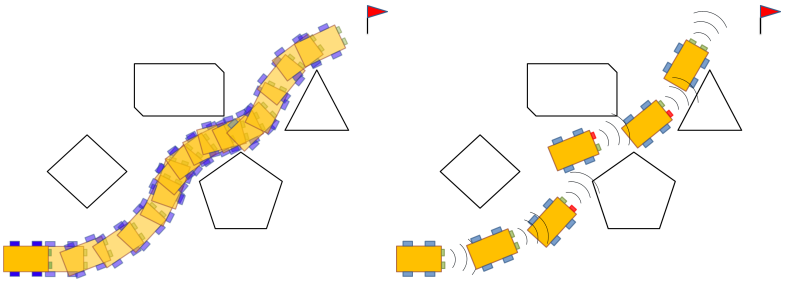


Figure 2.6: Schemata of obstacle avoidance during navigation towards the goal

Left: Path-based approach. Global navigation planning is computed to avoid collisions based on the map information.
Right: Behavior-based approach. The robot adapts its behavior based on sensor inputs during navigation

behavior-based navigation is more biologically plausible and requires far less computation for position control. As methods of path-based navigation have been briefly described in the previous chapter (see Section 2.2.2) and in this section, we will focus on behavior-based methods.

Behavior-based navigation systems with diverse concepts have been developed in recent decades. Early research models navigation behaviors with a set of preprogrammed rules, which are triggered according to sensor information (Brooks, 1986; Connell, 1990). In order to reduce complexity, different statuses are defined to which the sensor data clusters. The drawbacks of these methods are the inflexibility and the incompliance of the state representation. All the statuses have to be defined manually, which may not cover all the possible situa-

tions. A multi-layer control architecture is built to decompose a complicated cognitive task into a list of subtasks, which is sorted from abstract, complex (high-layer) to basic (low-layer) behavior. With the help of this architecture, a robot can react quickly based on the low-layer behaviors and execute complex tasks through the high-layer behaviors. Seraji and Howard (2002) developed a method based on fuzzy logic which assesses the environment information with a set of fuzzy rules. Based on this information, navigation behaviors are activated and integrated based on the corresponding behavior weights, which generates a more general and smooth motion. Neural networks (Gavrilov and Lee, 2007; Pomerleau, 1991) are another option for robot control in which the response behavior can be learned automatically using supervised learning. For example, Echo State Networks (Hartland and Bredeche, 2007) are employed to adjust the navigation behavior based on not only the current sensor value but also the history, which helps the robot to identify the obstacles more precisely.

As one of the advantages of behavior-based approaches, the robot can detect and handle unknown obstacles based on the feedback of the sensors, which is appropriate for dealing with environment dynamics. However, one problem is that it may perform poorly within a complex room since too many reactive behaviors will be triggered, which will prevent the robot from navigating to the target successfully. The quality of sensors (e.g., detection range, accuracy, etc.) and the speed of the motion response may influence the navigation behavior strongly. Moreover, the strength of the behavior-based approaches is also their weakness: as the accurate environment information (e.g., distance) is not considered during path

planning, these navigation methods may result in suboptimal solutions (Borenstein and Koren, 1991). Compared with this, path-based methods may optimize the navigation trajectory (see Section 2.2.2), but they need a complete representation of the environment. That is hard to maintain when the environment changes. Adaptive learning is necessary in this case, and hybrid architecture promises to be a nice solution that combines path-planning and behavior-based control in a complementary manner.

With the information acquired from the behavior control (e.g., obstacle avoidance), the robot can adapt its internal environment representation and avoid obstacles' positions proactively in the planning phase. In addition, from the biological point of view, as the dorsolateral striatum generates automatic behavior, and the dorsal striatum coordinates the goal-directed behavior (Balleine et al., 2007), these co-existing mechanisms are important for animals to produce flexible navigation behaviors. Many computational models based on adaptive learning have been presented. For example, Donnart and Meyer (1996) present a control architecture, MonaLysa, which consists of a planning module that generates sequences of actions for reaching goals and a reactive module for avoiding dangerous situations. Salient states are extracted during the navigation, which help the robot remember the obstacles and avoid them when repeating the same navigation task (i.e., from the same starting position to the same goal). Based on Hidden Markov models, Bennewitz (2004) develops methods for adaptive learning of robot navigation in a dynamic environment populated by humans and robots. Knudson and Tumer (2011) approach a neuro-evolutionary algorithm for robot nav-

igation in a cluttered environment. Combinations of behaviors are generated in the redundant neural network system and, through a cost function, optimized automatically based on evaluations of the results of their actions.

Because assistive robots in homes must be efficient in serving users and avoiding collisions, they should perform path planning and reactive behavior calculations at the same time. For this reason, we developed a hybrid control system that integrates synergistically a reflex-like behavior for obstacle avoidance with plan-based navigation (Yan et al., 2013). A spatial representation of the environment is built, which allows the robot quickly to adapt the navigation path towards a target during walking and guides the robot by showing the direction in which the movements have to be made. In addition, a reactive model evaluates input from sensors to control the robot's actions and avoid (moving) obstacles. By analyzing sensors' feedback, the system detects permanent changes in the environment, and the corresponding spatial memory weights of state transitions are adapted, which adjusts the decision-making during navigation. Details of the system's architecture will be described in the following chapters.

2.4 Neural Mechanism of Spatial Cognition

Human-like intelligent navigation ability is an ultimate goal of robot navigation. While the robot navigation methods are still based on a very detailed environment representation and confront the challenge of computational complexity, a human can easily abstract the environment with significant features,

find the way to the destination and react to unplanned situations spontaneously. In order to rebuild these functionalities in a robot, it is necessary to understand how position information is processed and stored in the brain. In this section, we will briefly review the neural mechanism of spatial cognition in the brain, which provides a theoretical basis for spatial learning and spatial cognition. Computational models based on these theories, in particular, artificial neural network models, will be introduced, which enable a robot to learn the spatial information and realize autonomous decision-making.

Spatial cognition refers to humans' and animals' ability to gather information about the environment, organizing and using the spatial knowledge, and revising it when the environment changes (Montello, 2001). It is a fundamental ability of humans, which helps us to achieve different tasks like navigation and grasping. As proposed as a "cognitive map" by Tolman (1948), the spatial information of an environment is presented with features and relationships in the cognitive map, which enables an animal to navigate flexibly based on the abstract information in the cognitive map and to make detours by adjusting the map structure when an obstacle appears. The spatial information is represented in an abstract fashion because usually a person does not measure distances to the environment but describes the surrounding features (even with just a few symbols). On the other side, the location is encoded redundantly in a set of features, which helps the person to locate his position even when the environment has changed a lot.

Recent advances in neuroscience provide insight into the neural mechanisms of spatial cognition in humans and ani-

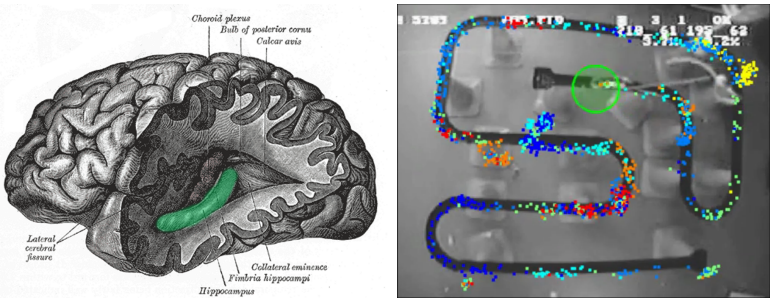


Figure 2.7: Hippocampus and place cells

Left: The position and shape of the hippocampus in the brain[1].
Right: Hippocampal neurons' activity at distinct place fields while a rat is moving in a maze[2]. The colors of dots index neurons that are active when the rat visits different areas, and the current position of the rat is labeled with the green circle.

mals (Burgess et al., 2002b; Maguire et al., 1998). An important finding is the place cell in the hippocampus of rats by O'Keefe and Dostrovsky (1971) and Andersen et al. (2006). Their activity rate is strongly related to the rat's location in the environment. Later, head direction cells were found in the rat's brain, first in the postsubicular cortex of the hippocampus (Taube et al., 1990), then in other related brain areas, e.g., lateral mammillary nucleus, posterior cortex, lateral dorsal nucleus and ventral striatum (Blair et al., 1998; Chen et al., 1994;

[1]Source: Image from Gray's Anatomy. `http://www.bartleby.com/107/illus739.html`

[2]Source: Snapshot from a video by Dr. Fabian Kloosterman. `http://kloostermanlab.org/research/spatial-code/`

Cho and Sharp, 2001; Lavoie and Mizumori, 1994; Mizumori and Williams, 1993; Taube, 1995). These cells fire selectively when the rat faces a specific orientation, and provide a signal of the rat's heading direction during navigation (Pennartz et al., 2011). Moreover, anticipatory head direction signals are found in the anterior thalamus (Blair and Sharp, 1995; Taube and Muller, 1998). Another important discovery is the grid cell in the entorhinal cortex (Hafting et al., 2005), which fire with a periodic hexagonal pattern in enclosed spaces. Together, these cells constitute a coordinate system that provides a representation of the location based on the animal's internal position sense in the environment and forms the basis of a cognitive map.

A cognitive map can be represented in different ways; for example, as a topological map (Cuperlier et al., 2007; Martinet et al., 2011), a continuous attractor network (Milford et al., 2004; Samsonovich and McNaughton, 1997; Samsonovich and Ascoli, 2005), etc. Toussaint (2006) developed a model using a self-growing mechanism (Fritzke, 1995) that forms a map with a dynamic size, which is flexible for exploring an unknown environment. However, this model does not provide an efficient way of handling (moving) obstacles in a real world. The RATSLAM model developed by Milford and Wyeth (2010) provides a nature-inspired way for mapping, which represents the spatial information in its pose cells by combining the internal sensing and the external visual perception. However, the experience map in RATSLAM builds mainly line-like trajectories in space rather than mesh-like representations of space due to a strict rule to connect cells, which constrains the generation of flexible navigation.

To acquire robust robot navigation, Weiller et al. (2010) proposed an unsupervised learning method to learn navigation behavior associated with state transitions automatically and control the robot during navigation by selecting the action with the highest value. Weber and Triesch (2008) and Witkowski (2007) present neural network models that learn associations between adjoining states and actions that link them. In addition, closed-loop control models for other behaviors, e.g., arm reaching, apply similar methods of planning (Herbort et al., 2010). A drawback of these models is that the representation of the state space is hardwired, which means that they only work in a well-defined environment. The action model in these models is discrete, and the robot is controlled by a winner neuron's action signal. Hence, the executed action might not be accurate in the continuous real world because of the discretization error, or a fine-meshed action space would be required, which increases the learning effort strongly.

Given a population code for state estimation, there will be different actions suggested by the network for the robot to take. As the robot needs to be controlled by a concrete motion signal, a synergistic method is required to generate motion behavior based on the multiple inputs. In addition, the method should also consider the system dynamics of the robot in order to provide a smooth motion control and avoid sudden changes of the movement, which requires the system to perceive the actual action inputs, memorize the previous actions, and coordinate them properly for decision making.

Based on the current neural finding and understanding of the working principle of neurons in the brain, a neural field theory (Wilson and Cowan, 1973) has been drawing more at-

tentions in cognitive science and robotics. For example, Erlhagen and Bicho (2006) achieved a goal-directed robot navigation behavior with real-time obstacle avoidance with dynamic neural fields. Because of the distributed information encoding, i.e., neural population coding, the dynamic neural fields can generate stable signals by updating the activation patterns, while also canceling noise. Therefore, the dynamic neural field model has been seen as a simple but effective way to model motion perception (Giese, 1999), and has been widely used for robot control because of its distributed representation and the dynamic integration of information (Cuperlier et al., 2005; Torta et al., 2011a; Toussaint, 2006).

2.5 Summary

In this chapter, we reviewed the state-of-the-art localization and navigation technologies for socially assistive robots that allow robots to detect and to approach people in a real environment. Related backgrounds have been introduced, such as the person/robot localization, robot navigation, and robot-environment interaction. Due to the high complexity of a real-life environment, a robot needs to realize intelligent autonomous behaviors based on its limited sensor signals. Although these behaviors are hard to be computed, we consider that they can be achieved by human beings and animals, and a model that emulates human's behavior, or information processing could be useful. Therefore, we summarized the neural principle of spatial cognition and computational models of them, which are helpful for robotics to generate intelligent human-like behavior. These approaches are an important ba-

sis for further development and based on them, we will present our methods of person and robot localization and navigation in an ambient intelligent environment. We will describe details of our architecture in the following chapters.

Chapter 3

Indoor Person and Robot Localization

Despite numerous approaches, indoor (person) localization is still a challenging task in a real, cluttered home environment, which needs to cover all different situations in daily life. Most tracking methods are reliable under a certain condition. For example, a motion detector based on background subtraction is a good tracking indicator by finding movement information, which is typically provided by the person's motion. On the contrary, when a person does not move when he or she is sitting on a sofa, no motion information will be detected. In this case, the motion detector cannot localize the person correctly. On the other hand, the motion detector may be distracted by the change of light conditions or image noise because the difference of background appearance can be recognized as motion information. The color obtained from the clothes and skin can be a reliable tracking feature. However, the color

pattern needs to be frequently adapted in a real-life scenario. For example, when a person changes his clothes, the old color information will be irrelevant, and we have to learn the color pattern again. A feature-based recognition method may not be appropriate for real scenarios due to the lack of the pattern information (e.g., a face may not be recognized when only part of it is observed). Other features such as clothing color have to be frequently adapted when the person changes his or her clothes, which makes it unrealistic for daily usage. Body shape or skeleton-based techniques could provide robust person detection, but both require the camera to be placed in front of the user. A multiple-camera array is therefore needed to cover the entire space, which increases the complexity of the system. On the contrary, a ceiling-mounted camera can observe the whole environment, but people's shape varies when they are at different positions in the room.

We dedicated ourselves to developing a general model that can cover as many situations in an environment as possible to achieve reliable indoor person and robot tracking. As the model developed by Triesch and Malsburg (2001) showed that the integration of different sources of information can cope with complex perceptual tasks, which is supported by biological evidence (Murphy, 1996), we consider that multiple visual information sources can be used in combination to reliably detect and localize a person's position in diverse situations. Hence, a hybrid probabilistic neural architecture for multiple visual cue integration is developed where each visual cue processes the image data separately and returns a detection value, which is combined using a majority voting scheme based on an adaptive Sigma-Pi network (Weber and Wermter, 2007). By

combining the results of different visual cues, the target can be detected reliably under certain conditions. Diverse environment situations could be covered, which results in an overall good tracking performance. For example, our approach can track a person with or without motion information and is robust against environment noise such as moving furniture, changing light conditions, and interactions with other persons. The target person can be learned through the adaptivity of the cues, which acts as a memory and helps the system track the person robustly.

In order to improve the tracking robustness, we consider probabilistic filter techniques (Khan and Shah, 2009; Lanz and Brunelli, 2008; Qian et al., 2007; Smith et al., 2005) and use particle filters to model the person's and the robot's movement, as well as to filter out noise information. Our localization model uses a ceiling-mounted camera to locate the target person and the robot. To cover the entire room, a fish-eye lens without calibration is applied to obtain a wide-range view, which simplifies the system installation. The system should detect a person under various conditions, i.e. while a person is walking, sitting, or lying, and be robust against image noise. Although the system is designed initially for tracking a person when (s)he is alone in the room, it should avoid mislocalization when a distracting person is in the room as well. In addition, the system should distinguish the robot and the target person when the robot is navigating to the person and estimate the orientation of the robot because it is important for the robot's control. Since the pose and color (e.g., the clothes' color) of the person cannot be estimated, it increases the difficulty of detecting the person. Details of the system architecture and

the models will be described in the following sections.

3.1 General Approach

The general approach of our model is illustrated in Figure 3.1. Based on the input from the ceiling-mounted camera, the visual information is processed by the following three models: (1) a probabilistic model using a particle filter for tracking and decision-making, (2) a set of visual cues that detect a target person based on different visual features and (3) an adaptive polynomial cue combination architecture, which uses a Sigma-Pi network to update these cues and their reliabilities during tracking.

The person's position is estimated using the distribution of particles based on system dynamics and the current observations using results of the multiple cues. Each particle contains coordinate information: for a person tracking, it is the coordinate information x, y of the center of a bounding box (with respect to the image frame), and for the robot localization, the orientation information o is also included. Image patches around particles are selected and processed using the image input of the camera, whose results are used to estimate the position of the target and update the tracking system at the same time.

The entire process of particle filtering is split into two phases: *prediction* and *adaptation*. In the prediction phase (black arrows in Figure 3.1), the particle filter estimates the state of the target object, here the position of the target person based on the current state. A bounding box is defined around each of the particles with a constant width and height.

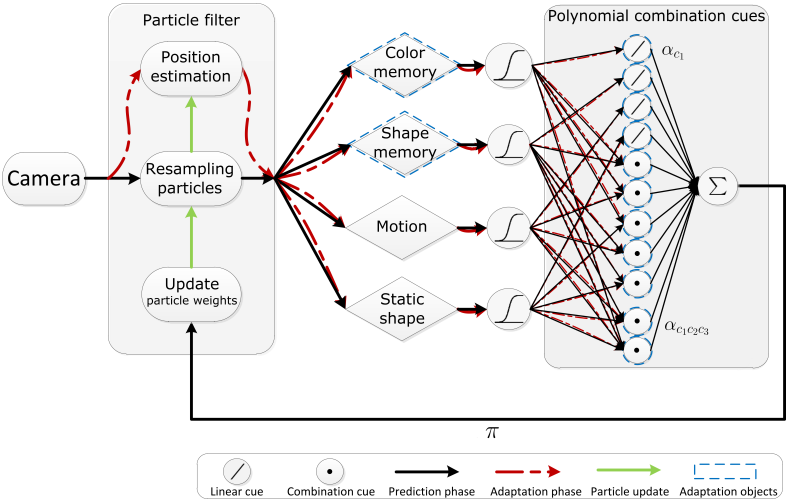


Figure 3.1: Architecture of the localization model.

The image input is segmented with these bounding boxes, and the image patches are processed with visual cues. As only these images' patches around particles are processed instead of the entire image, the computation is simplified significantly, which helps the system to obtain real-time ability. Values of the visual cues are further processed with the polynomial combinations, and the weights of the particles are changed and the particles are re-sampled (see green arrows). Based on the change of the particles' distribution, the position estimation of the person is updated.

The adaptation phase works similar to the prediction phase. However, the main task is to validate the tracking performance, learn the visual features and adapt the correspond-

ing reliabilities of them. After the prediction phase, an image patch around the estimated position is segmented and processed by the visual cues (see red dashed arrows in Figure 3.1). According to the output of these cues, the results of the polynomial combination cues are calculated. When the tracking results of cues agree with each other, results of the corresponding combination cues increase. Then, reliabilities of visual cues will be adapted using these findings. For example, when the target is detected by the motion and color information at the same time, the reliabilities of both cues will be increased. When the sum of the reliabilities is higher than a threshold, the memory of these cues will be updated (labeled with the blue dashed lines). This is important for visual cues that need to be learned at first, because after initialization, these cues cannot provide useful information at the beginning, and the reliabilities indicate how well these features are learned.

The polynomial combinations use a high-order feed-forward neural network (a sigma-pi network) to compute the correlation of visual cues, which consists of not only a linear combination, but also uses high-order multiplication to generate virtual cues between different inputs to enhance the likelihood when several cues coincide with their detection. Each cue extracts specific visual information based on computer vision, and the activities of visual cues are generated via activation functions and scaled by their connection weights, which are called reliabilities (see α_{c_1} and $\alpha_{c_1c_2c_3}$ in Figure 3.1). Because no shape and color pattern exist at the beginning of tracking, shape memory and color memory cues are gathered during tracking and an adaptive learning mechanism is used to ad-

just the reliability of visual cues. The reliabilities of these cues are low at the beginning of tracking, which means they do not contribute to the decision-making, and their values increase when the features of the target object are learned. The output of the Sigma-Pi network is passed to the particle filter. With the collaborative contribution of each cue, the tracking performance can be improved significantly.

Multiple visual cues produce detection results using individual techniques. These visual cues can be applied where each of them focuses on the person detection based on specific features (e.g., motion, color, shape, etc.), and together they support the person detection in various situations. This process provides a flexible way of information fusion in a synergistic manner, which is useful to cover different environment situations and improve the tracking robustness by ensuring that parts of feature cues work properly while the others are distracted. In our model, we first use the following cues for development the person localization model, while other visual cues can be integrated easily using the same principle:

- Motion detection

  As a basic but efficient method, the motion detection is robust to find an arbitrary moving target without the need for training. However, it cannot provide information when the person is not moving and is sensitive to noise (e.g., blinking of a television). This method can also be distracted when the environment (background) is changed because the motion information is detected by comparing the actual image with a reference image. Therefore, a dynamic lighting condition could distract the tracking result significantly.

- Color memory

  The clothes and skin's color of the target is a feature that contains rich information, which is independent of the environment. A single color or a distribution of multiple colors (e.g., using a histogram backprojection) can be used for tracking the skin's/clothes' color of the user. Compared with the motion information, color is more stable under various lighting conditions and is irrelevant to the form of the object, which is helpful to track an object or a person with a strong change in form. The main problem of the color information with respect to our problem is how to obtain the color information at first. Because the user may change clothes daily, the color information of the person needs to be learned before tracking. Hence, an automatic learning process is required to update the color memory.

- Static shape recognition

  Shape recognition has the advantage that it is irrelevant to the color and intensity information that helps it to be very robust under different environment conditions. Some methods, such as the Hu moments extracted from the image provide a shift, rotation and scale invariant property, which helps the system to obtain a better generalization ability to detect a person with a pose that has not been stored. One problem is that the classifier of the shape recognition has to be trained decently before using these features. For this reason, in our work we use a static shape recognition model based on a multilayer perceptron network to detect the shape of the person. The

shape information is stored in the neural network after using supervised learning, and the detection is processed based on the output of the network. The main drawback of the static shape recognition is that a large data set is required for training, and the recognition may be failed when the form of the person differs from the training data significantly.

- Short-term shape memory

  A short-term shape memory is robust against changes of environment conditions. Compared with the static shape recognition, it does not need the intensive training phase and can detect the target simultaneously by finding the similarity of the current pose of the target with the stored features. However, similar to the color memory, as the first pattern of the shape has to be trained before tracking, an online learning method is needed. A temporary shape memory cue is built based on SURF features. During person tracking, features of the current tracking object are memorized, which evaluate the shape consistency over a small-time window.

We will describe the details of particle filter in section 3.2, the details of visual cues will be presented in section 3.3 and the sigma-pi network in section 3.4.

3.2 Particle Filter

The particle filter, also known as sequential Monte Carlo method, is an approximation method for representing probabilistic distributions with a density estimation based on Bayesian rules

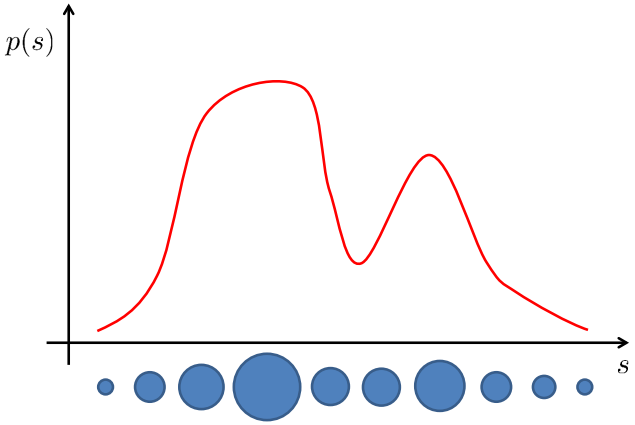


Figure 3.2: Visualization of particle filter weights
The radius of a circle denotes the particle's weight.

(Fox et al., 2001). The method was firstly applied in control systems and was proven that it is an efficient filter/estimation technique without the need of an assumption of the state transition as well as the noise model (Gordon et al., 1993). Nowadays, the particle filter becomes an important method in robotics, and is widely used for noise canceling, state estimation and decision-making (Lanz and Brunelli, 2008; Qian et al., 2007; Smith et al., 2005). In our work, we applied particle filters to represent the location of the target.

3.2.1 Background of the Particle Filter Algorithm

The particle filter is based on the Bayesian filter, which estimates a posterior probability based on the a prior probability and the current observation. Unlike the Kalman filter that

has a form of Gaussian distribution, the particle filter aims at dealing with arbitrary distributions for a set of samples, i.e. particles and estimate a density function. Assume that $s_{0:t} = \{s_0, s_1, \ldots, s_t\}$ is the time sequence of an agent's state that is described by dynamic equations:

$$s_t = g_t(s_{t-1}, u_t) \tag{3.1}$$

where u_t denotes noise signals of the state transition, and $z_{0:t} = \{z_1, z_2, \ldots, z_t\}$ is a time sequence of the corresponding observations of the agent which is described by:

$$z_t = h_t(s_t, v_t) \tag{3.2}$$

where v_t denote noise signals of the observation. Then, the posterior of the agent's state can be estimated as follows according to the Bayesian filter (Fox et al., 2001):

$$P(s_t|z_{0:t}) = \eta P(z_t|s_t) \int P(s_{t-1}|z_{0:t-1}) P(s_t|s_{t-1}) \mathrm{d}s_{t-1} \tag{3.3}$$

where $P(s_t|s_{t-1})$ denotes the probability of the state transition from s_{t-1} to s_t at time t, $P(z_t|s_t)$ is the observation model, and $P(s_t|z_{0:t})$ is the probability of a state given all previous observations from time 0 to t. The η is a normalization constant. Because $P(s_t|z_{0:t})$ describes "what the state looks like", it is also called the belief of the state.

In a discrete computing model, the belief of the state s_t at time t under the observation $z_{0:t}$ can be computed according to the previous distribution $P(s_{t-1}|z_{0:t-1})$:

$$P(s_t|z_{0:t}) \approx \eta P(z_t|s_t) \sum_i \pi_{t-1}^{(i)} P(s_t|s_{t-1}^{(i)}) \tag{3.4}$$

where the probability distribution of the states is represented by a set of particles $\{i\}$, which contain the state information. Because the beliefs of the states are expressed by corresponding weight values $\pi^{(i)}$, the probability distribution can be approximated in the following form:

$$P(s_t|z_{0:t}) \approx \sum_i \pi_{t-1}^{(i)} \delta(s_t - s_{t-1}^{(i)}) \tag{3.5}$$

where π denotes the weight factor of each particle with $\sum \pi = 1$ and δ denotes the Dirac impulse function. As shown in Figure 3.2, a distribution can be represented by the state of particles and their weights. The higher the weight value, the more important this particle is in the entire distribution. The mean value of the distribution is computed as $\sum_i \pi_{t-1}^{(i)} s_t$ and can be used to estimate the state of the agent, if the distribution is unimodal.

Different particle filter models have been proposed (Doucet et al., 2000; Gordon et al., 1993; Green, 1995), among them the sequential importance resampling (SIR) algorithm has been widely used. The algorithm of SIR shown in Algorithm 1, which consists of three phases: prediction, update, and resampling. The iteration steps of the SIR algorithm are illustrated in Figure 3.3. In the prediction phase, the SIR first predicts the current probability distribution of the agent based on the previous state and the transition. The states of particles are updated according to the transition model and the action, and a new distribution of particles is computed. After the prediction, the new distribution is validated in the update phase based on the new observation and particle weights are adapted using Bayesian filter.

Algorithm 1 Sequential Importance Resampling (SIR)

Draw samples for N particles from the proposal distribution q:

$$s_t^{(i)} \sim q(s_t | s_{0:t-1}^{(i)}, z_{0:t})$$

where $s_t^{(i)}$ denotes the state of the particle i.
For all particles from 0 to N, update their importance weights $\pi_t^{(i)}$:

$$\pi_t^{(i)} = \pi_{t-1}^{(i)} P(z_t | s_t^{(i)})$$

Normalize the importance weights $\pi_t^{(i)}$:

$$\pi_t^{(i)} = \frac{\pi_t^{(i)}}{\sum_j \pi_t^{(j)}}$$

Compute the effective number of particles:

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\pi_t^{(i)}\right)^2}$$

If $\hat{N}_{\text{eff}}$ is less than a threshold, resample the particles with the probabilities proportional to their weights and reset the weight values:

$$\begin{aligned} s_t^{(i)} &\propto \pi_t^{(i)} \\ \pi_t^{(i)} &= \frac{1}{N}, \qquad \text{for} \quad i = 1 \dots N \end{aligned}$$

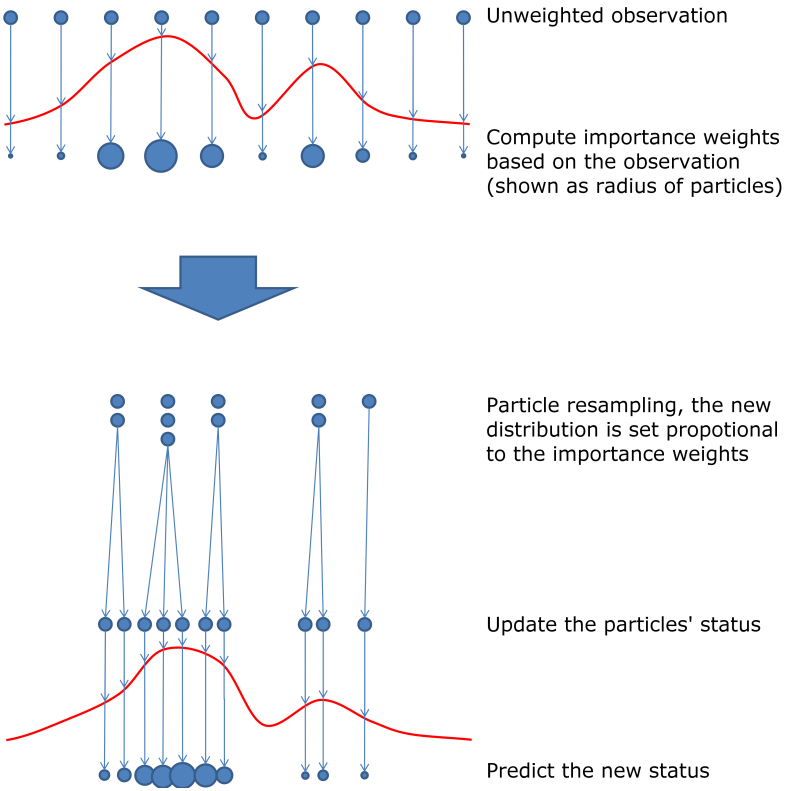


Figure 3.3: Schema of the particle filter update

The particle resampling is a useful technique to avoid the degeneration problem, which means that particle weights converge potentially to a single state. In this case, most of the particles have low values that cannot represent the state space correctly. During resampling, a number N_{eff} is calculated to evaluate how many particles are effective. When N_{eff} is small, the resampling process will be triggered that replaces particles according to the state's weights and averages the particles' weights. More particles will be moved to the area with higher value, which helps to represent the distribution more accurately because the low-value areas do not contain useful information. The concrete system modeling will be described in the next section.

3.2.2 Using Particle Filter for Person and Robot Localization

Particle filters are used in our work for an indoor person and robot tracking based on a camera view. Compared with other state estimation techniques, such as a Kalman filter, particle filters can represent a more generic distribution and accelerate the state estimation significantly by using random sampling. Each particle contains position information. In the person tracking system, the person's state is represented by the x- and y- coordinates in the image, i.e. $s_{\text{p}} = \{x, y\}$. The direction of a person's motion is hard to predict, because, for example, an arm movement during rest could be wrongly perceived as a body movement into the corresponding direction. Hence, we do not use the direction of movement information, but describe the transition model $P(s_{\text{p},t}|s^{(i)}_{\text{p},t-1}, a_{\text{p},t-1})$ of the person with

a Gaussian distribution:

$$P(s_{\mathrm{p},t}|s^{(i)}_{\mathrm{p},t-1}, a_{\mathrm{p},t-1}) = \frac{1}{\sqrt{2\pi\sigma(a_{\mathrm{p}})^2}} e^{-\frac{(s^{(i)}_{\mathrm{p},t-1} - s^{(i)}_{\mathrm{p},t})^2}{2\sigma(a_{\mathrm{p}})^2}} \tag{3.6}$$

where $\sigma(a_{\mathrm{p}})^2$ is the variance, $s^{(i)}_{\mathrm{p},t-1}$ are the previous states, $s^{(i)}_{\mathrm{p},t}$ are the current states and $a_{\mathrm{p},t-1}$ is the executed action. Movement information from the motion cue (see section 3.3) in the action variable $a_{\mathrm{p},t}$, however, is informative for the person's movement distribution that we account for by increasing $\sigma(a_p)$ when motion is detected. The $\sigma(a_{\mathrm{p}})$ is then set to either of two values:

$$\sigma(a_{\mathrm{p}}) = \begin{cases} v_1 & \text{if motion detected} \\ v_2 & \text{else} \end{cases} \tag{3.7}$$

where v_1, v_2 are constant parameters with $v_1 > v_2$. When there is no motion detected, the probabilistic distribution will shrink to a small area that allows the particles only to move close around the previous position. This method affects the behavior in a way that when an object is identified, a human would remember its position when the object does not move.

In contrast, since the robot's motion can be estimated based on the given motion command, we define the transition model of the robot according to the robot's motion signal. For this reason, the robot's state consists of a three-dimensional feature vector: the x, y position and the orientation θ, i.e. $s_{\mathrm{r}} = \{x, y, \theta\}$. While the robot walks, the state of the robot is first predicted with a dynamic model:

$$s^{(i)'}_{\mathrm{r},t} = g(s^{(i)}_{\mathrm{r},t-1}, a_{\mathrm{r},t}) \tag{3.8}$$

The new position prediction is calculated according to the given rotation and the walking speed command using triangle calculation. Then, the probability distribution is computed with a Gaussian distribution:

$$P(s_{\mathrm{r},t}|s^{(i)}_{\mathrm{r},t-1}, a_{\mathrm{r},t-1}) = \frac{1}{\sqrt{2\pi\sigma(a_{\mathrm{r}})^2}} e^{-\frac{(s^{(i)\prime}_{\mathrm{r},t} - s^{(i)}_{\mathrm{r},t})^2}{2\sigma(a_{\mathrm{r}})^2}} \tag{3.9}$$

Similar to the person's particle, the variance of the Gaussian is controlled by the motion command of the robot. For example, when the robot turns quickly, the variance of the orientation will be larger which results in a higher uncertainty of the orientation.

The particles are initialized randomly in the image at the beginning of the tracking. Then, image patches around particles, which could cover the view of the robot and the person, are taken and processed with visual cues to detect the target person/robot. This strategy accelerates the system compared with traditional pixel-wise search window methods, since the search space now is proportional to the number of particles. The particle weights are computed based on the output of the visual cues using the Sigma-Pi network. Where the sum of weighted cues returns a large value, the particles will get larger values and increase the probability of them in the distribution, which shows that a person is more likely to be in this position. The position of the person/robot is finally estimated as the mean of the distribution. In order to keep the network exploring, we take randomly 5% particles with low weights and replace them to random positions at each step to search actively for a possible position of a person.

3.3 Visual Cues

Compared with some other range sensors, a vision system uses camera sensors that provide rich information through a sequence of images. By analyzing with different algorithms, visual features such as color, motion, etc. can be detected, which helps the system to localize the target based on specific information. For example, the motion cue segments a person via comparing the current image frame with a background frame, which works while the person walks. The shape memory cue considers the consistency of the person's shape and detects a person using a short-term memory. These visual cues help the system to detect the target under diverse situations.

The following visual cues are covered in our work:

- Motion cue
- Color memory cue
- Shape cue
- Shape memory cue

The motion cue and the shape cue are predefined without the need of online adaptation while the color memory cue and the shape memory cue learn detection patterns based on the tracking results. The combination of predefined and adaptive cues helps the system to improve the robustness during tracking. We will explain each of these cues in the coming sections.

3.3.1 Motion Cue

Finding a moving object in a video sequence is regarded as a fundamental function in computer vision. Among various

tracking methods, the background subtraction seems to be one of the simplest but most powerful methods. The basic idea of background subtraction is as follows: assume that a camera is statically mounted in a room, we can determine the change of the camera image by comparing the current image I from the camera view with a static background image B. As furniture is placed statically and the lighting condition in an indoor environment is usually stable, the change of camera view could contain the motion information of objects.

The image difference between two frames $M(\mathbf{x}, t)$ can be computed as follows:

$$M(\mathbf{x}, t) = |I(\mathbf{x}, t) - B(\mathbf{x})| \tag{3.10}$$

where $I(\mathbf{x}, t)$ is the value of the pixel $\mathbf{x}$ of the frame t, $B(\mathbf{x})$ is the value of the pixel $\mathbf{x}$ of the background image. The value of $M(\mathbf{x}, t)$ is compared with a threshold h and the motion area is labeled with a step function:

$$f(M(\mathbf{x}, t)) = \begin{cases} 0, & \text{if } M(\mathbf{x}, t) < h \\ 1, & \text{otherwise} \end{cases} \tag{3.11}$$

Through this computation, the pixels where the current frame is not equal to the background image are labeled as 1 and the rest as 0.

Various methods for person detection based on background subtraction have been developed and the most important methods have been reviewed in (Benezeth et al., 2008; Piccardi, 2004). For example, simple methods like running average provide an acceptable tracking accuracy with limited memory requirement, while complex methods such as Mixture of Gaussian (Stauffer and Grimson, 1999) and Kernel Density

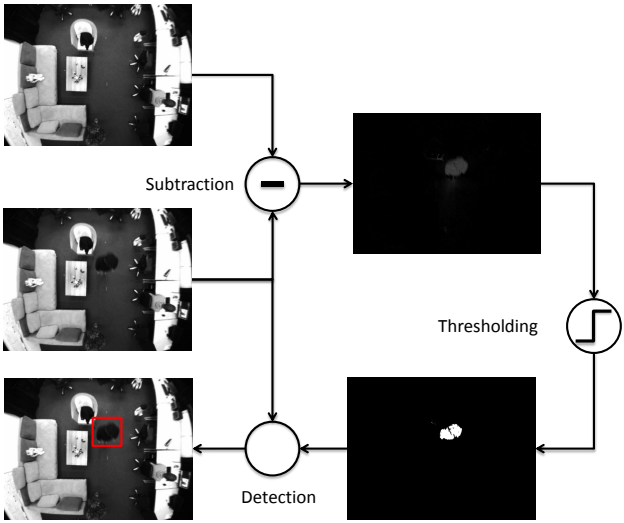


Figure 3.4: Schema of the motion detection through background subtraction

Through comparing the current frame (a person is moving in the room) with a reference background frame (before the person enters the room), the pixels different to the background are determined. The difference image is converted to a black-white image by thresholding. Blob detection is performed, and a bounding box of the detected area is labeled in the current frame.

Estimation (Elgammal et al., 2000) are robust against image noise but need more computational power. It has also been pointed out that the different background subtraction methods are difficult to compare, and the choice of the method should be based on the content of the scenario. For example, a basic method could provide a sufficient detection result in an

indoor environment with well-controlled light conditions, but will fail if used in an outdoor environment. The background subtraction method can be combined with other techniques to increase the robustness (for example with a Kalman filter (Zhong and Sclaroff, 2003)), but a major assumption of this method is that the tracked object is moving, which may not hold in the real situation.

In our work, we use the basic background subtraction because it is simple but fulfills the requirement of motion detection well. The concept of the implemented algorithm is shown in Figure 3.4. A condition of the background subtraction method is that the background needs to be constant. However, this condition is hard to be realized, because the lighting situation in a real environment changes frequently. A slight change of the light condition will be recognized as a movement, which can distract the detection results. In order to solve this problem, a common technique is used that estimates the background with the help of a running average method:

$$B(\mathbf{x}, t) = (1 - \alpha)B(\mathbf{x}, t - 1) + \alpha I(\mathbf{x}, t) \tag{3.12}$$

The background image is now not constant but dynamic according to the previous frame and the current frame. When the lighting condition changes, the background image will be adapted, which eliminates the disturbance over time.

3.3.2 Color Memory Cue

Color tracking is another important method for object detection in computer vision. Different from background subtraction methods, which are based on comparing image differences,

color tracking methods work by finding image parts that have the same/similar color as a template. Considering that color features such as a person's skin or clothes' color is stable, color tracking is capable of detecting a person easily by filtering out unrelated colors. Color detection by defining the target's color explicitly with several conditions is a simple but useful method for object tracking. For example, Kovac et al. (2003) defined a skin color classifier with combined rules, and Jordao et al. (1999) developed an adaptive mechanism for calibrating the target feature automatically. A color memory is therefore used in our model to detect the target based on the stored color pattern. As the appearance of the target (e.g., the clothes of the user) may be changed, the color memory needs to be learned during tracking.

Color Space

Color space is a special definition of colors that represents the color information with a set of digital or analog values. It is essential for visual object tracking, because the color information could be encoded in diverse spaces, which have different properties. Many color spaces have been designed and among them, the RGB space, HSV/L and YC_rC_b are the widely used. RGB refers to a three-dimensional color space of the red, green and blue primary colors (see Figure 3.5 left). It is extensively used for displaying and processing digital images. However, due to the correlation between different color channels, mixing of chrominance and luminance information, it is inconvenient for selecting and recognizing specific colors (Kakumanu et al., 2007).

The HSV color space (see Figure 3.5 right) is another

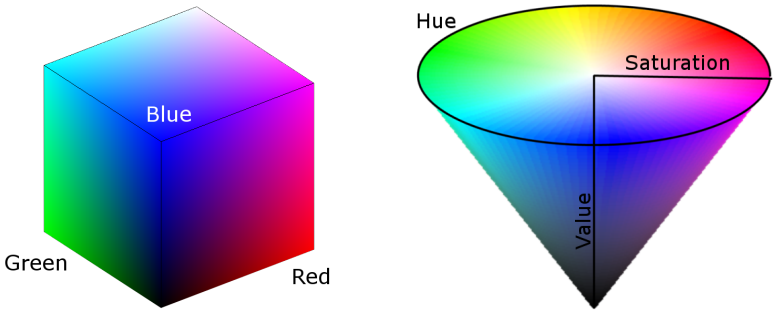


Figure 3.5: Visualization of the color space
Left: RGB-Space; Right: HSV-Space

The RGB color space is shown as a cube where dimensions are defined by three primaries: the *Red*, the *Green*, and the *Blue*. Compared with this, the HSV color space is visualized as a cone with three channels: the *Hue* (angle of the cone circle that represents the pure color), the *Saturation* (distance of the color to the axis of the central axis of the cone) and the *Value* (the height information that represents the brightness of the color).

popular color representation in computer vision. Compared with the RGB space, HSV represents the color information in three channels: *Hue* defines the dominant color, *Saturation* describes the colorfulness in proportion to its brightness and *Value* measures the luminance (Kakumanu et al., 2007). The clear discrimination of the luminance and chrominance properties is helpful for tracking color robustly. As the *Value* channel stores the luminance information, which can be related to the brightness of the environment, the influence of the change of the light can be easily compensated by neglecting the *Value* information. Another advantage of the HSV is its simple struc-

ture and the transformation of the HSV space from the RGB space can be processed as follows:

$$H = \arccos \frac{0.5\left((R-G)+(R-B)\right)}{\sqrt{((R-G)^2+(R-B)(G-B))}} \quad (3.13)$$

$$S = 1 - 3\frac{\min(R,G,B)}{R+G+B} \quad (3.14)$$

$$V = \frac{1}{3}(R+G+B) \quad (3.15)$$

Another commonly used color space is the YC_rC_b space, which is a non-linear encoding of the RGB information based on gamma-corrected RGB primaries. The Y-channel represents the luminance information by a weighted sum of the RGB information, and the C_r and C_b represent the blue-difference and red-difference color information. Because the luminance and the chrominance information are separated explicitly, this color space is also suitable to present features for color tracking (Hsu et al., 2002; Phung et al., 2002). One particular property of the YC_rC_b space is that it is not an absolute color space, but depends on the used RGB primaries. Since we want the tracking system to be robust under different lighting situations, we choose the HSV space for tracking a target object.

Single Color-based Tracking

Single color-based tracking is a simple method that detects a target with the help of reference color information. When the reference color represents the target well and is distinct from other colors, the single-color tracking can detect the object robustly by filtering out irrelevant colors. Basically, it can be processed with three steps:

1. Converting the example image and the camera image frame from RGB color space to HSV space.

   The HSV space has a clear separation of the brightness (V channel) and color (H and S channels), which is useful to improve the tracking robustness by neglecting the image brightness. Therefore, in our work we only use the H and S channel for color representation.

2. Filtering out pixels from the camera image, when their H and S value are not similar to those of the example image.

   A difference between the image pixel values and the reference value is calculated. The pixels whose difference value is greater than a threshold are set to 0 and the others are set to 1, which builds up a black-and-white image.

3. Compute the center of mass of the remaining pixels.

   Here we use an image moment for processing the center of image mass, which will be described in details in the following section.

Several experiments are conducted with this method. As shown in Figure 3.6, we select the part of a pink ball and draw the trajectory of its movement. The success rate of tracking under a well-constructed environment is high. However, it becomes unstable if the environment is cluttered. Another disadvantage is that this method uses a single, or a small range of color as a pattern. Since the clothes of a person may consist of multiple colors, only a part of them can be detected in this

case. Many approaches use distributed color to address this problem, which will be described in the next section.

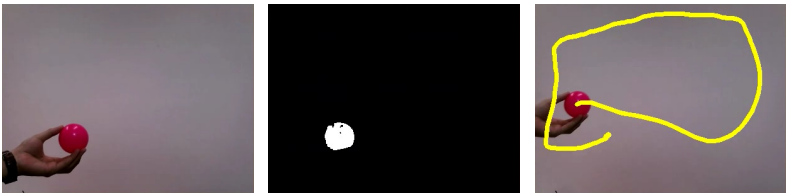


Figure 3.6: Example of tracking a ball

Left: The initial frame. Center: The ball detection based on pink color. Although the person's hand also moves, since its color is not equal to the ball, it will not be detected. Right: The trajectory of the movement of the ball.

Tracking based on Distributed Color

The general idea of distributed color modeling is simple: since a single color may not represent the object completely, an estimation of the desired object by combining a set of color features could increase the tracking performance. For person tracking in an indoor environment, the target person can be found by analyzing image patches that contain combined color information with known percentages. It is useful for clothes color tracking because clothes may have multiple colors.

The color information can be represented in different ways, for example using histograms (Novak and Shafer, 1992) (describe the color with a list of numbers that indicate how many pixels have the color within an interval), neural networks (Dong and Xie, 2005; Rowley et al., 1998) (color information stored

in the connection weights of the network), fuzzy logic (Chen et al., 1995) (describing the color with linguistic terms), etc. In addition, it can also be combined with Bayesian filter (Swain and Ballard, 1991) to increase the robustness. Moreover, the color template can be represented with parametric models, for example, with single (Hsu et al., 2002) or mixture (Mckenna et al., 1998) Gaussian distributions, which has the advantage of less memory requirement and the ability of generalization of the color information from the training data. However, as parametric models are an approximation of the color distribution, poor sampling may influence the detection rate significantly.

Since in our scenario, a user can wear different clothes in daily life, the color of his appearance will change. Therefore, approaches that require a long training time are not appropriate in this case. A rapid construction of the color template is required, and for this reason, we focus on the study of the histogram-based approach using Bayesian filter. The color histogram is a statistical representation of the color information of an image. It consists of a list of bins and each of them represents a range of colors. Values of these bins are computed by counting the number of pixels that have the color within the ranges assigned to these bins. These values provide a unique index of the image which is relatively invariant to translation and rotation (Stockman and Shapiro, 2001). Due to these advantages, the histogram becomes an important feature for computer vision.

The calculation can be explained with the following example. Given a grey-scaled image with a size of 10×10 pixels (see Figure 3.7 left), the value of each pixel is a single value

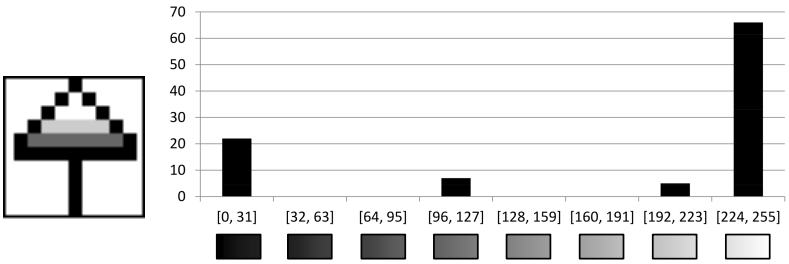


Figure 3.7: Example of computing a histogram

Left: an example image with the size of 10×10 pixels.
Right: The computed 8-bin histogram. The x-axis denotes the interval of the intensity of the gray image, and the y-axis denotes how many pixels are counted in the corresponding area. Because most of the pixels in the image are white (i.e. intensity = 255), a long bar is built in the area $[224, 255]$.

in the range of $[0, 255]$. If we compute a histogram with eight uniformly distributed bins, the range of the brightness values can be segmented like:

$$\text{Brightness} = \underbrace{[0, 31] \ \cup \ [32, 63] \ \cup \ \ldots \ \cup \ [224, 255]}_{8 \text{ bins}} \quad (3.16)$$

An 8-bin histogram can then be calculated by counting the number of pixels that belong to the corresponding bins. For example, the icon shown in Figure 3.7 left can be represented with a histogram value $\{22, 0, 0, 7, 0, 0, 5, 66\}$.

The detection of similarity of two images can be processed as follows. Assume that we have a reference image I_{E} and a testing image I_{T} with the same size, we calculate first $n-$bin

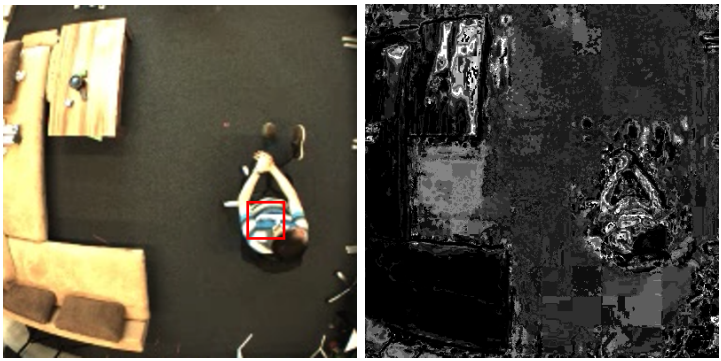


Figure 3.8: Experiment of histogram backprojection

Left: The image frame from a ceiling-mounted camera.
Right: The back projection image of the ratio histogram R based on the person's clothes color. Brighter pixels indicate a higher probability of a skin-colored object (at that corresponding position).

histograms of both, i.e. H_{E} and H_{T}. The similarity of the histogram ($\mathrm{S}_{\mathrm{Hist}}$) is calculated as follows:

$$\mathrm{S}_{\mathrm{Hist}} = \frac{\sum_{i=1}^{n} \min(H_{\mathrm{E}}(i), H_{\mathrm{T}}(i))}{\sum_{i=1}^{n} H_{\mathrm{E}}(i)} \tag{3.17}$$

As for images with the same size, they have the same number of pixels and the sum $\sum_{i=1}^{n} H(i)$ is constant. Hence, $\sum_{i=1}^{n} H_{\mathrm{E}}(i) = \sum_{i=1}^{n} H_{\mathrm{T}}(i)$ and $\sum_{i=1}^{n} \min(H_{\mathrm{E}}(i), H_{\mathrm{T}}(i)) \leq \sum_{i=1}^{n} H_{\mathrm{E}}(i)$. The higher $\mathrm{S}_{\mathrm{Hist}}$ is ($\max \mathrm{S}_{\mathrm{Hist}} = 1$), the closer is the color of the test image to the example image.

Image areas that have the similar color in comparison to the reference can be found by detecting areas with high $\mathrm{S}_{\mathrm{Hist}}$. To visualize these results efficiently and to use them to lo-

calize the position of the target, a technique with the name *backprojection* (Swain and Ballard, 1991) has been developed and proven to be a very powerful method. The *backprojection* builds up a grey-scaled image that indicates the color likelihood ratio to the reference image. A "ratio histogram" R is defined as follows:

$$R_i = \min\left(\frac{M_i}{I_i}, 1\right) \tag{3.18}$$

where M_i is the value of the bin i of the target image's histogram, and I_i is the value of the corresponding bin i of the current image's histogram.

The backprojection image uses the ratio histogram values to visualize the color similarity, which can be used to estimate the location of the target by selecting pixels with high intensity. For example, a demonstration of the person tracking based on skin color is shown in Figure 3.8. Based on the current image view and color information of the clothes of the sitting person (see Figure 3.8 left, the clothes' color is selected with the red bounding box). The ratio histogram of the target image patch and the entire image is calculated. Then, for each pixel in the current image, the corresponding histogram bin i of the pixel's color is determined and the color information of this pixel is replaced by the values R_i of the ratio histogram. Based on the clothes' color, the position of the person is labeled with lighter pixels.

Since the color information of the target person cannot be defined in advance, in our system we use the color cue to memorize the clothes' and the skin's color of the tracked person online. A small detection range is defined with a bounding box with the size of 50×50 pixels around each particle. During

tracking, a small image patch (with the size of 50×50 pixels) around the estimated position of the target person is selected and the histogram of it is learned. Then, a backprojection image of the entire image is built, and for each particle the color cue is computed by counting the pixels within the detection range around it.

3.3.3 Static Shape Cue

Shape detection refers to the techniques that find a target using specific shape features (e.g., edge and corner detection filters). Unlike the color cue, which is sensitive to light changes, the shape tracking is robust against changes of the light condition (no color information is required), and against image noise and is capable of detecting unknown objects via pattern generalization. Successful shape detection approaches have been developed in the last decades, for example, for face and gesture detection (Zhang et al., 1997).

We choose image moment features (the Hu-Moments) in our work to represent the object's shape features. The Hu-Moments (Hu, 1962) are a special kind of image moment that is invariant to image rotation, translation and scale. This gives us a good generalization ability to detect unknown targets. A multilayer perceptron (MLP) network is employed to store the Hu-Moments values using supervised learning and to classify the new image input to detect the target.

The process of calculating the Hu-Moments is shown in Figure 3.9. It consists of three steps: preprocessing, Hu-Moments calculation and detection using an MLP network. In the preprocessing step, image patches around particles are segmented using the same detection range (i.e. a bounding box

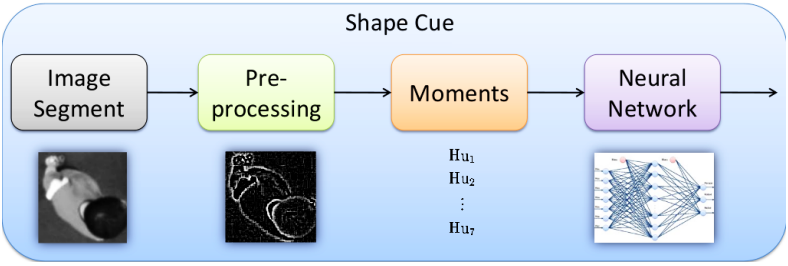


Figure 3.9: Schema of computing the shape cue based on Hu-Moments

with the size of 50×50 pixels, which is used for calculating the histogram), and filtered with a Laplace of Gaussian (LoG) filter (with a 3×3 pixel kernel) to detect edge information in the image. The LoG filter is an image filter that processes the input image by convoluting with a filter kernel. In general, an image filter with a size of $m \times n$ pixels is defined as:

$$g(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} f(i,j) I(x+i, y+j) \tag{3.19}$$

where $a = \frac{m-1}{2}$ and $b = \frac{n-1}{2}$. The function $f(i,j)$ denotes the kernel function of the convolution and $I(\cdot)$ denotes the image matrix. Features with different properties can be filtered out using specific kernel functions, for example, a *Sobel filter*, *Gaussian-blur filter*, etc. Because a Sobel filter is direction-sensitive according to the chosen Sobel kernel function, edges in the corresponding orientation to the kernel function will be detected more easily than edges vertical to the orientation. To overcome this weakness, we use a Laplacian operator with

second-order derivatives defined as follows:

$$f(\cdot)_{\text{Laplacian}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.20}$$

The Laplacian operator is able to detect edges in horizontal, vertical and diagonal directions simultaneously. In order to reduce the noise signal further, a Gaussian-blur operator is applied before using Laplacian filtering:

$$f(\cdot)_{\text{Gaussian}} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{3.21}$$

This sequence calculation of Gaussian-blur and Laplacian filtering is together known as the LoG filter. Based on the resulting edge image, we detect the counters of the person and the 7 values of the Hu-Moments are computed based on them. The MLP network is trained using these seven moments and classifies whether the image patch contains the target person. Details of the Hu-Moments as well as the MLP network will be introduced in the following sections.

Image Moments

One of the simplest ways of comparing the shape of two images is to compute their *image moments*. Generally speaking, the image moments refer to the features processed by integrating the image pixels in the horizontal and the vertical direction (Hu, 1962). A generic definition of moment can be given

as:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) x^p y^q \mathrm{d}x\mathrm{d}y \tag{3.22}$$

where $f(x,y)$ is a function of x and y, and p,q are the order of the moment. As for digital image processing, M_{pg} can be rewritten as:

$$M_{pq} = \sum_x \sum_y x^p y^q I(x,y) \tag{3.23}$$

here $I(x,y)$ is the value of pixel (x,y) in the image I. The image moment can present diverse information depending on the chosen order. For example, for a binary-color image, the zero-order moment M_{00} is computed as follows:

$$M_{00} = \sum_x \sum_y I(x,y) \tag{3.24}$$

Through summarizing the pixel values (here constant 1 for white pixels and 0 for black pixels), the area of the white pixels is obtained. Similarly, after calculating the M_{01} and the M_{10}:

$$M_{01} = \sum_x \sum_y y I(x,y) \tag{3.25}$$

$$M_{10} = \sum_x \sum_y x I(x,y), \tag{3.26}$$

we can obtain the center of the mass immediately by dividing them with M_{00}:

$$\overline{x} = \frac{M_{10}}{M_{00}} \tag{3.27}$$

$$\overline{y} = \frac{M_{01}}{M_{00}} \tag{3.28}$$

With this simple moment, it is possible to locate the position of the target. For example, in Section 3.3.2, the center of the pink ball is determined by calculating the center of the mass of the threshold image using above equations.

There are many variants of image moments, and the most widely used are the Hu-Moments (Hu, 1962), which define seven values to present the rotate-, scale- and shift-invariant features of images. The computation of Hu-Moments can be done with simple arithmetic calculations. First of all, we calculate the *center moment* μ_{pq} based on the raw image:

$$\mu_{pq} = \sum_x \sum_y (x - \overline{x})^p (y - \overline{y})^q I(x, y) \tag{3.29}$$

After normalizing them with the following equation, we obtain:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\left(1+\frac{p+q}{2}\right)}} \tag{3.30}$$

The seven Hu-Moments are calculated based on η_{pq} as follows:

$$\begin{aligned}
M_1 &= (\eta_{20} + \eta_{02}) \qquad (3.31)\\
M_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2\\
M_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2\\
M_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2\\
M_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right)\\
&\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \left(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right)\\
M_6 &= (\eta_{20} - \eta_{02}) \left((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right)\\
&\quad +4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\\
M_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) \left((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2)\right)\\
&\quad -(\eta_{30} + 3\eta_{12})(\eta_{21} + \eta_{03}) \left(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right)
\end{aligned}$$

As these values are scale, shift and rotation invariant, results of these values could be very similar, even if the target person is at different positions. Hence, the Hu-Moments are applied in our work to represent the shape of the target object. One problem of the Hu-Moments is that they are sensitive to deformation. In order to remedy this, an MLP network is employed as a classifier that is trained through a supervised learning method. Details of the network structure as well as the learning method will be described in the following section.

Classification via Multilayer Perceptron Network

In order to identify the image Hu-Moments that belong to the target person, a classifier based on a multilayer perceptron network (Terrillon et al., 1998) is used in our work. A multilayer perceptron (MLP) is a feed forward neural network, which is a standard technique for pattern recognition based on supervised learning. An MLP network is able to approximate extremely complex functions and to solve classification problems. Successful applications such as speech recognition, image recognition, etc., have been developed and widely used. The network consists of multiple layers of neurons, which are connected fully with their neighbor layers in a directed graph. Diverse activation functions can be applied, for example, a linear function with two parameters a and b:

$$y = ax + b \tag{3.32}$$

Alternatively, a non-linear function such as a sigmoid function as follows:

$$y = \frac{1}{1 + e^{-ax-b}} \tag{3.33}$$

A connection weight w is assigned to each connection in the network, which is trained to learn the significant features of the target. The target information is stored redundantly in the connection weights, which allows the network to detect the target with good generalization ability.

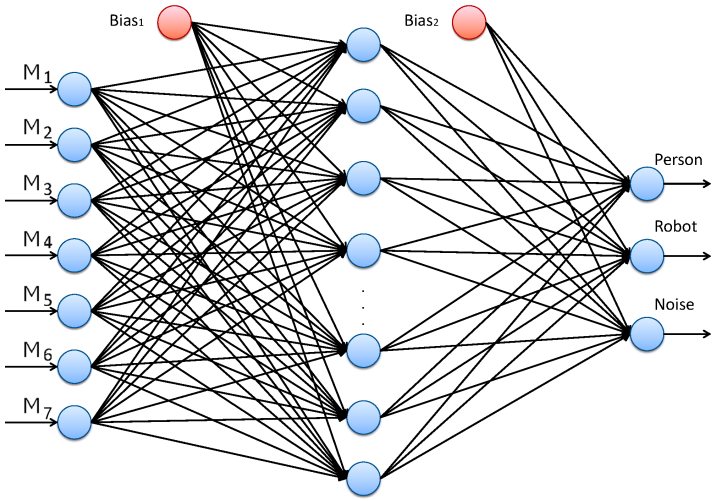


Figure 3.10: Schema of MLP network for shape classification

A schematic diagram of MLP-classifier using Hu-Moments is illustrated in Figure 3.10. Seven input neurons connect directly with the Hu-Moments. In the middle layer, we use 30 neurons with the sigmoid activation function. Three output neurons are defined with three detection categories, i.e. the image of the robot, the person and otherwise from the camera view. During the supervised learning phase, the example im-

ages are labeled with reference outputs. The training data set of each category contains 1500 images for learning and 500 for testing. The output of the neural network can be intuitively interpreted as "whether the image segment looks like a person or a robot". During the training phase, according to the images' category, the activity of the corresponding neuron will be set as 1 and the other two as 0.

The MLP network is usually trained with supervised learning methods that use a set of training data for training the connection weights of the network, and in our work we use the standard back-propagation learning. For each training step, an error value between the desired activities of the output neurons of the example data and the actual activities of the output neurons of the MLP. In our work, we use the seven values of the Hu-Moments ($M_1 - M_7$) as the input of the MLP network. Then, the difference between the network output and the desired output is calculated as follows:

$$E = \frac{1}{2} \sum_i (y_i^{\text{out}} - d_i)^2 \tag{3.34}$$

where y_i^{out} denotes the output of the neuron i in the output layer, and d_i denotes the desired output. A learning rule is applied for updating the connection weights in the network:

$$w(t) = w(t-1) + \Delta w(t) \tag{3.35}$$

with

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w(t)} + \alpha \Delta w(t-1) \tag{3.36}$$

The connection weights between the output layer and the hidden layer as well the connection weights between the hidden

layer and the input layer are updated. After the training, the network is able to classify the input image by generating activities of the output neurons. Because the supervised learning generalizes the detection from the training data, the MLP network is able to detect a person in unknown images.

3.3.4 Short-term Shape Memory

The short time shape memory detects the target object via evaluating the shape consistency over a small time-window under the assumption that the change of shape is slow. During the object tracking, the shape memory is updated based on the features of the current tracking object, which is used to detect the future objects. Besides the advantage of the shape features that they are robust against intensity changes, a short-term shape memory can detect a target in different situations without the need for training a classifier in advance based on the online learning. This is useful to compensate the failure of the static shape cue, which cannot learn all the shapes of the target. In a sequence of image frames, the shape of a target should be similar in adjacent frames. The similarity of shapes provides a possibility of tracking by rapidly memorizing the object features and using them for detection thereafter.

Two shape features are experimented with our work for building the shape memory: the scale invariant feature transformation (SIFT) (Lowe, 2004) and the Speeded Up Robust Feature (SURF) (Bay et al., 2006). Both features can be calculated instantly during image processing and used for detection in real time. Furthermore, they have the advantage that they are scale, rotation and transformation invariant, which provides a robust tracking ability under different conditions

(including rotation, shift, slight deformation, etc.). We will compare the performance of both methods and describe both methods briefly in the following sections.

Scale Invariant Feature Transformation

The scale invariant feature transformation (SIFT) is a widely used method in computer vision for object recognition through feature matching, which was developed by Lowe (1999). As in the real world it is hard to recognize a target object in different images of the target because of different camera views and scale, invariant features are very useful for object recognition by filtering out influences of image shift and rotation. The SIFT algorithm addresses this problem and successfully solved it by extracting distinctive invariant features from images, which are invariant to image orientation and image scale, and are robust to affine transformation, image distortion and illumination changes from different camera views.

The SIFT algorithm combines different techniques to realize a robust recognition function. For example, the image scale invariance is achieved by scaling the input image repeatedly to build up an image pyramid structure, and the rotation invariance is realized by assigning the dominant orientation of the features. Moreover, the SIFT algorithm is developed under consideration of efficiency problem and is optimized to be used in the real application. It uses a Difference-of-Gaussian (DoG) filter to determine interesting points in the image, which approximates a Laplace-of-Gaussian filter without the need of image convolution. Also, through a fast feature matching technique the algorithm can match features rapidly. Overall, the SIFT algorithm is able to recognize objects robustly under

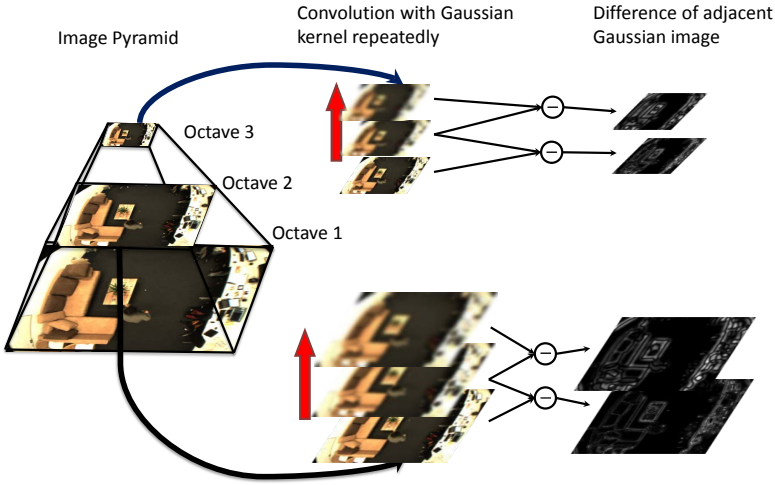


Figure 3.11: Schema of building up the difference of Gaussians pyramid

various situations, and reaches the speed requirement for real usages. Details of description and experiment results are explained in original papers (Lowe, 1999, 2004).

In order to obtain scale invariant features, SIFT uses a method for getting the position of the features, i.e. the scale invariant interest points from the image space. As shown in Figure 3.11, an image pyramid is first built that scales the raw image to different octaves. For each octave, the input image is repeatedly filtered by Gaussian kernels to blur the image at different levels. The Difference of Gaussian pyramid is then constructed by subtracting the adjacent image layers within the same octave, which results in images that approximate edge information. An interest point is determined by find-

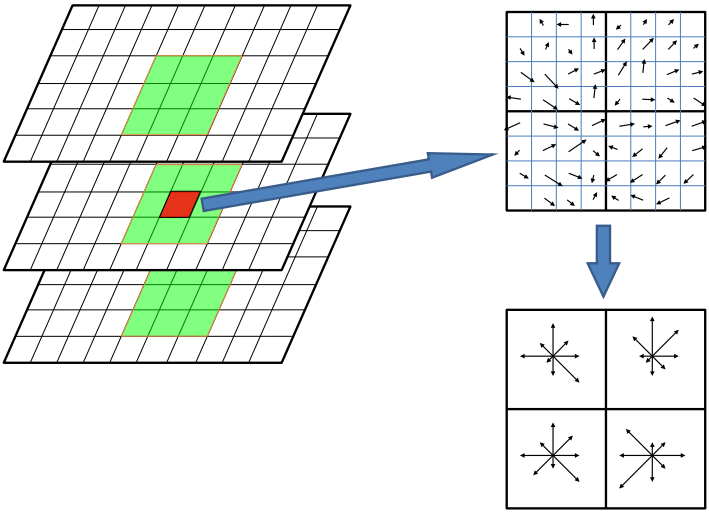


Figure 3.12: Schema of the SIFT feature

Left: Detecting an interest point by finding the extreme value.
Right: Schema of SIFT descriptor. Arrows here show the magnitude and the orientation of the image gradient within the subregions.

ing the extreme value in the difference of Gaussians pyramid, which is the local maximal value in the current spatial space as well as in the scale space (see Figure 3.12). This provides significant corner information that can be detected easily from different views.

Based on the position of the image interest points, feature descriptors that represent significant image information for recognition are generated. In the SIFT algorithm, feature descriptors around interest points are built which are 128-

dimensional vectors of the local gradient directions around the interest points. They describe the major orientation information of pixels around the interest points, which is shown in Figure 3.11. As corners and edges may have different orientations, the SIFT descriptors provide a rich feature pattern that can be used for recognition.

In order to describe the image feature on a scale- and orientation-invariant manner, the SIFT features are normalized as follows:

- Estimating the descriptor size of the interest point

  For each of the interest points, a rectangle area around the interest point with the size of $s \times \text{size}_c$ is determined, where s is the scale level of the interest point and size_c is a constant.

- Estimating the dominant orientation

  A local histogram for representing the image gradient directions is summarized over the pixels around the interest point. Then, the (multiple) peak(s) of the histogram is detected as the primary orientation of the image.

- Calculating the feature representation

  The above-determined area around the interest point is rotated based on the dominant orientation and then meshed. For each of these grids, we calculate the gradient magnitude and orientation of the image inside the grid. After filtering with a Gaussian kernel around the interest point, the results are summarized in an orientation-based histogram (for example with 8 bins, each of them

contains a range of 45°) over 4×4 subregions. The final feature descriptor contains then these values, i.e. an $8 \times 4 \times 4$-dimensional vector.

The interest point and the feature descriptor together construct the SIFT-feature. Because the 128−dimensional vectors of descriptors estimate the orientation of the image gradient within the subregion accurately, SIFT features usually provide an excellent recognition quality. However, due to the high complexity of the algorithm, the performance of SIFT feature computation is low, and it is challenging to achieve a real-time ability (especially using a resource-limited device). Approximation methods are applied to tackle this problem, and among them the Speeded Up Robust Features (SURF) becomes a successful method that accelerates the calculation significantly. Details of the SURF will be described in the next section.

Speeded Up Robust Features

Because of the repeated filtering and scaling of the raw image, the high computational complexity of SIFT features prevents the use of this method in an application with a high real-time requirement. To improve it, Bay et al. (2006) present another feature, which merges the concept of SIFT and the Haar-like features. One key technique for accelerating the computation is the integral image (Viola and Jones, 2001) which is used for pre-processing. The integral image is a novel representation of images by summing up the pixel values over the image as

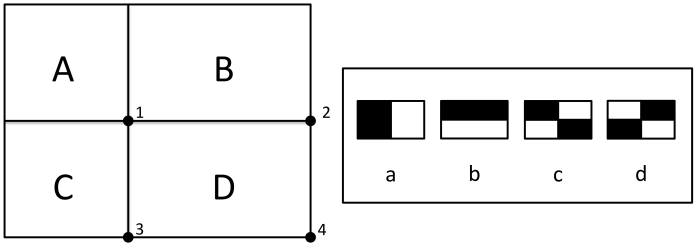


Figure 3.13: Schema of Haar-like features

Left: Integral Image; Right: A set of basic Haar-like features
The surface area of D can be processed by the position of points: $(4 + 1) - (2 + 3)$.

follows:

$$s(x, y) = s(x, y - 1) + I(x, y) \quad (3.37)$$

$$II(x, y) = II(x - 1, y) + s(x, y) \quad (3.38)$$

where (x, y) indicates the pixel position of an image, $s(x, y)$ is the cumulative row sum with $s(x, -1) = 0$, $I(x, y)$ is the pixel value and $II(x, y)$ is the value of the integral image. With the help of the integral image, any integration operation of the sub-area (or the entire) original image can be converted to simple linear operation. As shown in Figure 3.13 left, the value of an integral image at location 1 is the sum of the pixel intensities in rectangle A, and the value at location 2 is the sum of pixels intensities in the rectangle $(A + B)$. Similarly, the sum within area D is processed as $4 + 1 - (2 + 3)$ (Viola and Jones, 2001).

The detection of interest points is processed with an approximated Harris-corner detector using Haar-like features (Fig-

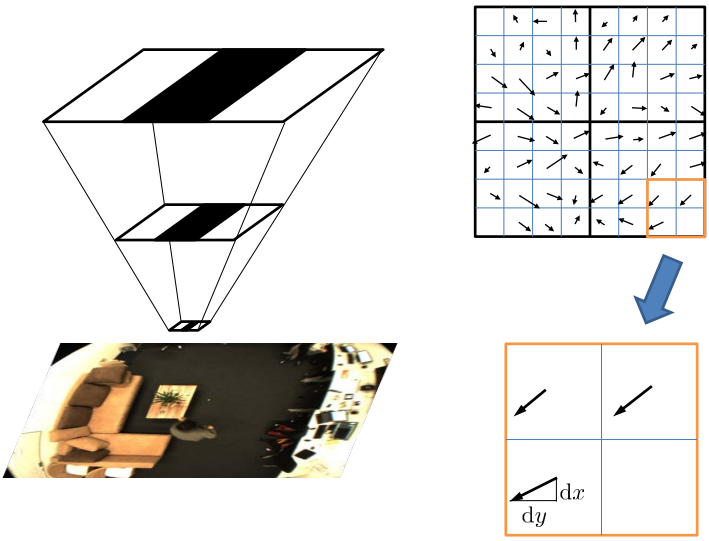


Figure 3.14: Image pyramid of SURF Features.

Other than the SIFT pyramid, which scales the image to detect interest points, SURF uses Haar-like features with different scales to detect extreme values. Then, the feature descriptor of the interest points is defined as a string of values: $\mathrm{d}x, \mathrm{d}y, |\mathrm{d}x|, |\mathrm{d}y|$. Compared with the SIFT feature descriptor, the SURF feature descriptor is constructed compactly as a $4 \times 4 \times 4$ vector.

ure 3.13 right). Haar-like features are digital image features with a similar shape of a Haar-wavelet signal. A typical Haar-

wavelet signal can be written as:

$$\phi(x) = \begin{cases} 1 & 0 \leq x \leq a/2, \\ -1 & a/2 \leq x \leq a, \\ 0 & \text{otherwise} \end{cases} \tag{3.39}$$

Similarly, the Haar-like features are represented by a set of rectangles and pixels. Inside, these rectangles are assigned with different values (for example 1 for the white area and -1 for the black area in Figure 3.13 right). The output of a simple rectangle Haar-like feature is the difference of the sum of pixels within two (the black and the white) rectangle areas (Viola and Jones, 2001). Although these operations are consuming on the original image, it can be achieved easily by processing a few linear operations on the integral image.

Another improvement of the SURF feature is instead of scaling down the raw image in the SIFT, the SURF constructs the image pyramid by scaling up the Haar-like features. Small edge features will then be filtered out by a larger Haar-like feature. Although the filtering needs an image convolution, it can be significantly simplified with the help of the integral image (Figure 3.14). Based on the output, the locations of the interest points are detected by searching the extreme value using a $3 \times 3 \times 3$ searching window in the image and scale space.

The feature descriptor of SURF uses the distribution of the first-order Haar-wavelet response in the $x-$ and $y-$axis, which consists of a 64-dimensional feature vector. To determine the dominant orientation, an area around the interest point with a radius of $6s$ is selected, where s is the scale level of the Haar-wavelets. The response of Haar-wavelets along the $x-$ and

the $y-$direction with a size of $4s$ is computed and weighted with a Gaussian filter whose mean is at the interest point. Then, the dominant orientation is estimated by determining the orientation with the highest sum of the responses inside a sliding orientation window. A feature descriptor is extracted by summing up the responses of the Haar-wavelets. A square window, which is centered on the interest point and aligned to the dominant orientation, is defined with a side length of $20s$. This square window is meshed with 4×4 grids and for each of these grids the wavelet responses are calculated. After that, the wavelet responses are accumulated in the way of $\sum \mathrm{d}x$, $\sum \mathrm{d}y$, $\sum |\mathrm{d}x|$ and $\sum |\mathrm{d}y|$, which constructs a four dimensional vector. Consequentially, each of the SURF features is built with 64 values that contain not only the magnitude ($\sum |\mathrm{d}x|$ and $\sum |\mathrm{d}y|$) but also the sign information ($\sum \mathrm{d}x$, $\sum \mathrm{d}y$). This allows the fast feature matching of this method by only considering the features with the same type of sign features, which reduces the computational complexity further.

We examined SIFT and SURF features and found that the SURF features provide a better performance while they keep a good detection accuracy for tracking the person while the person is moving in the room. Because for surveillance and also robot navigation, it is important to have a real-time tracking performance, we applied SURF features for modeling shape memory cues after evaluation. The experiments as well as results will be described in the successive section.

Structure of Shape Memory for Person Tracking

In order to improve the tracking performance using the consistency of the shape feature of the target person, we developed a

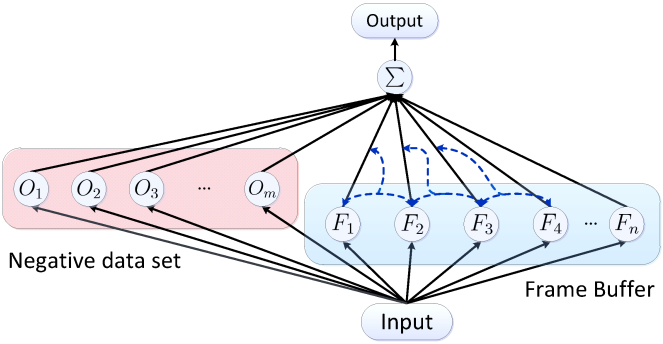


Figure 3.15: Structure of the shape memory cue

system to learn the temporary shape memory while tracking. The system structure, which is shown in Figure 3.15, consists of a feature buffer $F_1 \dots F_n$ to store the temporary features of the target person, and a negative data set $O_1 \dots O_m$ to store features of the background. The feature buffer is defined to store image features of the last 30 frames. When the target person is localized by other cues, features of the target will be extracted and pushed into the buffer. Since the change of the person's shape is continuous and slow, the features of previous frames from a recent time are similar. Therefore, the output of the shape memory is calculated based on the correlations between the new input image feature and the features in the buffer. The higher the correlation is, the higher the output of the shape memory cue is. We also consider features of the background and a negative dataset is defined to store features from the static background such as from sofas, tables

and chairs, which could help particles to avoid mislocalization in these areas proactively. When a particle is located in the furniture area, the negative data set will be activated, and a negative value will be added to the output. The output of the shape memory cue is processed by summing up of values, and higher value is produced when a strong correlation between the current image feature and the stored previous target's features is found.

3.4 Visual Cues Integration

In the last section, we introduced visual cues that detect the target using diverse algorithms. Each of the visual cues is able to detect the target with individual features, which provides a good tracking/detection performance under a certain condition. One research question here is which one among these outputs from visual cues should be more trustworthy. For example, the color memory needs to be learned before using because the person may change his clothes. At the beginning of tracking, the output of the color memory could not provide useful information and should not distract the general tracking. For this reason, results from the visual cues need to be combined synergistically to improve the overall detection rate.

We consider that for perfectly built multiple visual cues, these visual cues should agree with their detections during the target object tracking. In other words, a correlation of outputs of visual cues could help to increase the detection robustness. In our work we use a polynomial high-order network, which is inspired by a Sigma-Pi network (Zhang and Muhlenbein, 1994) for multiple cue integration. The sigma-pi network uses

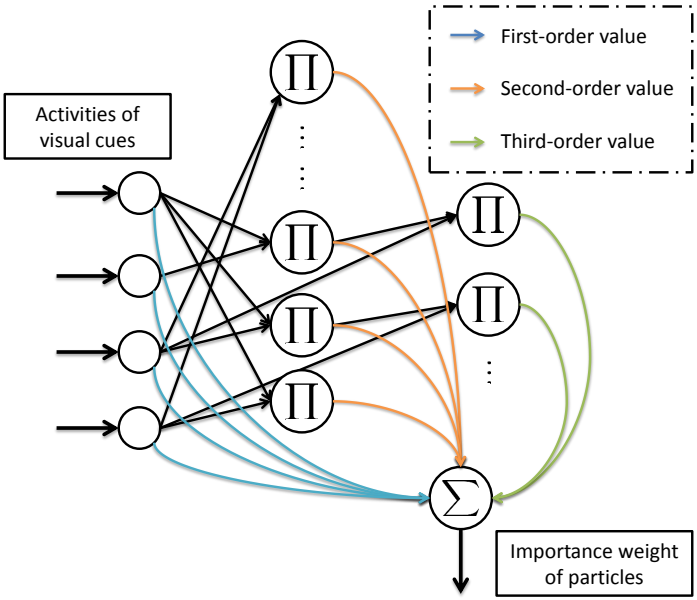


Figure 3.16: Schema of a Sigma-Pi network

outputs of visual cues as well as further generated cues as neural input (see the first-order, second-order and third-order values in Figure 3.16). A connection weight is assigned to each of these combinations, and the output of the network is processed by multiplication of the connection weights with corresponding neural activities. The output of the network shows a majority voting of results of visual cues, which is used for particle weights' calculation. Here a higher value indicates the higher probability that the image patch belongs to the target, which increases weights of particles within these areas

and changes the particle distributions. For each particle i, the particle weight $\pi^{(i)}$ is calculated based on the output of the Sigma-Pi network as follows:

$$\begin{aligned}\pi^{(i)} = &\sum_{c}^{4} \alpha_c^l(t) A_c(s_{t-1}^{(i)}) + \\ &\sum_{c_1>c_2}^{4} \alpha_{c_1c_2}^q(t) A_{c_1}(s_{t-1}^{(i)}) A_{c_2}(s_{t-1}^{(i)}) + \\ &\sum_{c_1>c_2>c_3}^{4} \alpha_{c_1c_2c_3}^c(t) A_{c_1}(s_{t-1}^{(i)}) A_{c_2}(s_{t-1}^{(i)}) A_{c_3}(s_{t-1}^{(i)})\end{aligned} \tag{3.40}$$

where $A_c(s_{t-1}^{(i)}) \in [0,1]$ is the activity of cue c at the position of particle i which can be thought of as taken from a saliency map over the entire image (Itti et al., 1998). As said, a higher activity shows that the image patch contains features of the target, which affects the particle filter update and particles in this area acquire higher particle weights. We use a sigmoid activation function to process the neuron activities, which can be described by Eq. (3.41):

$$A(y_c) = \frac{1}{1 + e^{-(g \cdot y_c)}} \tag{3.41}$$

where y_c is the output of the visual cues and g is a selected scale factor. The activity of each cue is scaled by the reliability α, which represents "how plausible" this cue is. This is important because at the beginning of tracking, no color information is learned, and therefore the output of the color cue should be ignored. The coefficients of the polynomial cues, i.e. the network weights $\alpha_c^l(t)$ denote linear reliabilities, $\alpha_{c_1c_2}^q(t)$

quadratic and $\alpha^{c}_{c_1c_2c_3}(t)$ cubic combination reliabilities of visual cues. The quadratic and cubic combinations of the four basic cues yield the further ten combination cues. Compared with traditional multi-layer networks, the Sigma-Pi network contains the correlation and higher-order correlation information between the input values.

The reliability of some cues, like motion, is non-adaptive, while others, like color, need to be adapted on a short timescale. This requires a mixed adaptive framework, as inspired by models of combining different information (Bernardin et al., 2008; Triesch and Malsburg, 2001). An issue is that an adaptive cue will be initially unreliable, but when adapted it may acquire high quality in predicting the person's position. To balance the changing qualities between the different cues, the reliabilities α will be evaluated with the following equation:

$$\alpha_i(t) = (1-\epsilon)\alpha_i(t-1) + \epsilon f(s'_t) + \beta, \forall\alpha \tag{3.42}$$

where ϵ is a fixed learning rate and β is a constant value. $f(s'_t)$ denotes an evaluation function and is computed by the combination of visual cues' activities:

$$f_c(s'_t) = \sum_{i\neq c}^{n} A_i(s'_t)A_c(s'_t) \tag{3.43}$$

where s'_t is the estimated position and n is the number of the reliabilities. In this model n is 14 and contains 4 linear, 6 quadratic and 4 cubic combination reliabilities. When more cues are active at the same time, the combination of cues provides a higher output that leads to the increase of the corresponding active cues' reliability α.

3.5 Experiments

Experiments for person and robot tracking are conducted in a home-like laboratory environment with a single room. As shown in Figure 3.17, a ceiling-mounted camera is used for observing the target person. In order to cover the entire room with a single camera, a fish-eye camera lens is applied, which causes a strong image distortion. For this reason, the shape of the target person could be entirely different when (s)he stands at different positions in the room (e.g., Figure 2.3 in Chapter 2), which distracts the person tracking using traditional shape-based methods.

The goal of our experiments is to track a person (or a robot) in real-life-like scenarios. Different situations, such as static situations (such as when the person is sitting or standing on a spot) as well as dynamic situations (such as when the person is walking in the room) are tested. Besides, different disturbances such as moving furniture and changes in lighting conditions are tested. A distracting person who has a similar shape feature but differs by color should be ignored. Therefore, a list of tasks is defined, which includes:

- Moving person
- Sitting person
- Distracting person
- Changing light condition
- Changing furniture position
- Distracting person using CLEAR 07 data set (Bernardin and Stiefelhagen, 2008)

Figure 3.17: A home-like experimental environment

Furniture is observed by the ceiling-mounted camera that is shown in the left lower inset in the image.

All these tests have been carried out in our ambient laboratory. A similar room or data set can also be used to evaluate the algorithm, but the calibration of the camera setup in the new room and the training data of the MLP network are needed because the shape of the person may be different due to the change in the camera view. A particle filter with 30 particles is used for tracking, and according to the approximate size of the tracked object, we define a segmentation area of 60×60 pixels around particles for image processing. A reference image is captured at the beginning to obtain the initial background model. The SURF features of the initial background model

are extracted and stored as a negative dataset. When a person is moving in the room without a known color pattern, the shape and motion cues will detect the person, and the particles will merge to the position of the person. The histogram of the image patch around the estimated position of the person will be updated, and the SURF features of this image patch will be extracted and be pushed into the memory buffer. Based on the output of the visual cues, the corresponding reliabilities of visual cues are adapted according to Eq. (3.42). The details of the scenarios, as well as the corresponding results, are shown in the following sections.

3.5.1 Test Scenarios

Moving Person

Tracking a moving person is a basic task in the experiment, and it can be achieved using motion information. A test example of tracking a moving person is visualized in Figure 3.18. At the beginning of tracking, all particles are initialized at random positions, with uniform weight in the image (see frame 5). When a person enters the room (frame 100), the movement of the person is detected by the motion cue. The particles in the detection area receive a positive value from the motion cue, thus, their particle weights will be increased. According to the updating and resampling of the particle filter, particles close to the person get higher values, and the distribution of particles will then move towards the detection area. While the person is moving, the motion cue will provide activities at the person's location, and particles around the person will increase their weights. By repeating these processes, the particle filter

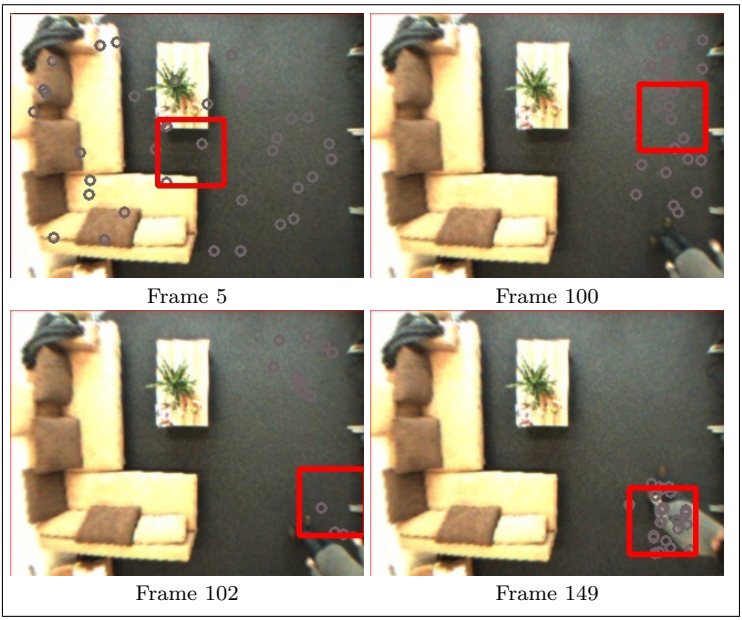


Figure 3.18: Tracking a person while moving in the room

estimation will then converge to the person's position. After the person is localized (frame 149), the shape features, as well as the color histogram, are learned and the reliabilities of visual cues (e.g., the shape memory cue and the color cue) are adapted accordingly to improve the tracking robustness.

Sitting Person

Although motion information is robust and easy to be detected, a pure motion tracking system only works while the target person moves. In the case that no motion is detected,

for example, when the person stays in a position for a long time, the system has to rely on other cues in order to maintain localization. Since some cues (e.g., the color and the shape memory cue) need to be trained before being used, the features of color and the shape memory cues should be learned during tracking, which helps the system track the target robustly.

In this test scenario, we demonstrate person tracking of a person with a static pose (e.g., while sitting). Since motion may not be sufficient for tracking, other features such as shape and color memory are learned and used for person detection. A test scenario is shown in Figure 3.19: a target person is first localized based on motion detection during walking, and then the shape as well as clothes' colors are learned during tracking. After that, the target person moves to a sofa and sits there (frame 401). Although no motion information is obtained at this moment, the particle filter keeps tracking the person successfully based on the information on other cues, e.g., the color and the shape memory cue. The learned features provide a strong signal to localize the target person, which could also avoid a mislocalization caused by wrong motion estimation. For example, when a distracting person close to the target person moves, which provides a noise motion signal, the particles are still locating the target person robustly based on the results of the decision-making (See frame 420 and frame 423: some of the particles move away from the target person due to the noise motion signal and then come back after a short time).

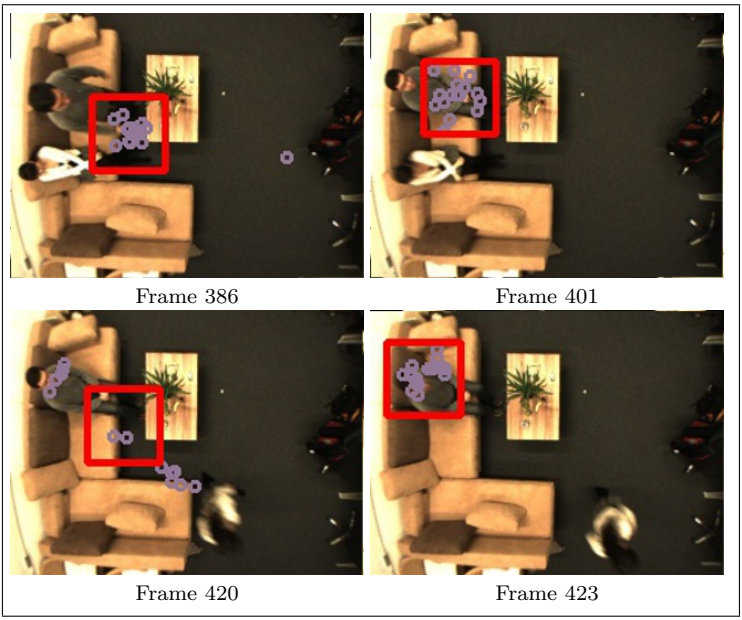


Figure 3.19: Tracking a person while sitting close on a sofa

Walking with Distracting Persons

As the system learns the features of the target person while tracking, rather than specifying them manually, the movement of another person may also be detected by the motion cue, which could disturb the localization. Therefore, it is important to evaluate the performance of tracking a single target person among other persons in the room. In this task, we test the possibility of tracking a target person when another person is in the room. Both persons may walk separately or cross each other, which could lead to an overlap of their shapes

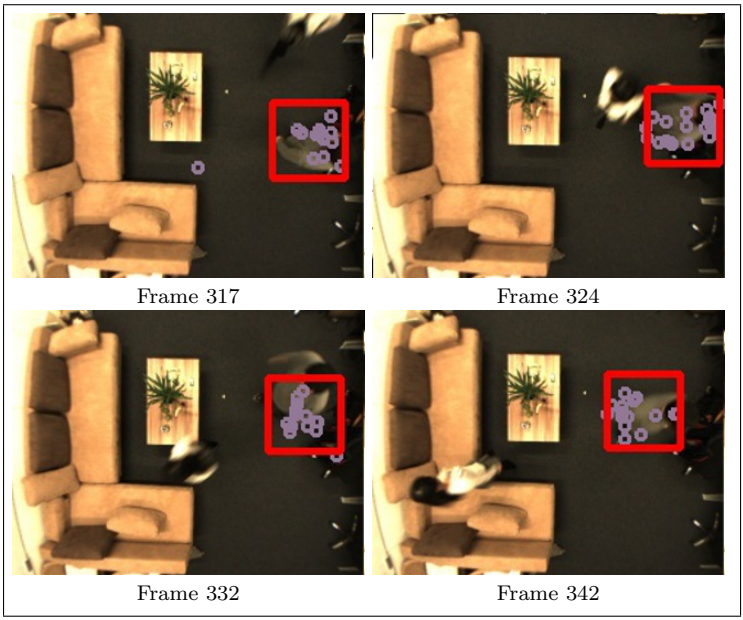


Figure 3.20: Tracking a person while crossing a distracter person

and disturb the color memory and the shape memory feature for a short time. The system should be robust against these disturbances (also the motion, which can provide strong noise information) and localize the target person successfully.

In Figure 3.20, we show a test scenario of person tracking while a distracter person traverses. In the beginning, we first let the system track a target person successfully (see frame 317). During tracking, features such as color and shape information of the target are learned, which could increase the tracking robustness. After that, a distracting person enters

the image and walks around the target person (frame 324). When two persons come very close to each other, motion information of both persons will be merged, and parts of the features of the distracting person may be memorized. In this case, the tracking system is still able to filter out the noisy information (frame 332). Moreover, while the target person stands and the distracting person provides lots of motion information, the system is capable of detecting the target person using the learned color and shape information.

In the previous test scenario, shown in Figure 3.19, the target person sits first on the sofa close to another person (frame 386). Since the target person does not move after that (frame 401), no motion information from the target person is detected. When the other person stands up and moves, some particles are disturbed due to the wrong motion information (frame 420). However, the color and the memory cue are still capable of recovering the system quickly, and the particles will come back to the target person again (frame 423).

Changing Light Condition

The lighting condition of a real environment changes steadily. Although slow changes could be compensated with algorithms (e.g., the moving average of the motion detection), a quick and significant change may cause problems for object tracking. The large difference between the previous and the current image frame could be recognized as motion information. In addition, low brightness could also disturb the color and feature detection. The clothes or skin color may change under a different light source, and parts of the shape features may be difficult to detect in a dark environment.

In this section, we challenge the person localization by changing the light condition in a short time, which will disturb the target detection of multiple visual cues. In this case, the localization may fail and feature learning of the shape and the color memory will be paused when the sum of the reliabilities is below a threshold. After resuming the good light condition, the system should recover the tracking as soon as possible with the help of the learned features. When the bad light condition keeps for a long time (e.g., more than 5 seconds), the wrong color information may be learned and the color memory cue may distract the tracking system. In this case, the reliability of the color cue will drop because the output of it disagrees with the other (e.g., the motion cue could track the person again when the dark light condition becomes stable) until the new color information is learned correctly. A test scenario is shown in Figure 3.21. After a person is located by the particle filters (frame 85), we first switch the lights off to build a darker environment and switch the lights on again. Due to the dramatic change of the intensity, the particles lose the target person (frame 105), and the distribution of particles becomes wider. Note that due to the change of intensity, the motion cue provides only noise signals here. In this case, the color and shape memory helps the system detect the person. The recovery is shown in frame 115, where, after a short time, the particles converge to the person again.

Changing Furniture Position

Besides moving persons, moving chairs, as well as a blinking television, may be detected wrongly based on the motion cue, which disturbs the person tracking. Therefore, it is impor-

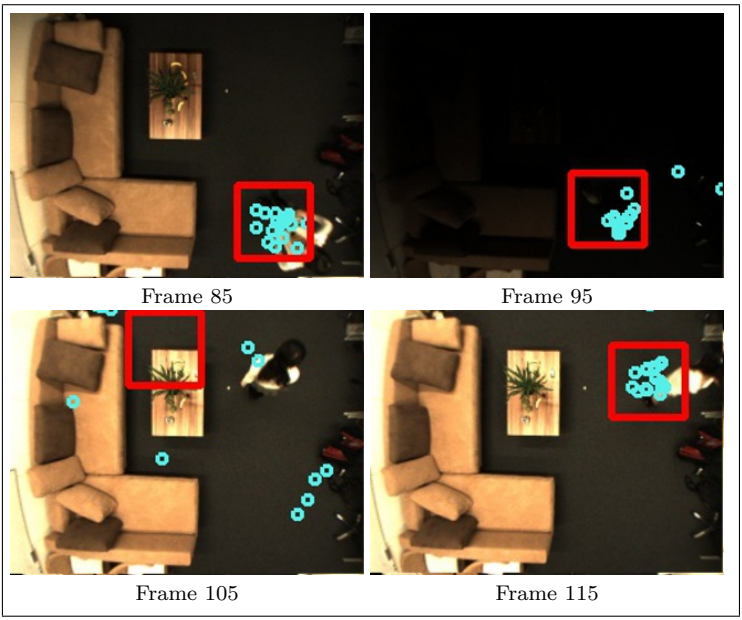


Figure 3.21: Tracking a person while changing light condition

tant to test the tracking under a changing environment. The experiments with changing furniture position are conducted similarly as experiments with distracting persons. Compared with the experiments with distracting persons, one significant difference is that while a distracting person moves independently to the target person, the change of the furniture would be done by the target person. For example, when a person grasps a chair and brings it somewhere else, the first motion of the chair will happen where the person grasps it, which causes a merged motion blob, and parts of the chair's features could be learned while the person is moving together with the

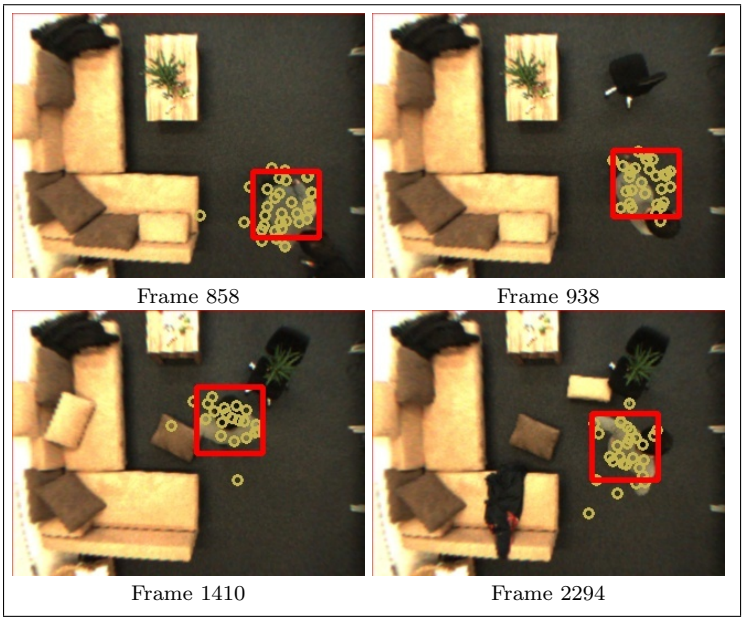


Figure 3.22: Tracking a person while changing objects' positions

chair. Since the motion detector cannot distinguish between the furniture and the person (the motion information of both is merged), the motion cue may determine the target person incorrectly, and the features of the furniture, together with the features of the moving person, may be learned wrongly. The tracking system needs to be robust in order to filter out these pieces of noise information based on other visual cues, for example, using the learned shape and color information.

We experiment with the person tracking in a cluttered room environment, and a test scenario is shown in Figure 3.22.

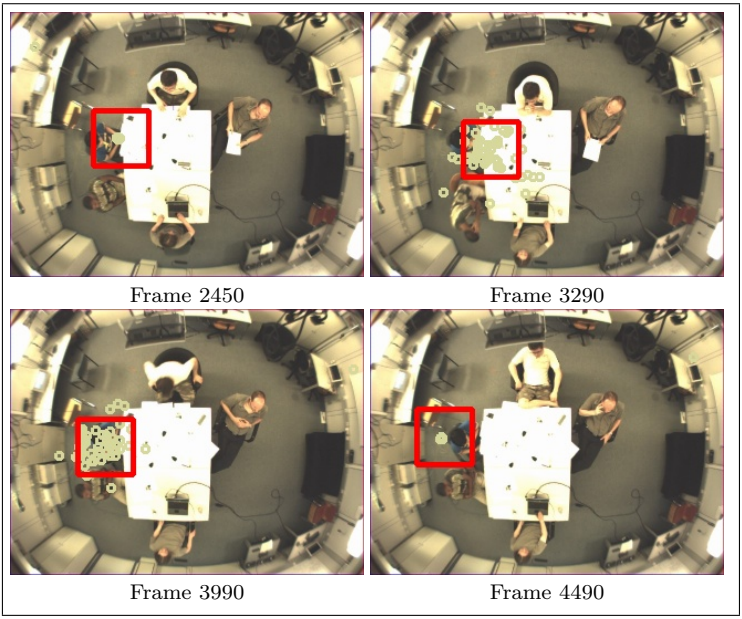


Figure 3.23: Test with the "CLEAR 07" data set

The shape and the color information of the target person is learned when the person is moving in the room. After that, the person moves some objects in the room (cf. the chair, the plant, etc.). Although a motion cue may provide incorrect information, and particles may follow the movement of objects wrongly, the learned color and shape cues could still detect the person, which helps the particles to recover the tracking quickly.

Distracter Person Using CLEAR 07 Data Set

We conducted a set of experiments based on the fish-eye camera video of the CLEAR 07 short sample data set to evaluate tracking performance based on external data. The idea of our system is to monitor a target person who is alone in the room. Since the "CLEAR 07 multiple persons tracking" data set is designed for evaluating tracking performance for multiple persons, our current system will rely on selecting a target person. Therefore, we can only evaluate the system when one of the people is tracked. The experiment is shown in Figure 3.23. We select the person at the middle left position as the target person and the others as distracting persons. While several persons may be moving at the same time, the selected target person is tracked successfully based on the combination of cues.

3.5.2 Evaluation

We refer to the CLEAR MOT Metrics (Bernardin and Stiefelhagen, 2008) to calculate the tracking performance. This Metrics is a method for evaluating the tracking performance of a multiple-target tracking system based on multiple cameras. Two large data sets are used (i.e., CLEAR 2006 and CLEAR 2007) with the CLEAR MOT metrics to compare the tracking quality to various methods. The CLEAR MOT Metrics considers different errors during tracking: missing m (target cannot be detected any more), false positive fp (localization goes to wrong place) and mismatching mme (tracking of multiple objects works, however, targets are wrongly labeled). The tracking accuracy **MOTA** is then computed according to these

three failure values as follows:

$$\textbf{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{3.44}$$

where g_t refers to the frames that the target objects can be found in the image.

Since in our work only a single person is tracked in the system, we slightly adapt the method so that only the frame number of missing m and of false positives fp have been counted to calculate the object tracking accuracy (OTA). The threshold distance of a false positive was defined as 40 pixel: the target person is labeled as localized when the distance between the center of the bounding box of person tracking, and the center of the target person is smaller than this distance. In addition, since the video data from the ceiling camera belongs to the evaluation of 3D person tracking using multiple sensors, the results of the person tracking will not be compared with our method because we only use one camera.

11 videos were evaluated, and the results are summarized in Table 3.1. As we can see, 89.96% of the images on average have a correct tracking result. The best case is the changing light condition in the day scenario which indicates that a slight change of light under sufficient sunshine does not disturb the tracking system at all. The worst case is the changing light condition in the night scenario. However, it is also the hardest test, because the lights are the only light source. The light condition is changed totally when most of the lamps are switched off and a person can hardly be observed from the camera video (see frame 95 in Figure 3.21). In comparison, the success rate of tracking a person based on single motion detection could

Table 3.1: Experiment results

Name	**Total Frames**	m	fp	**OTA (%)**
Person moving scenario 1	2012	19	22	97.96
Person moving scenario 2	2258	169	12	91.98
Person moving and sitting scenario 1	1190	78	21	91.68
Person moving and sitting scenario 2	980	22	130	84.18
Changing environment scenario 1	1151	89	30	89.66
Changing environment scenario 2	1564	157	141	80.94
Changing light condition in night scenario	160	17	59	52.50
Changing light condition in day scenario	540	0	3	99.45
Distracting person scenario 1	1014	48	35	91.81
Distracting person scenario 2	700	57	26	88.14
Distracting person scenario CLEAR 07	2122	188	52	88.68
Total	13691	844	531	89.96

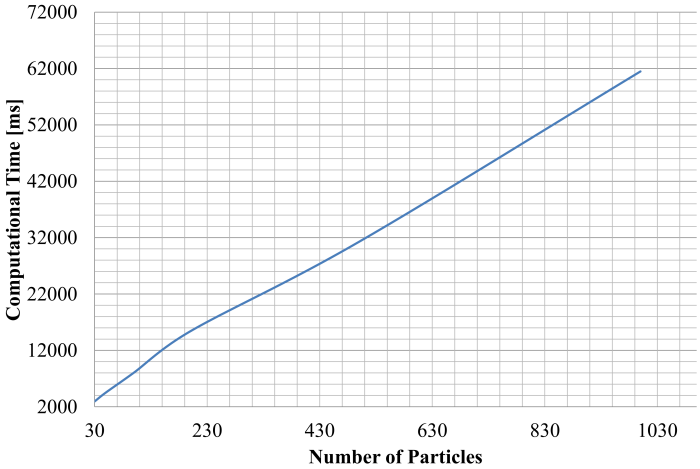


Figure 3.24: Computational time for different number of particles

reach 69% on average, and the color and memory cue alone could not achieve the tracking task.

Real-time Capability

A real-time capability is desired for a tracking system that works in a real environment. Therefore, we investigate the computational complexity of our model qualitatively as well as quantitatively. Among the used visual cues, the shape memory cue using SURF has the highest computation complexity $\mathcal{O}(wh)$, where w denotes the width of the image and h denotes the height of the image. Since these visual cues are computed for each particle, the cost of computing visual cues of all particles is then $\mathcal{O}(pmn^2)$, where m denotes the number

of particles, p the number of linear visual cues and n denotes the small detection window. To assign the reliabilities it takes $\mathcal{O}(p^2)$ and to resample the particle costs $\mathcal{O}(m)$. Thus, the total computational effort is $\mathcal{O}(pmn^2) + \mathcal{O}(p^2m) + \mathcal{O}(m)$. Because the number of visual cues is constant, for example here $p = 4$, the total cost is then $\mathcal{O}(mn^2)$. For each step, only small image patches are processed, which is much more efficient than the global searching methods. As n is far smaller than w and h, $n^2 << wh$ and the computational complexity is controlled by selecting the number of particles m. Through controlling the number of particles, the total resulting computational time will be different. As we can see, the growth of the computational time is approximately linear to the increase of particle numbers, which fulfills our expectation.

A set of speed test has been made and visualized in Figure 3.24. For each test we process 100 frames of images. Although a very large m may even slow down the computation (e.g., see Figure 3.24 when more than 1000 particles are used), by choosing an appropriate m, the system could work much faster than processing the entire image while keeping a good tracking quality. Figure 3.25 shows the tracking quality of a target person using different number of particles. Through selecting an appropriate particle number (in our work 30 - 50), a good tracking quality can be achieved with a sufficient real-time capability.

Reliabilities of linear cues

The contribution of visual cues can be evaluated by their reliability values. The more often a visual cue helps to find the person, the higher the reliability of this cue will get. The re-

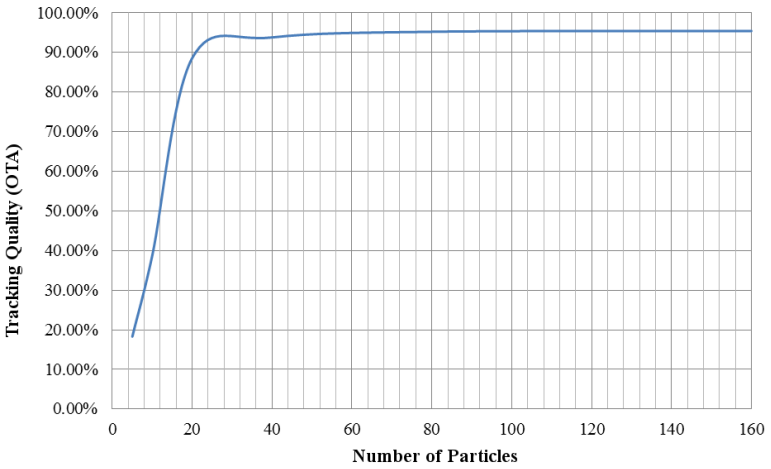


Figure 3.25: Tracking quality using different number of particles

liabilities of linear visual cues are displayed in Figure 3.26, which are important to visualize the adaptation of reliabilities of the motion, the shape, the color memory and the shape memory cues. The x-axis of the diagram denotes the frame number and the y-axis the weight values. For example, at the beginning of the tracking, the color cue has a small reliability value since the histogram has not yet been learned. When the color information is trained (for example after frame 300), the reliability of the color cue increases to a high value and the color cue becomes more important for the target detection. The shape memory cue usually has a high value because this cue memorizes the shape of the target person that is very reliable. The motion cue provides a high value at the beginning

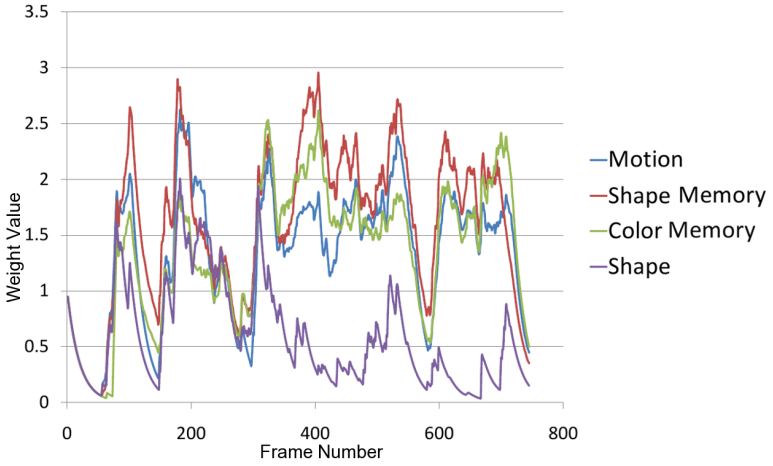


Figure 3.26: Reliabilities of linear cues

and supports the other cues (e.g., the memory cue) to learn features. The shape cue has a lower value than the others, because the shape of a person is always changing and is hard to be classified continually. Moreover, we can see that at certain point the reliabilities of all cues drop off together (e.g., around frame number 600). This could happen when the particle filter misses the target person, for example, when the person moves out of the image. One problem that may arise is that when all the cues fail, the reliabilities could be very low and need longer time to recover. One possible way to improve this is that we could set up the β value in Eq.(3.42) differently for visual cues so that the reliabilities of them could converge to different level when they fail. For example, the motion cue could have a higher β value than the other so that the motion

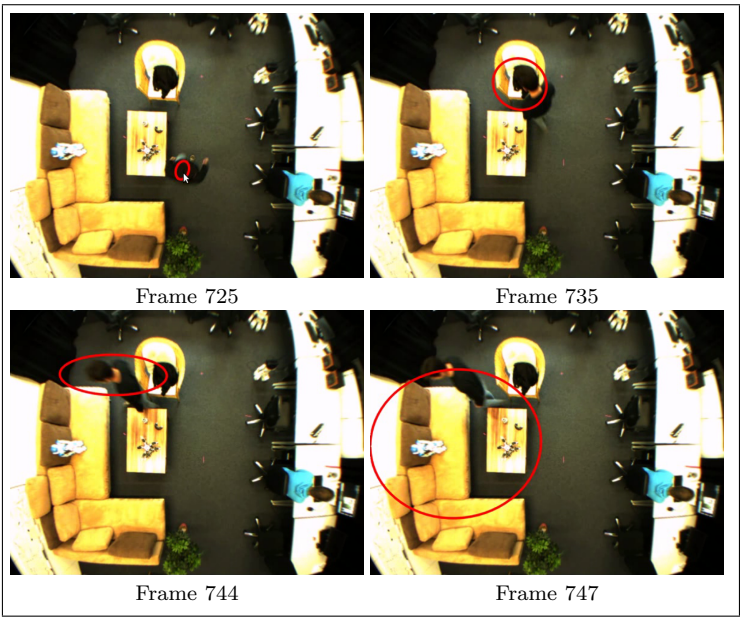


Figure 3.27: Experiment with the CAMshift algorithm

information could dominate the localization when no other useful information is available.

3.5.3 Comparison with other Algorithms

Experiments are conducted in our laboratory environment to compare the tracking performance of our system with other algorithms. Here, we provide the results of two state-of-the-art feature-based tracking methods: Continuously Adaptive Meanshift (CAMshift) (Bradski, 1998) and the Tracking-Learning-Detection (TLD) algorithm (Kalal et al., 2012). Both algo-

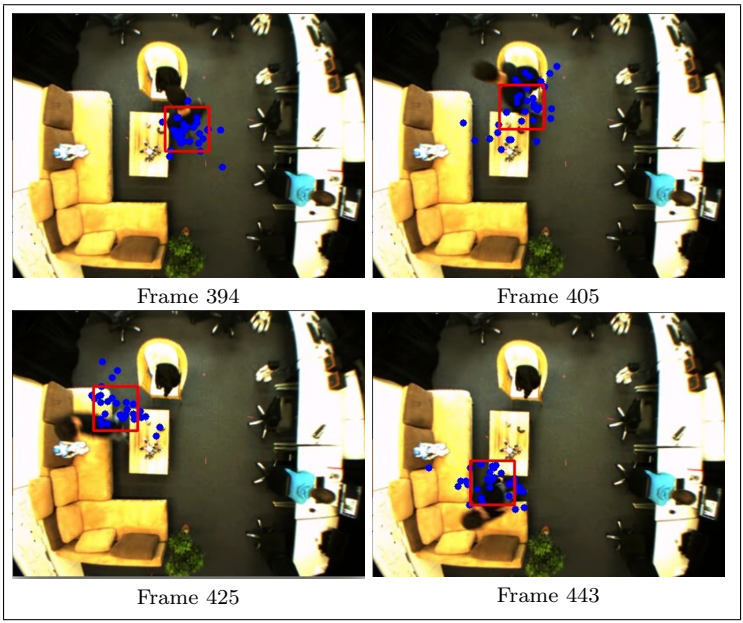


Figure 3.28: Experiment with our tracking algorithm

rithms and our tracker run with the same video resources, and the results will be analyzed in the following sections. Other methods (e.g., motion detection) will not be tested here, as their drawbacks are known.

CAMshift

The CAMshift algorithm is a popular method for tracking objects based on color information[1]. It uses histogram color in-

[1]Source code for experiments: Example code from OpenCV, `http://opencv.org/`.

formation to represent the target object and determines the target by finding the area with the highest probability (the density of pixels that contain the desired color). The method could precisely estimate the position, as well as the size of the target, using an adaptive searching window, and since only the color information is required, the method is able to deal with target deformation. It has also been seen a successful tracking method, for example, for face tracking.

However, since the tracking is based on pure color representation, the distribution of the color of the target has to remain stable. As shown in Figure 3.27, based on the target's color representation, the target person is labeled with a red ellipse. When the color of the target changes quickly during tracking (frame 735 and frame 744), noise information may be adapted, and the target will be lost (frame 747). Compared with this, our tracking system provides a robust detection result during the entire video (see Figure 3.28).

Tracking-Learning-Detection (TLD)

The Tracking-Learning-Detection algorithm is a novel method developed by Kalal et al. (2012) that aims at tracking unknown objects in real time through long-term feature learning. The algorithm consists of three parts: a *tracker* that estimates the target position based on the motion information; a *detector* that recognizes the target based on the learned features in the past, and a *learning* that estimates the false negatives and false positives of the detection using a novel P-N learning method and generates features of the target as well as image noises to be learned. Given the initial position and the size of the target, the TLD algorithm is able to track the target based on

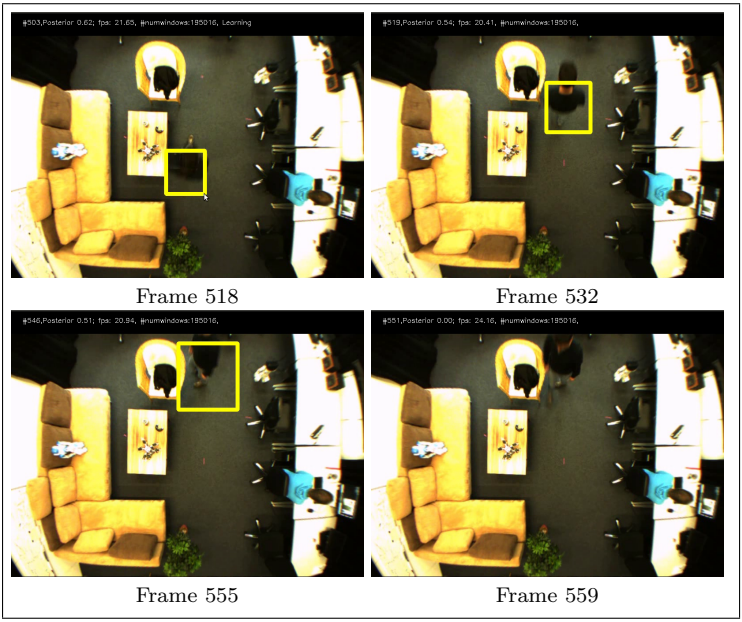


Figure 3.29: Experiment with the TLD algorithm

the motion information, learn new features of the target and the noise information from the background, and detect the target using the learned pattern without the need for extra information.

We test the TLD algorithm[2] by evaluating the previous video file, and the results are shown in Figure 3.29. As we can notice here, the TLD is unable to track the target person successfully. After a short tracking period (from frame 518 to frame 555), when the target person turns up, the model cannot

[2]Source code: `http://www.gnebehay.com/tld/`

find any known features and loses the person. As the P-N learning measures the similarity of the current image patch with the learned positive and the negative image patches, a quick change of the shape of the target person observed by the ceiling-mounted camera view may provide a low similarity and cannot be classified. In addition, the tracker of the TLD, based on median flow, may not estimate the new position of the target person correctly, because the shape of the person deforms significantly using the fish-eye lens and the ceiling-mounted camera view. The performance of TLD, in this case, is worse than our algorithm.

3.6 Discussion

In this chapter, we presented a real-time person tracking system using a ceiling-mounted camera. A hybrid probabilistic algorithm is proposed for localizing the person based on different visual cues. A particle filter is developed in our model to detect the target person effectively using the small image patches around particles, estimate the new position of the target person, and update the reliabilities of the visual cues based on the position estimate. We apply a high-order combination network (i.e. the Sigma-Pi network) to integrate the results of different cues with corresponding reliability weights. This model indicates a human's ability to recognize objects based on different features. When some of the features are strongly disturbed, e.g., by the noise of motion information, the system is capable of providing a robust tracking by recovering the detection with other features. The particle filter resembles an active attention selection mechanism, which allocates most

processing resources to positions of interest. It delivers a robust object tracking quality while keeping the real-time ability. Nevertheless, since some visual cues (e.g., the shape and the color memory) need to be learned during tracking, they could only provide useful information after the features are learned correctly.

Advantages of this combined multiple-cue system are that it provides a flexible architecture for multiple-cue integration, which is easily extendable by adding other features. (In our work we also included the entire TLD algorithm as a visual cue.) In addition, non-visual cues (e.g., audio) could be used to enhance the tracking accuracy. Based on an adaptive learning method, features such as color histogram and shape features can be adapted online to improve the robustness of person identification. With this short-term memory mechanism, the system could master the challenge of an unstructured environment, as well as moving objects in an ambient intelligent system. Accordingly, our model has the potential to be further developed as a robust method for object detection and tracking in complex conditions. Though the initial design of the tracking system is to monitor a single person when the person is alone at home, the system is scalable to track multiple persons through a dynamic management of particle filters.

Chapter 4

A Neurocognitive Model for Robot Navigation

Robot navigation is an important core function of socially assistive robotics that guides a robot to move to the target position to accomplish its tasks. Although it is a well-established research topic, the navigation has similar problems as the person localization described in the previous chapter: Among many existing methods, it is still hard to achieve a robust, flexible navigation behavior in a cluttered, dynamic environment with limited representation. Most of the state-of-the-art SLAM algorithms use expensive sensors (e.g., laser scanners) for environment perception, which greatly increases the system cost. Nevertheless, the measurement failure, due to the signal noise, makes it difficult to obtain a correct representation of space, which could distract from the navigation planning.

Although a concrete path planning is helpful in acquiring an optimized navigation trajectory, it is laborious and insufficient when the environment changes (e.g., when objects move). Another disadvantage of the state-of-the-art SLAM is that it is only limited to mapping the room and solving path planning. Further cognitive functions, such as environment understanding, are hard to extend.

To overcome these challenges, we developed a novel navigation system that combines the robot navigation system with an ambient intelligent environment to extend the sensor range of the robot and use a ceiling-mounted camera in a room, as the sensor input, to detect the position of the robot, as well as the target person. The ambient sensors can extend the view of the robot with this integration. For instance, during the robot's navigation towards a target person, the ceiling camera can help the robot to locate the person when (s)he is out of the range of the robot's sensor. This little external support could help the entire navigation system significantly to realize complex behavior with a low-cost hardware system.

Another research question we tried to answer is how to achieve a decent trajectory planning while keeping high efficiency and flexibility for dealing with new situations. Considering that human beings and animals could reach the target position in a large scaled environment easily, guided by a spatial feeling and without very detailed motion for each step, the understanding of the spatial relation and real-time action for fast maneuvering is crucial to accomplishing this task. Our navigation system, therefore, integrates planning and reflex-based obstacle avoidance to adapt the navigation behavior during environment changes while maintaining effi-

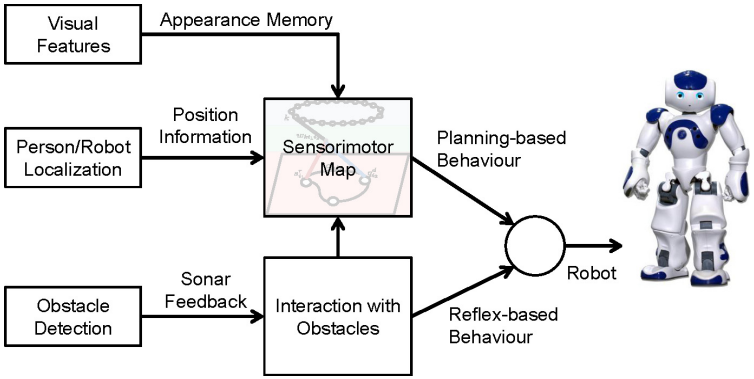


Figure 4.1: General architecture

ciency based on the optimized planning trajectory. Inspired by the animal's spatial cognition ability, we use a sensorimotor map to represent the environment, the relation between spatial states, and the corresponding actions that lead to the states' transfer. Details of the system will be described in the following chapters.

4.1 Approach

The architecture of our navigation framework is shown in Figure 4.1. Three sensors are applied: (1) a robot camera fixed on the robot's head for extracting the visual features of the environment, (2) a ceiling-mounted camera of the ambient intelligent system to localize the person and the robot, and (3) sonar sensors installed on the robot's chest for detecting obstacles. The visual features and the localization information are used for building up a spatial cognition functionality of the

robot, in our model with a sensorimotor map, and the sonar signals are used for obstacle detection and avoidance, which also helps the robot adapt to its sensorimotor map.

Two kinds of navigation behavior have been developed in our model: a *planning-based behavior* that guides the robot strategically based on the spatial knowledge stored in the sensorimotor map and a *reflex-based behavior* that interacts with the environment in real time and reacts to the unknown objects to ensure the safety of the robot. As the core component of the entire system, the sensorimotor map stores the spatial information and the corresponding motor information during the change of spatial status. A path planning can be processed based on the sensorimotor map that results in motion information, in our model a desired orientation towards the target. Meanwhile, the reflex-based behavior controls the maneuvering of the robot in real time, which generates the detailed robot motion and avoids dangerous areas. Hence, a flexible and efficient navigation control is realized through a synergetic combination of these two behaviors.

As a person usually does not collide with obstacles while (s)he is walking, the position information of the person is useful to represent the free space in the room, and in our system, this location information is used for building up the sensorimotor map and planning navigation. Through the integration of the localization and the navigation system, the robot is able to detect the target person and navigate towards him autonomously. While a reflex-based behavior is activated, feedback of the interaction with the environment will be sent to the sensorimotor map and adapt to the corresponding state automatically. This adaptation is important for the robot to

move in a dynamic environment and helps the robot avoid potential danger position proactively. The visual features are used for gathering the appearance information of the environment, which is anchored on the robot's position on the map. Through the integration of appearance features, the robot is able to process more cognitive tasks, such as localizing an object, which is observed during the robot's movement. The details of each component will be described in the following sections.

4.1.1 Sensorimotor Map

The sensorimotor map is the core of the navigation system. As shown in Figure 4.2, it consists of three components: *(1)* a spatial memory that learns the structure and the appearance of the environment, *(2)* an action memory that learns an inverse control model, and *(3)* an action layer for robot control. The spatial memory layer represents the spatial information of an environment. It contains two types of spatial information: *states* that present the features and *connections* that present the relations between different features. When a person, as observed by the ceiling-mounted camera, or a robot visits a novel location, the features of this location and the relation with neighboring places will be stored. When the robot is at a specific position, the neurons whose features match the ones obtained from the current position will be activated. This resembles the neural activity of the place cells in the hippocampus, whose firing rate is dependent on the current location. The spatial memory is represented by a Growing When Required (GWR) network (Marsland et al., 2002). Compared with the Growing Neural Gas model (Fritzke, 1995) that we

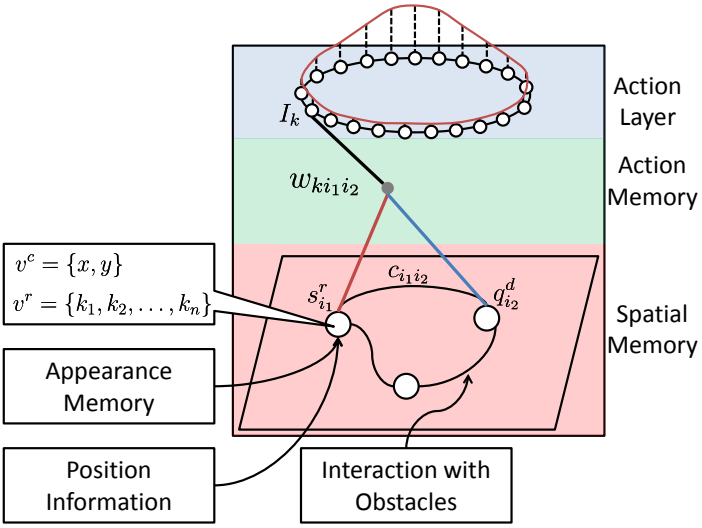


Figure 4.2: Architecture of the navigation model

used in Yan et al. (2012b), GWR does not grow over time, but only when the novelty is detected. When the entire space is explored and no change happens, no neuron and connection of the GWR network will be updated, which provides better convergence properties.

The network consists of a set A of neurons, each associated with feature vectors v, and a set N of connections to describe the relations (i_1, i_2) between neurons i_1 and i_2 in A. Different features can be represented in the neurons, for example the x, y coordinate information of the robot and the visual landmark information, which resembles the visuospatial perception (which will be described in chapter 4.3). These features are used to determine the position of the robot as well

as the target. Neurons and connections will be allocated or updated dynamically using a competitive Hebbian learning rule. A connection weight $c_{i_1 i_2} \in [0, 1]$ is defined for each connection (i_1, i_2) to indicate how "easily" a robot can move along this connection. The higher $c_{i_1 i_2}$ is, the easier the robot can walk through the link. When a connection is created, its connection weight $c_{i_1 i_2}$ will be initialized to 1 and adapted during the robot navigation.

During navigation, the desired next state will be estimated in the spatial memory layer to guide the robot to the target position (see Section 4.2). A reward signal is set at the target neuron and is broadcasted through the network, and the desired next state according to the robot's current state is estimated. As an output of the spatial memory layer, the activity signal of the present state of the robot, i.e., q^c (the red line in Figure 4.2), and the desired next state, i.e., q^n (the blue line in Figure 4.2), will be sent to the action memory layer.

The motion of the next step is planned according to the current and the next desired robot status. In order to convert the state transition to a concrete motion control, an action memory that represents the action information of each state transition is employed. Motions are combined and updated in the action layer with a ring-form dynamic neural field (DNF), which shows the desired orientation of the robot. Details of the computation of q^c and q^n, as well as the action memory, are described in the following sections.

Learning Position Information

The map building is based on the position information $\xi = \{x, y\}$ from the person or the robot localization and orienta-

tion model using the ceiling-mounted camera (for details about the person- and robot localization please see Yan et al. (2011, 2012a)). At the beginning, the GWR network is initialized with two randomly placed neurons that are linked to each other. Then, based on the new input, the network will be updated, and new neurons are created to learn the new data distribution. This growing feature is essential to achieve a good data representation while keeping resource usage optimized.

During position information learning, the winner neuron i^* and the second winner neuron i^{**} are first found by calculating the map activity s (later we will use s^p to indicate a person and s^r to indicate a robot in Section 4.2) based on ξ and the coordinate information of the neurons v_i^c with the following equation:

$$s_i = e^{-\frac{||v_i^c - \xi||^2}{2\sigma^2}} \tag{4.1}$$

where σ is a constant parameter and the index of the winner and the second winner neuron is computed as follows:

$$i^* = \arg\max_i s_i \tag{4.2}$$

$$i^{**} = \arg\max_{i \neq i^*} s_i \tag{4.3}$$

These two neurons are important for network growing, and a new node will be added between neurons i^* and i^{**} when the following two conditions hold:

1. The activity of the winner neuron s_{i^*} is smaller than a

threshold activity a_t, which means that the person is far from the position represented by any map unit, and

2. the firing counter h_{i^*} has become smaller than h_t, which means this neuron cannot move a lot anymore.

When these conditions match, a new neuron is inserted in the GWR with the following steps:

1. Insert a new neuron r with the average weights of the winner neuron and the current position:

$$A \leftarrow A \cup \{r\} \tag{4.4}$$

$$v_r^c = \frac{1}{2}(v_{i^*}^c + \xi) \tag{4.5}$$

where A denotes the list of the neurons and v_r^c the coordinate information of the new neuron.

2. Insert links between r and i^* as well as between r and i^{**}

$$N = N \cup \{(r, i^*), (r, i^{**})\} \tag{4.6}$$

where N denotes the list of the connections.

3. Remove the current connection between i^* and i^{**}

$$N = N/\{(i^*, i^{**})\} \tag{4.7}$$

The coordinate information of the winner neuron (i.e., $v_{i^*}^c$) as well as its neighborhood neurons (v_i^c for all directly adjacent neurons of i^*) is updated based on the input. Each neuron is assigned an age factor age_i, which can increase incrementally, and a firing counter h to control the adaptation efficiency. In

order to improve the convergence of the network and to have a homogeneous distribution of the neurons, the adaptation is as follows:

$$\Delta v^c_{i*} = \epsilon_{i*} h_{i*} (\xi - v^c_{i*}) \quad (4.8)$$

$$\Delta v^c_n = \epsilon_n h_n (\xi - v^c_n). \quad (4.9)$$

where ϵ_{i*} and ϵ_n are the fixed learning rates of the winner and the neighborhood neurons, and h_{i*} and h_n are the corresponding firing counters. The firing counters (initialized with $h_0 > 0$) are calculated as follows:

$$\Delta h_{i*} = \tau_b (\mu h_0 - h_{i*}) \quad (4.10)$$

$$\Delta h_n = \tau_n (\mu h_0 - h_n) \quad (4.11)$$

where τ_b, τ_n and μ are constant parameters for controlling the adaptation. The firing counters indicate how "active" a neuron is. When a neuron is added to the network, its firing counter is initialized as a high value h_0, which allows it to adapt its features quickly. During the iteration, h decreases towards a small value μh_0 and the neuron loses its mobility, which ensures that the neuron's positions become stable. After adaptation, we increase the age of all edges that connect with neuron i^*:

$$\text{age}_{(i*,n)} = \text{age}_{(i*,n)} + 1, \quad (4.12)$$

and delete the connection whose age is over a threshold age_m. Isolated neurons that have no neighborhood will be deleted as well. For details of parameter setting, please see Table 4.1.

4.1.2 Forward and Inverse Model

The forward and inverse model represents the robot control signals coupled with the state transition in the spatial mem-

Table 4.1: Parameter table of the sensorimotor map building

σ	20	ϵ_{i^*}	0.05
ϵ_n	0.02	τ_b	0.165
τ_n	0.066	μ	0.09
h_0	1	age_m	50
a_t	0.7	h_t	0.5

ory layer. The action information is learned in the weights $w_{ki_1i_2}$ that connect fully with the neurons k in the action layer. Depending on the way of controlling the robot, the action information can be presented in a different form, for example as force, velocity, angle value, etc. In our case the robot is controlled by adjusting its heading direction. We use a ring-form dynamic neural field, which will be described in the next section. During map building, when a robot is moving in a room and its spatial representation changes from i_1 to i_2, the action executed at that time will be associated with this state transition. The robot memorizes actions for state transitions and is able to navigate through recalling appropriate memories. Instead of calculating action online, the motion control is achieved by combining action behaviors, which increases the computational efficiency significantly.

We use second order weights $\{w_{ki_1i_2}\}$ (also called Sigma-Pi weights (Weber and Wermter, 2007)) to store the action information associated with the state transfer in the spatial memory. When the robot moves from i_1 to i_2 in the spatial

memory, the corresponding second order weight will be activated with the input signal $s^r_{i_1}$ of the current state i_1 and $q^d_{i_2}$ of the desired next state i_2, and the output I_k towards unit k in the action layer will be computed as follows:

$$I_k = \sum_{i_1,i_2} w_{ki_1i_2} s^r_{i_1} q^d_{i_2} \tag{4.13}$$

where the value of $q^d_{i_1}$ is calculated based on the reward spreading from the target, which will be described in section 4.2. Eq. (4.13) sums up those inputs because multiple connections may be activated at the same time using the distributed representation of the robot's position. The neural field of the action layer has 36 nodes, hence $k \in \{1, 2, \ldots, 36\}$. Assuming that there are m neurons in the spatial memory layer, the total number of connection weights $w_{ki_1i_2}$ are $36m^2$, which grows quadratically according to the number of spatial memory neurons. However, most of the weights are close to 0 since the action layer is sparsely connected with the spatial memory layer.

The connection weights of the DNF can be learned based on the observation of the person's movement, or based on the robot's location and motion information. Here, we describe the method of action learning regarding the observation of the person's movement, because the action learning based on the robot's movement is the same in principle. Based on the person's location, the winner neurons with respect to the person's position will be determined at first and all the connections $c_{i_1i^*}$ between the winner neuron i^* and its neighborhood neurons (indexed with i_1) will be adapted. Assuming that a connection $c_{i_1i^*}$ is active, the direction associated with this connection is

calculated, and the corresponding weights $w_{ki_1i^*}$ are trained. Since there are two possible walking directions for every connection (from i_1 to i_2 and from i_2 to i_1), we train both weights at the same time as follows:

1. According to the position (x_{i_1}, y_{i_1}) of neuron $\{i_1\}$ and (x_{i^*}, y_{i^*}) of neuron $\{i^*\}$ of the spatial memory, we calculate the possible orientation $o_{i_1i^*}$ of connection $c_{i_1i^*}$ using inverse trigonometric functions:

$$\begin{aligned} \Delta x &= x_{i^*} - x_{i_1} \\ \Delta y &= y_{i^*} - y_{i_1} \\ o_{i_1i^*} &= \arcsin\left(\frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}}\right) \\ o_{i_1i^*} &= \pi - o_{i_1i^*} \qquad \text{if } \Delta y < 0 \end{aligned} \tag{4.14}$$

   And then we calculate the opposite orientation $o_{i^*i_1}$:

$$o_{i^*i_1} = o_{i_1i^*} + \pi \tag{4.15}$$

2. Two bumps of activation with the size of the DNF are being created in the shape of a circular normal distribution, one around the link orientation:

$$p_{ki_1i^*} = \frac{e^{\kappa \cos(\frac{k \cdot 10\pi}{180} - o_{i_1i^*})}}{2\pi J_0(\kappa)} \tag{4.16}$$

   and the other around the opposite orientation:

$$p_{ki^*i_1} = \frac{e^{\kappa \cos(\frac{k \cdot 10\pi}{180} - o_{i^*i_1})}}{2\pi J_0(\kappa)} \tag{4.17}$$

where $p_{ki_1i^*}$ is the k-th connection weight of the action memory for orientation $o_{i_1i^*}$, κ is a constant and $J_0(\kappa)$ is the modified Bessel function of order 0 (Abramowitz and Stegun, 1965):

$$J_0(\kappa) = \frac{1}{\pi} \int_0^{\pi} e^{\kappa \cos(\theta)} \mathrm{d}\theta \tag{4.18}$$

3. We minimize the errors between the current activation p and the second order weights w, i.e., $\frac{1}{2}||p_{ki_1i^*} - w_{ki_1i^*}||^2$. Here we use a gradient method for learning. The equation is shown as follows:

$$\begin{aligned} \Delta w_{ki_1i^*} &= \eta(p_{ki_1i^*} - w_{ki_1i^*}) \\ \Delta w_{ki^*i_1} &= \eta(p_{ki^*i_1} - w_{ki^*i_1}) \end{aligned} \tag{4.19}$$

where η is a fixed learning rate.

Note that the learning of the sensorimotor map (including the spatial map as well as the forward and inverse model) here is based on the pure position observation of the moving person as the person's motion is unknown. Compared with this, in case of learning the cognitive map by observing the robot's movement, the forward and inverse model can be built based on the robot's action signal and the learning method remains the same.

4.1.3 Action Layer

The action layer generates robot control signals based on the active action units during navigation. A DNF model is used

to merge these action signals and to adjust the robot's walking orientation by showing the desired robot orientation. The DNF is a biologically-inspired model of the neural dynamics in cortical tissues (Amari, 1977) to represent information with activities of a population (or field) of neurons (which is also called population coding). It is widely used in robotics to generate dynamic behavior (Cuperlier et al., 2005; Erlhagen and Bicho, 2006), which is useful to model a flexible navigation strategy. The model provides an activation bump that is stabilized with the neuron association. Each neuron in the network is associated with its neighborhood neurons, which is stimulated by the close neighborhood neurons and, inhibited by the neurons far from it. Based on this method, noise signals that appear randomly can be filtered out automatically with the help of the inhibitory connections. Because the target is not represented by a single output but a bunch of neurons, it resembles a probability distribution and DNF is robust against noise signals while keeping high efficiency.

A DNF is capable of integrating the multiple action codes received by the action layer and adjusting the robot's motion with smooth orientation behavior. It can be constructed with an arbitrary form based on the requirement and in our work a one-dimensional ring-form DNF with 36 neurons is implemented to represent the desired robot orientation in 10° increments. As shown in Figure 4.3, each neuron k of the DNF has a membrane potential u_k that represents the activity and lateral connections n_{kj} with other neighbor neurons j. Through the following updating rule (Eq. (4.20)), the DNF generates an activation bump dynamically to show the suggested orientation for the next step.

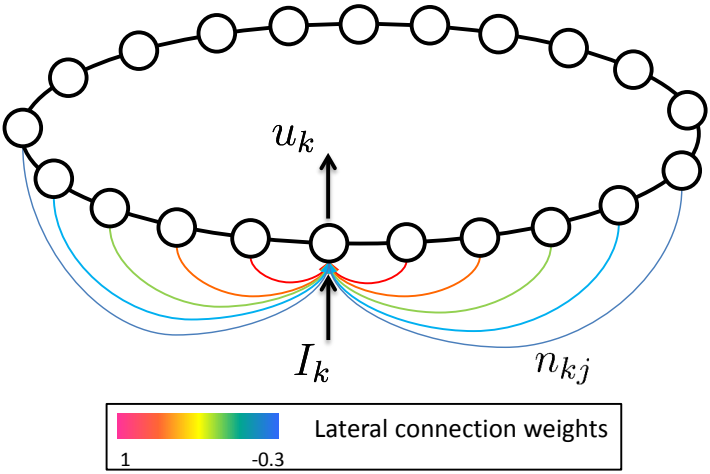


Figure 4.3: Schema of a ring-form dynamic neural field

$$\tau \Delta u_k = -u_k + \sum_{j=1}^{36} n_{kj} f(u_j) + I_k + h \tag{4.20}$$

where h is a rest potential, τ is a temporal decay rate of the membrane potential, and I_k is the input stimulus of the k-th neuron received from the second order weights that encode the desired robot orientation. We use here a Gaussian function with negative offset as the function n_{kj} (Figure 4.4) to describe the lateral interaction of neurons:

$$n_{kj} = \beta e^{-\frac{(k-j)^2}{2\sigma^2}} - c \tag{4.21}$$

where β is a scaling factor, σ^2 a variance, k, j the index positions of neurons and c a positive constant. Each neuron in

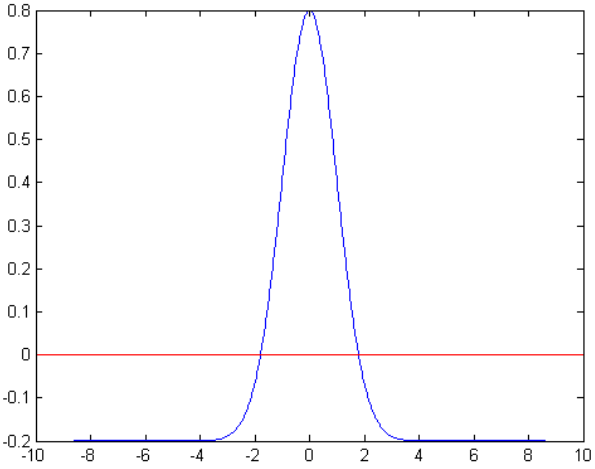


Figure 4.4: Schema of the Gaussian function for lateral interaction

the neural field receives positive stimuli from neighborhood neurons close to it while neurons far from it provide negative stimuli to inhibit its activity. The function $f(u)$ is a sigmoid transfer function of a single neuron with a constant offset g:

$$f(u_j) = \frac{1}{1 + e^{-(u_j - g)}} \tag{4.22}$$

The robot's desired orientation O^d is obtained based on the neuron activities of the neural field, where each neuron represents a specific direction. The orientation O^d is computed

through a weighted average of the activities as follows:

$$\hat{v} = \begin{pmatrix} \hat{v}_x \\ \hat{v}_y \end{pmatrix} = \begin{pmatrix} \sum_k u_k \sin(\frac{10k}{180}\pi) \\ \sum_k u_k \cos(\frac{10k}{180}\pi) \end{pmatrix} \tag{4.23}$$

$$O^d = \begin{cases} \arctan\left(\frac{\hat{v}_y}{\hat{v}_x}\right) & \text{if } \hat{v}_x > 0 \text{ and } \hat{v}_y > 0 \\ \arctan\left(\frac{\hat{v}_y}{\hat{v}_x}\right) + \pi & \text{if } \hat{v}_x < 0 \\ \arctan\left(\frac{\hat{v}_y}{\hat{v}_x}\right) + 2\pi & \text{if } \hat{v}_x > 0 \text{ and } \hat{v}_y < 0 \end{cases} \tag{4.24}$$

The control signal of the robot is computed by the differential of O^d and the actual orientation of the robot O^p estimated by the localization model (for details please see Yan et al. (2012a)), which is described as follows:

$$\Delta O = \begin{cases} -c_o & \text{if } O^d - O^p > d \\ c_o & \text{if } O^d - O^p < -d \\ 0 & \text{else} \end{cases} \tag{4.25}$$

where c is a constant rotation speed parameter and d is a constant threshold. Through experiments with the robot in the home-like environment we detected suitable parameters which are listed in Table 4.2.

4.2 Planning and Navigation

As described in section 4.1.2, the input signal of the action layer is computed based on the connection weights w of the DNF, the activity of the current state q^c and the activity of the next desired state q^d. These values are important to update

Table 4.2: Parameter table of the forward and inverse model

κ	0	η	0.2
σ	2	g	3
d	0.3	c	0.3
c_o	0.3		

the activity of the DNF and to control the robot's motion. Because the connection weights w are trained during map building and q^c are computed with respect to the robot's position, the remaining q^d that determine the next desired state of the robot is crucial for path planning.

In order to compute it, we assign each neuron of the spatial memory a reward value that spreads from the target state representations iteratively with an exponential decrease. Seeds of the reward signals are placed on the position of the target, which is determined first in the spatial memory. The localization of the target can be detected differently depending on the navigation task. For example, we can use coordinate information v^c or appearance memory feature v^r as initial signals to define the target. Once the target position is known, an initial reward $r_i(0)$ at the target location is calculated with the following steps:

1. Calculate the input signals m_i of the neuron i in the spatial memory. For approaching a person, we calculate the signals m based on the distance between the person's position ξ and the neuron's coordinate v^c. Assume that

the person's position is distributed with a position list ξ_s (indexed in s) with corresponding probabilities w_s where $\sum_s w_s = 1$, m is computed as:

$$m_i = \sum_s w_s e^{-\frac{||v_i^c - \xi_s||^2}{2\sigma^2}} \tag{4.26}$$

This means, the closer v_i^c of neuron i is to ξ_s, the higher is the activity of this neuron. Another possible method of calculating m_i for example based on feature matching is described in the following sections.

2. Normalize the match signals with a softmax function:

$$\widetilde{m_i} = \frac{e^{m_i}}{\sum_{i'} e^{m_{i'}}} \tag{4.27}$$

where small activities are depressed after the normalization.

3. Assign m_i to initial reward signals with a threshold filter:

$$r_i^p(0) = \begin{cases} \widetilde{m_i}, & \text{if } \widetilde{m_i} > \text{threshold}_m, \\ 0, & \text{else} \end{cases} \tag{4.28}$$

where p indices the neuron of the initial reward signal.

After the initialization, seeds of the reward signals are placed in a small area around the target position. Multiple units will contribute to localize the target object/person since the person's location is presented with a probabilistic distribution and the object features may be represented at different

positions. For each $r_i^p(0) > 0$, the reward signal will spread separately to the neurons connecting to the neuron i (listed in nl) iteratively:

$$r_j^p(t+1) = \lambda c_{ij} r_i^p(t), \text{ for } j \in nl(t) \text{ and } r_j^p(t) < r_i^p(t) \quad (4.29)$$

where λ here is a discount factor and c_{ij} is the corresponding connection weight. The neighborhood list nl will be updated for each iteration as follows:

$$\begin{aligned} n' \quad &\leftarrow \quad i \quad \text{if } i \text{ connects with neuron } j \in nl(t) \\ & \qquad\quad\;\; \text{and } r_i^p(t+1) < r_j^p(t), i \notin nl(t) \\ nl(t+1) \quad &= \quad n' \end{aligned} \quad (4.30)$$

After the spreading phase, the final signal r_j of each neuron $\{j\}$ will be calculated by summing up all the reward signals from the target's location distribution:

$$r_j = \sum_p r_j^p \quad (4.31)$$

Based on these reward signals, the robot plans its action by calculating the next position it should reach. Assuming that the robot's position is represented by a group of neurons α in the spatial memory, the next possible position should be among the neighborhood neurons that connect with neurons in α directly. The activity $q_{i_2}^d$ of these neighborhood neurons i_2, which connect with neurons $i_1 \in \alpha$, is computed as follows:

$$q_{i_2}^d = \sum_{i_1 \in \alpha} c_{i_1 i_2} s_{i_1}^r r_{i_2} \quad (4.32)$$

where $s_{i_1}^r$ is the neuron activity of the robot detection (cf. Eq. (4.1)) and $c_{i_1 i_2}$ is the connection weight (cf. Eq. (5.5)).

The higher $q^d_{i_2}$ is, the more desirable it is for the robot to be at this position. We scale $q^d_{i_2}$ to the range of $[0, 1]$ by dividing all the q^d_i with the maximal value $\max_i(q^d_i)$. Theoretically, the distance of reward spread is unlimited, but the strength of the signal decreases exponentially with a constant discount factor, which leads to small gradients at large distances. For decision-making, the robot only evaluates the states around the current location and uses a soft-max function to retrieve those with highest reward value. Neuronal noise would impose a limit at which the gradient towards the goal cannot be evaluated, but in the computer implementation, the limit will occur when the computer cannot distinguish the higher value due to the precision of the double float value. However, during our experiments this situation never appears.

4.3 Environment Grounding via Anchoring Visual Features

Localization and navigation of a robot towards the target person is essential for assistive robots to support patients' daily lives. However, in order to really service patients, more cognitive tasks have to be achieved. How a robot can help a person (especially who is mobility-impaired) to manipulate objects remotely is an important but challenging task. For example, a robot needs to find a blood oxygen measurement and fetch it to a user when the device is required. Since in real life the device can be placed at any arbitrary position, which cannot be pre-programmed, the robot has to analyze information gathered from the environment and determine the location of it. In order to realize this, a robot needs to not only represent

the spatial information of the environment for localization and navigation, but also understand the environment.

Inspired by biological evidences (e.g., visual landmarks help desert ants to return home (Collett et al., 1998)), we consider solving this problem by learning the appearance features of the environment, which can be integrated with the spatial memory. A further feature of our system is therefore developed that by anchoring the appearance features of the environment with the states in the spatial memory, visual associations are linked to specific locations in the map. A robot head camera is used to observe the environment while the robot navigates in the room, which extracts visual information and associates them with the spatial knowledge. These features allow the robot to learn the appearance of the environment to the corresponding neuron in the cognitive map during its navigation, which simulates the visuospatial perception and enables the robot to combine the cognitive tasks of locating and navigating to an object held in memory. Together with the spatial features that describe the location of the object, the visual anchoring allows the robot to achieve complex tasks such as fetching an object by showing an image of it.

4.3.1 Learning Appearance Features

Recalling the spatial memory described in section 4.1.1, the cognitive map contains *states* and *connections* information that represents the features and the corresponding relations between these features. As features of the cognitive map can be presented in arbitrary forms, for example the reading of a laser scanner or position information, combined sensory cues can be integrated into the states. In our case, we extend

the previous model that contains position information ("tells where the robot is") by combining it with visual feature information ("tells what the robot sees") that is observed by the robot's camera. Two kinds of features are presented in the neurons of the spatial layer: (1) the x, y coordinate information on the image from the ceiling-mounted camera, $v^c = \{x, y\}$, and (2) the appearance memory, which resembles the visuospatial perception, based on visual keypoints extracted from the robot's camera $v^r = \{k_1, k_2, \dots\}$.

While the robot is walking during or after map building, the visual stream of the robot's head camera is captured and visual features are extracted. Based on the location of the robot, the neuron i^* closest to the robot's location will be active using the spatial information (cf. Eq. (4.1)). The visual features extracted from the robot's camera will be assigned to $v^r_{i^*}$ when the winner neuron (i.e., i^*) changes:

$$\begin{aligned} k_{\text{new}}\{\cdot\} &= \text{Extract\_Features}() \\ v^r_{i^*} &\leftarrow k_{\text{new}} \end{aligned} \tag{4.33}$$

A buffer is defined for each neuron to store the last 64 visual features when the robot visits the corresponding place, irrespective of its orientation. We use SURF features (Bay et al., 2006) to present the information of keypoints (see Figure 4.5). The reason for choosing these features is because that the SURF features are scale-, shift- and rotation invariant, and they are robust against a change in light conditions while keeping the real-time ability. This is important because the example image of the target object may be different from what is observed while the robot is walking. Each keypoint contains the x, y position of the feature point in the image of

the robot camera and a 64-dimensional vector that represents the image gradients. As a result, the robot learns a memory by associating the extracted visual features with its current location in the spatial memory during navigation. When the robot visits the same place in the map again with a different orientation, the features from the new point of view will be inserted to the same neuron corresponding to this location. This memory is used for locating an observed object by comparing the similarity between the features extracted from an image of the target and the visual features stored in the appearance memory. For details of the SURF features and feature matching please read the original paper (Bay et al., 2006).

A demonstration of learning appearance features is shown in Figure 4.5. While the Nao robot walks in the room, the extracted SURF features from the robot's camera (plotted as red circles in the upper left subfigure) are stored at the corresponding state in the spatial memory (left under subfigure). After that, the robot is able to locate a target object by comparing features of a shown image with the stored visual features.

4.3.2 Object Finding and Path Planning Using Example Image

After the robot learns appearance information of the environment through its explorative navigation, it can determine the position of a target object by features matching of the example image with its visual appearance features stored in the spatial memory. Recalling from section 4.2 that the target position is estimated with the goodness-of-match signals, for the object finding task we redefine the term m_i of the neuron i in the spatial memory by comparing the similarity between v_i^r of

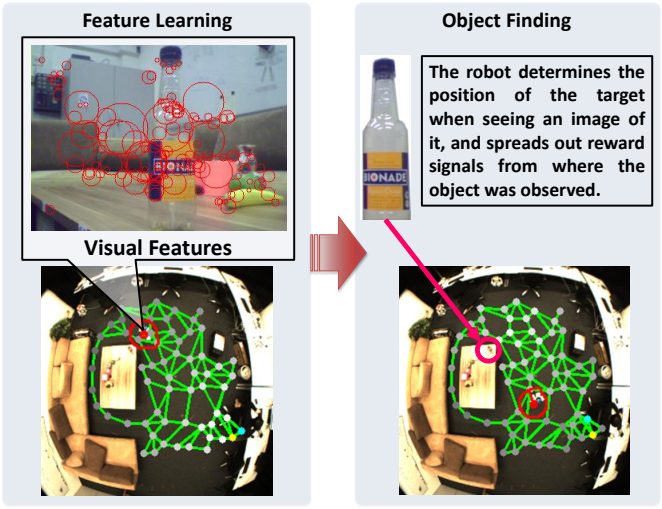


Figure 4.5: Schema of anchoring appearance features

neuron i and the features v_{obj} of the target object as follows:

$$m_i = \text{feature\_match}(v_i^r, v_{\text{obj}}) \tag{4.34}$$

where v_i^r are the learned keypoint features (cf. Eq. (4.33)) and v_{obj} are the keypoint features extracted from the target object. The more visual keypoints of the test object match v_i^r of neuron i, the higher is the activity of this neuron. Since the object may be observed from multiple positions (for example a few cans of soft drinks are placed in the room), a distribution rather than a single position will be labeled as the target. These facilities help the robot to choose a better route to approach the target. If no matched feature is found, $m_i = 0$. Based on these seed signals m, rewards are spread and the

path planning is processed in the same way as described in section 4.2. The robot can then navigate to the corresponding positions and process further tasks such as object grasping.

4.4 Experiments for Map Learning and Navigation

In order to evaluate the performance of our model, we implemented a simulator[1] to demonstrate the map building and the navigation functionality. The graphical user interface of the simulator is shown in Figure 4.6. A test environment is created automatically by clicking the "Generate Map" button, and a rectangle mazelike map is built in the main window where the white area denotes free spaces and the black areas represent the obstacles. Multiple robots can be inserted in the map with different colors, which is visualized as a triangle in the test environment (see the yellow robot).

A cognitive map is built while a robot/person explores the room, which is used for navigation planning. In the simulator, the position of the person can be controlled by moving the mouse inside the test environment through the "Move Person" function, placed randomly by using the "Random Position" function or moved with random velocity using the "Random Velocity" function. To interact with the environment, two sonar sensors for each robot are simulated, which are placed on the front side of the robot and can be visualized by choosing the "Sonar Sensor" option. The robot can be controlled manually by clicking the arrows in the control panel area or

[1]Source code available at: `http://journal.frontiersin.org/file/downloadfile/18094/octet-stream/Data%20Sheet%201.ZIP/13/1/56899`

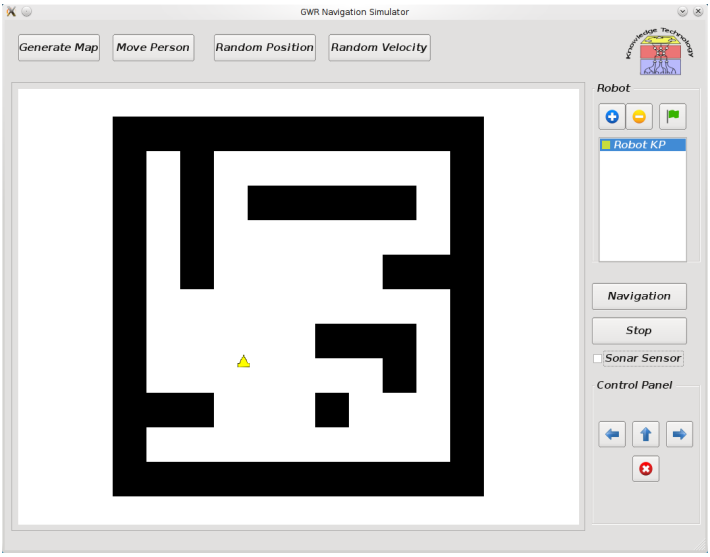


Figure 4.6: User interface of the simulator

autonomously by clicking the navigation button. When the robot reaches the target position or the stop button is clicked, the robot stops moving.

4.4.1 Map Building through Exploring the Environment

Figure 4.7 shows a test scenario for map building. At the beginning of map learning, the spatial memory is initialized with two neurons linked with each other with a connection (see Figure 4.7(a)). While the agent explores the environment, the closest neuron to the agent (winner neuron in red)

and the second closest neuron (second winner neuron in green) will be selected. The states of the winner and its neighborhood neurons will be updated to the person's position using the Hebbian-learning rule, and new neurons will be inserted when conditions match (see Figure 4.7(b)). Through repeating the map updating and neuron adding, the spatial memory will grow automatically and cover the entire free space in the room when all of the free space is visited by the person (Figure 4.7(d)). Since neurons are updated based on the agent's exploration and the agent cannot enter the space of obstacles (i.e., the black areas) in the simulation, only the traversable areas are learned and the obtained map represents the free space properly with a safe distance to obstacles. Moreover, because the free space clusters with a uniformly distributed network, it helps the robot to plan the optimal navigation trajectory.

4.4.2 Robot Navigation

The robot navigation uses the learned map for path planning while avoiding obstacles in real time, based on feedbacks of the simulated sonar sensors. Based on the spatial knowledge, a goal-directed plan towards an arbitrary goal position is computed. As shown in Figure 4.8(a), a reward signal spreads out from the neuron closest to the target position (labeled as the red dot) through the entire map with an exponential decrease (cf. Eq. (4.29)). Each neuron obtains a reward value, which is visualized with the brightness of the neuron (see Figure 4.8). The brighter the neurons are, the higher the reward they have. During the navigation, the robot looks for the state with higher rewards (i.e., neurons with higher brightness), according to its current state, and activates action memories that are stored

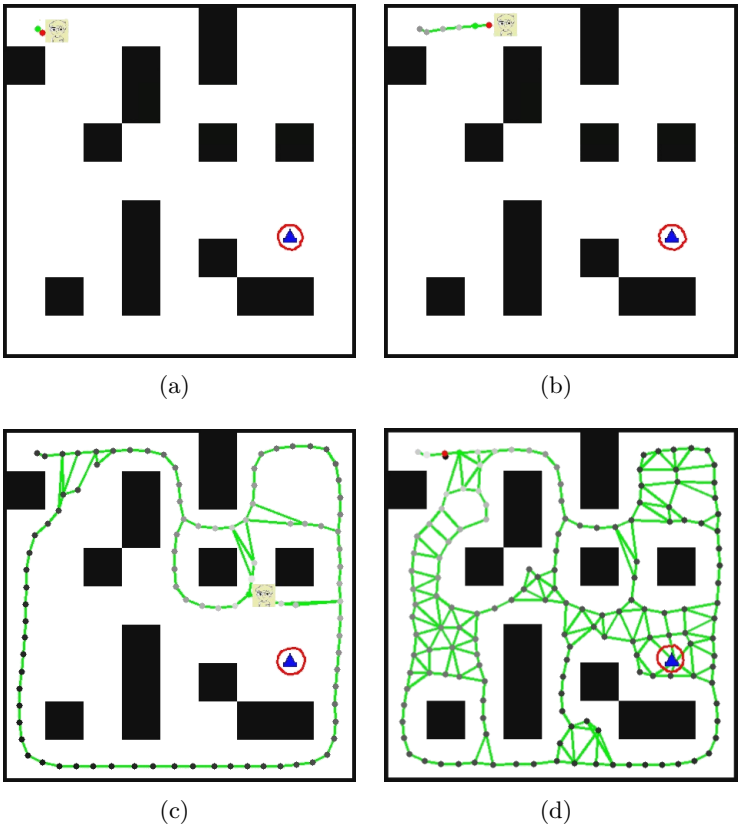


Figure 4.7: Cognitive map learning in the simulator
(a): the initial status of the cognitive map, the person's position is simulated with mouse input. Black blocks denote obstacles in the environment; (b) and (c): a cognitive map is growing based on the person's position; (d): a completed cognitive map of the traversable area is built, and the robot starts navigating to the target position labeled in red. The radius of the red circle denotes the activities of the DNF.

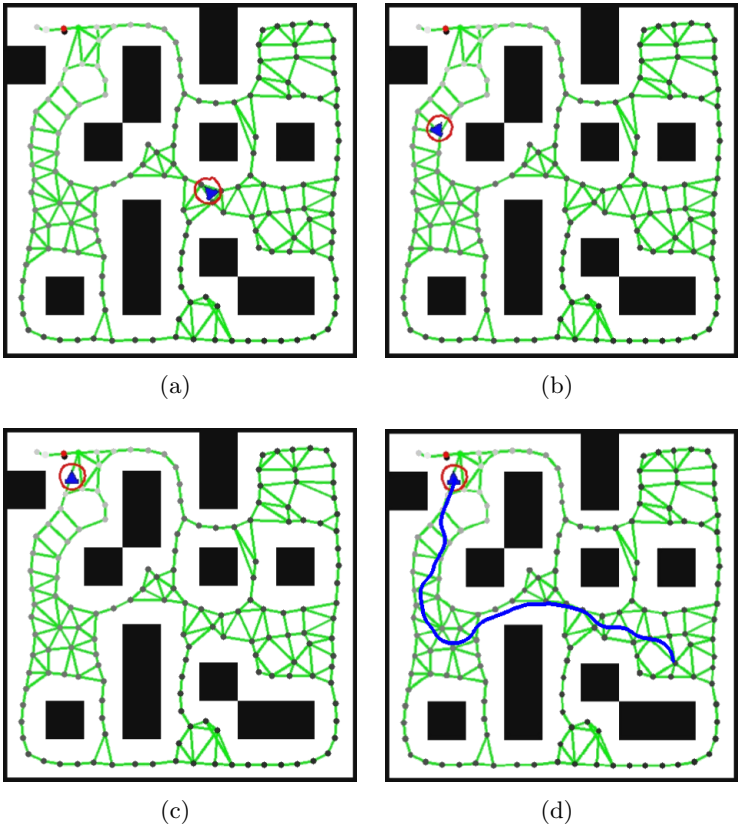


Figure 4.8: Robot navigation using the cognitive map in the simulator

(a) - (c): Different position during robot navigation. (d): Trajectory of the entire navigation process. The red circle denotes the activities of the DNF with different radius. The larger the radius, the higher the activity of the neuron.

in the corresponding connection between the current and the desired states.

The action memory, together with the robot's state and the desired state information, is sent to the action layer to compute the motion signals for navigation. As multiple action memories are active at the same time, these signals are merged in the DNF, and the activities of the neurons of the DNF are updated (cf. Eq. (4.20-4.22)), which builds up an activation bump to determine the desired orientation for navigation. Figure 4.8 illustrates a demonstration of robot navigation after the map building. During the navigation, the robot plans actions for the next step according to the reward signals of the map and its actual state. The robot moves towards states with higher rewards (i.e., neurons with brighter color) in order to reach the goal position. New actions are calculated based on the activated action memories, which are used to update the neural field for motion control. We visualize the output of the DNF with a red circle surrounding the robot's position. The basic radius of the circle is set to 15 pixels where activations are zero, and the larger the radius, the stronger activations the corresponding neurons have. By repeating the decision-making, the robot updates the DNF, which indicates the desired orientation of navigation at different positions (See Figure 4.8(a)-Figure 4.8(c).) until the robot reaches the target position. Because the robot's action is computed based on multiple action memories rather than a single command, the robot does not follow lines of the map and provides a smooth movement. Finally, the entire trajectory of this navigation is shown in Figure 4.8(d). Since the robot only concerns the action of the next step based on its current state,

this dynamic planning simplifies the trajectory computation and enables the robot to reconfigure its route rapidly when the environment changes.

4.4.3 Experiments in a Real Environment

In order to test the navigation system in a real-life environment, experiments of map building and robot navigation are conducted in a laboratory, shown in Figure 4.9. In order to resemble a cluttered home environment, furniture such as a tea table, two sofas, as well as a few plants are placed in the room. Similar to the localization, the ceiling-mounted camera is used to locate the person's position and the robot's position. While in the simulator, the position of the person and the robot are estimated with a Gaussian distribution using the mouse input; the position of the person is represented by the particle filter of the localization model. When the target person is moving in the room, the position is updated and used to learn the spatial information. Because a person does not usually collide with furniture, the position that he or she can reach will be considered as free space for the robot.

A Nao robot is chosen as the experimental platform in our work. The Nao robot is a programmable child-size humanoid robot, whose hardware was developed by Aldebaran Robotics. It has two arms and two legs with 21 to 25 degrees of freedom, and has an inertial measurement unit with gyroscope and accelerometer. 8 pressure sensors and 2 bumps are fixed to the leg and foot of the robot to control the walking behavior, and 2 pairs of ultrasonic sensors are fixed to the chest of the robot's body to measure the distance to the obstacles in front of the robot. The robot is also equipped with 2 loudspeak-

Figure 4.9: Test environment of the robot navigation

ers for audio playing and text-to-speech synthesis, as well as 4 microphones for voice recognition and sound localization. 2 HD cameras are installed on the robot's head to percept the visual information of the environment using computer vision techniques. Using its onboard Linux system, the robot is able to process complex tasks such as speech recognition, text-to-speech, object tracking, grasping, etc. The robot can walk autonomously and turn flexibly during walking, and is able to adapt its walking behavior to a different ground surface based on sensor feedbacks. In our experiments, we defined three actions: walking forwards, turning left, and turning right.

One of the test cases is shown in Figure 4.11. As we can see, similar to the experiments in the simulator, the map is first initialized as two neurons connecting with each other with a

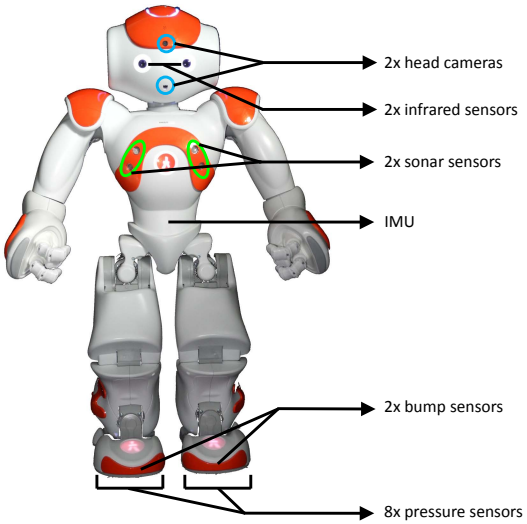
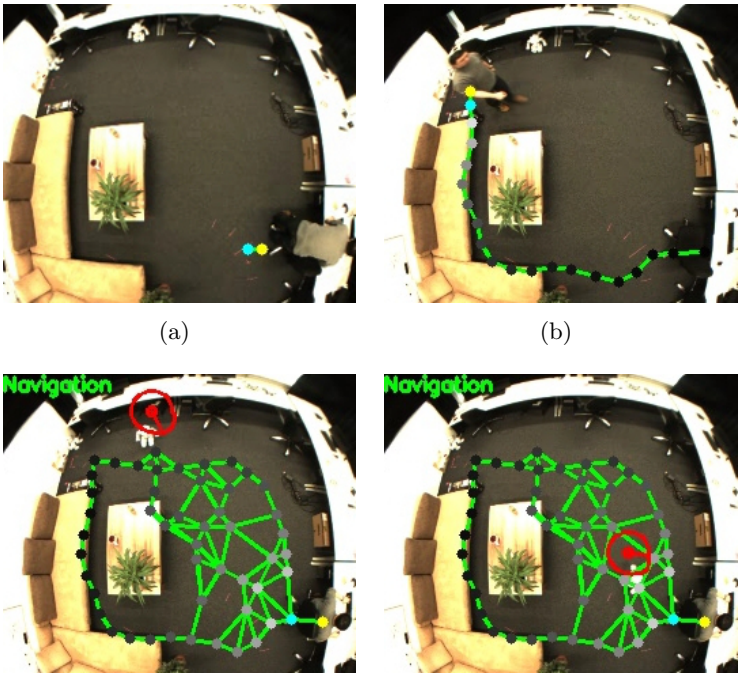


Figure 4.10: The humanoid Nao robot

link (see Figure 4.11(a)). When the person moves, the map adapts itself based on the observation of the person's position, and if the person visits a new position in the room, new neurons will be added autonomously (Figure 4.11(b)). The map grows and converges when most of the traversable area is visited by the person 4.11(c). The brightness of neurons in the map denotes reward signals spreading out from the target, i.e., the person's position. During robot navigation, the robot determines the next action according to its position and the reward signals of the map. The desired orientation is computed based on the activities of the DNF, which are visualized as the red circle around the robot. The short red bar with different orientation shows the estimated direction of the robot with



(a) (b)

(c) (d)

Figure 4.11: Cognitive map learning via observing a person's movement

The DNF is stimulated and produces the desired robot orientation, which is visualized by a red circle surrounding the robot. The estimated orientation of the robot is displayed with a short red bar from the center of the circle. Based on the estimated and desired orientation, the robot controls then its walking direction and approaches the person.

the particle filter. The walking direction of the robot is calculated based on the estimated and desired orientation, which guides the robot to walk autonomously towards the person. During robot's navigation, two sonar sensors detect obstacles that help the robot to avoid any danger area in a reflexive way, which will be described in the following chapter.

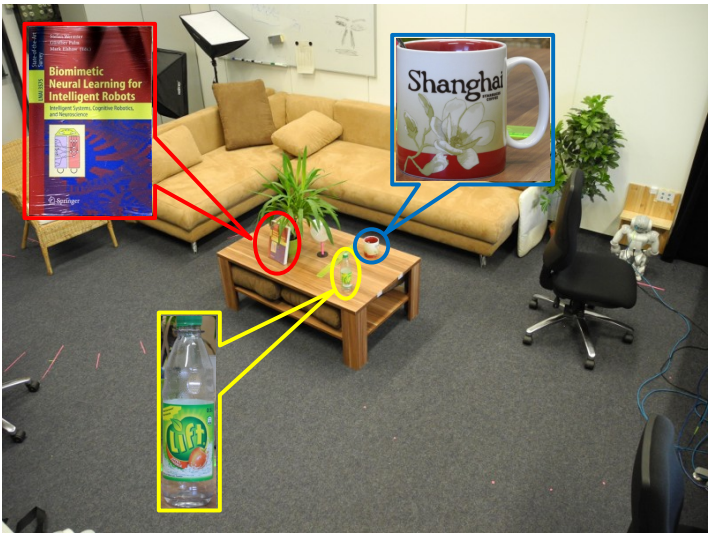


Figure 4.12: Test environment for object finding
Furniture (coffee table, sofas, etc.) is placed in the laboratory to simulate a real-home scenario. We placed several objects (a mug, an empty bottle and a book) on the coffee table to let the robot memorize them while it is moving in the room. After finishing the learning, the robot will be requested to find the object by showing it an image of the target object.

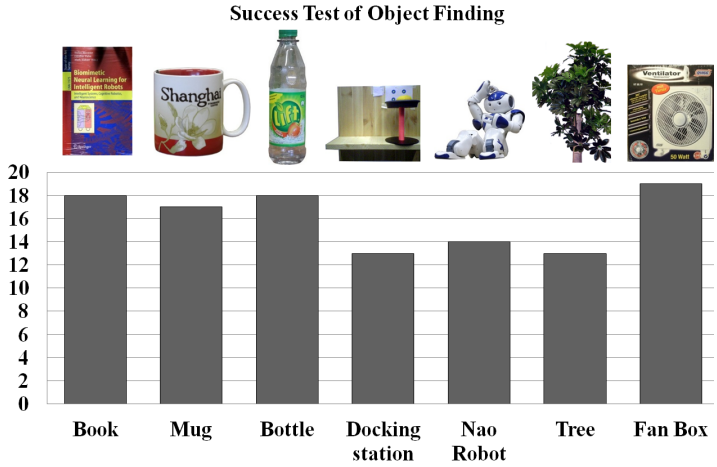


Figure 4.13: Result of object finding

4.4.4 Experiments for Object Finding

To evaluate the performance of object finding, experiments are conducted in a really home-like laboratory. Several objects are placed on the coffee table for testing feature anchoring, including a book, a cup, and a bottle (see Figure 4.12). After the cognitive map is built via observing a person's movement, the robot will firstly explore the room itself actively. The visual information of the environment around the robot is memorized while the robot is walking, and the extracted visual features are associated with the corresponding neurons in the map. After that, we show an image of an object to the robot, and the robot searches its visual memory to find the states that contain the visual features resembling the features of the target object.

Twenty experiments are conducted for locating different objects. Objects are placed in different positions each time and are observed by the robot during navigation. After learning of the appearance of the environment during the robot's walking, we check if the robot can find the correct position of each object by showing a picture of it. The object is considered to be identified correctly if the distance between the target node, i.e., the neuron with the highest reward in the spatial memory (see the brightness of neurons in the map), and the object is smaller than 30 pixels.

A demonstration of these experiences is shown in Figure 4.5. According to matching features extracted from the shown image and the features v^r of neurons (cf. Eq. (4.33)), reward signals m are set in the spatial memory, and they are spread across the entire network (cf. Eq. (4.28)-(4.31)). Then, based on these reward signals, as well as the robot's actual location, the robot decides its navigation behavior iteratively until it reaches the object's position.

The results of the experiments are summarized in Figure 4.13. As we can see, the success rate varies with the objects. For example, the docking station has the lowest success rate because the docking station has a simple structure and few features can be extracted. Due to the constraints of the robot's hardware, it can provide images with only 10 frames per second (fps), and features may be missing because of the image blur, which distracts from feature matching and object finding. On other hand, the book and the bottle can be localized easily because the cover of the book and the logo of the bottle contain rich, sharp features, which can be extracted and detected. In addition, poor lighting conditions could influence

the experiments and distract from feature learning. When the image is observed under a shadow, the features of the object may not be learned (e.g., the tree in Figure 4.13 with dark color) and may affect future detection as well.

4.5 Discussion

In this chapter, we presented a neurocognitive model for robot navigation based on spatial knowledge learning. Our neurocognitive architecture models the spatial context, the reward signal, the decision-making, and the action response as a whole. The spatial knowledge of the environment is modeled with an internal representation, i.e., the cognitive map, which allows the robot to select actions dynamically and to adapt its strategy when the environment changes. Not only the state transition but also the actions corresponding to them are represented on the map. In order to accelerate learning and to avoid possible danger caused by the robot's active exploration, the map is built by observing the movement of a person using our localization method with a ceiling-mounted camera (Yan et al., 2011). This design has been chosen in the context of a setup that uses a small socially assistive robot as a communication interface to the person (KSERA).

Our model meets the challenge of applying a neural system in realistic settings and uses a humanoid Nao robot as an experimental platform. Other than pure simulation models that simplify the environment or model the system dynamics exactly without considering noise, our robot needs to navigate by finding a moving direction among 360° in our home setup, which requires further refinement to match the requirements

of a real application. For example, in order to obtain a continuous robot control, our model represents the individual action for each state transition to get high control accuracy with minimal memory requirement and uses a population code to represent the state in a probabilistic manner. Using this distributed representation, multiple state transitions are active at the same time, and the corresponding action signals are merged via a ring-form neural field. This results in a smooth and continuous action control, where actions are generated that did not occur during map learning. Moreover, the cognitive map does not need to predefine its state space because it adapts itself using latent learning while a person explores the unknown environment. As no goal is needed here, the path planning is not learned for a specific target, which permits flexible navigation behavior towards an arbitrary possibly moving target.

The ceiling camera presents an affordable and minimal intrusive solution to localize a person anywhere within a larger room, even when the small robot cannot directly see the person. The camera supplies high-level visual input about the robot location in allocentric coordinates, which bypasses the need for learning a visual system that localizes from the robot's camera image, as has been done by Wyss et al. (2006). The localization input, which comes via particle filters (Yan et al., 2011), is compatible with distributed neural coding (Deneve, 2005; Huang and Rao, 2009; Wilson and Finkel, 2009). The map building itself resembles "latent learning", where there is no task done during the exploration of the room by a person. When the navigation task is active, the cognitive map, together with the reward signals from the target position, com-

poses a model-based reinforcement learning, which guides the robot in maneuvering to the target. As the robot control is not programmed but learned, the model can be applied easily in different rooms without camera calibration.

According to the cognitive maps introduced by Tolman (1948), animals and humans use internal spatial representation for path planning rather than purely following environment sensing during navigation. Our system fits this concept in the sense that the robot uses the internal state representation for path planning and navigation instead of reading the sensor input directly. The system consists of multiple layers of neural networks, which combine map building and localization with planning and navigation. The spatial memory is represented by a GWR network with self-organizing learning, which is related to the dynamic (e.g., cell growth (Eriksson et al., 1998)) place cells in the hippocampus (Gorchetchnikov and Grossberg, 2007). The ceiling-mounted camera simulates a high-level visual perception model not only for robot localization and map building but also for person detection from an arbitrary position in the room, which replaces an internal perception model.

The navigation is planned in real time on the basic of the reward signal spread through the spatial memory network from the target position. Since there is activity away from the robot's actual location, this could correspond to the activation patterns observed in hippocampal cells, which do not strictly encode the current position of a rat but represent places the rat is considering visiting in the near future (Van Der Meer and Redish, 2010). Different actions weighted with their corresponding activities are fused to generate actions that are

more precise and robust with respect to the discretization of the grid of the spatial map. As a result, our model is able to perform flexible navigation to reach an arbitrary target without pre-training with a fixed goal position.

A method for environment grounding by anchoring visual appearance features to achieve complex cognitive tasks is also presented in this chapter. The appearance is represented by the visual features and associated with the corresponding states in the spatial memory. When an object is required, the robot is able to locate the position of the target object by comparing its visual appearance memory with the features of the target. This allows the robot to complete complex tasks by showing it an image of the object, such as fetching this object, and helps to extend the functionality of socially assistive robots. The method is evaluated using real scenarios, and complex tasks such as environment learning, object finding, approaching, and retrieving are demonstrated successfully.

Because the appearance memory is discreet (i.e., extracted when the robot visits a node in the spatial memory), the resolution of the tested object depends on the distance between the robot and the object at that time. The appearance feature is represented with SURF features that are rotate, shift, and scale-invariant. This is necessary because the example image shown by the user may not fit exactly with the stored appearance memory (e.g., due to the different distance and pose). Moreover, the closer the robot to the object, the higher the resolution of the object in the camera, which provides more features for feature matching. This permits the robot to locate the object correctly on the spatial memory. When the robot revisits the same place, new visual features of the po-

sition are learned in the corresponding node, which helps the robot learn the environment from different views. Because the target object may be observed from different positions, the target location can be represented by multiple states in the spatial memory. With this distributed target representation, the robot is able to plan an efficient trajectory according to its current position.

The appearance features can be used not only to locate the object but also to determine the location of the robot itself. In future work, we will explore the robot localization based on the visual features captured from the robot's camera. Moreover, the localization using the robot's camera is useful to extend the working space of the robot when an ambient sensor is not available. Comparing the object-finding scenario where the object and the environment are represented by scale-invariant features to the robot localization using appearance landmarks, the features need to be scale variant in order to be sensitive to distance changes. The major challenge here is that when the position is observed from different view angles, the visual appearance may be totally different. How to find the association of different appearances and identify the same object will be an interesting research topic for future development.

Chapter 5

Adaptive Learning of Dynamic Environment

The change of environment is common in the real-life space, since furniture may be replaced, objects may be moved, and persons may go to different places. In this case, a robot needs to consider all these factors and avoid possible collisions during navigation, which increases the difficulty of robot navigation compared to a laboratory environment. A robot should detect these changes and react to them quickly in order to avoid danger situations to survive in the real scenario. Because the existing map does not fit the current situation anymore and also to keep the obstacle avoidance intelligent and efficient, an adaptation learning of the environment and a quick replanning based on the new map is required. Moreover, quick replanning is important for coordination of the mobile behavior of multiple robots that can be used for teamwork e.g., robot soccer, swarm, etc.

In this chapter, we present an interaction mechanism that provides a reflex behavior to protect the robot passively during navigation and adapt the navigation strategy online based on the reflex feedback. A simple obstacle avoidance method is developed based on two sonar sensors for detecting objects close to the robot to maneuver the robot in a danger area. Based on the detection results of the obstacles, an adaptation of the spatial knowledge will be conducted by adjusting the connection weights in the spatial memory, which helps the robot to improve its navigation behavior. Considering that connection weights $c_{i_1 i_2}$ in the spatial memory play a central role in path planning (cf. Eq. (4.29)), which indicates how "easily" the robot can follow that link), two parts of connection weights with distinct functions are defined as follows:

$$c^{r^*}_{i_1 i_2} = c^{g}_{i_1 i_2} + \sum_{r} c^{t_r}_{i_1 i_2}, \qquad r = 1, 2, \ldots;\ r \neq r^* \tag{5.1}$$

where $c^{g}_{i_1 i_2}$ denotes global weights and $c^{t_r}_{i_1 i_2}$ denotes the temporary connection weights of the robot r with respect to the states of other robots. The higher the connection weight is, the easier can the robot access this connection. When the robot detects obstacles during navigation, the corresponding connection will decay until the weights decrease to zero and the connection will be mentally cut off. The global connection weights represent the status of the traversable environment based on the robot environment interaction. Based on the feedback of the sonar sensors, the robot detects obstacles and adapts the global weights correspondingly. The temporary connection weights are computed based on the localization of multiple robots. We consider that for each robot, the other robots are recognized as moving obstacles. Therefore,

for multiple robot navigation, the awareness of other robots with respect to one robot is useful to coordinate pro-actively their moving behavior. In the following sections, we describe the mechanism of obstacle avoidance as well as learning both kinds of connection weights in detail.

5.1 Method

5.1.1 Reflex-like Obstacle Avoidance

Beside the path planning that guides the robot strategically towards the target, an efficient, real-time obstacle avoidance method is useful to adapt the robot's behavior for avoiding danger situations, which are not estimated during planning or appear suddenly due to environment changes. Therefore, we developed a simple reflex-like method for obstacle avoidance based on the sonar sensor signals of the Nao robot. For each step, we compute a signal $G(s_1, s_2)$ according to the sonar sensor signals s_1 and s_2 with a nonlinear function:

$$G(s_1, s_2) = \frac{a}{1 + e^{-b(s_1+s_2-c)}} \tag{5.2}$$

where constant parameters a scales the output signal, b adjusts the sensitivity of the output according to the input signals, and c controls the threshold of the signal response. The further the robot is to an object, the larger is value s_1 (or s_2). Two kinds of obstacle avoidance strategies will be triggered based on $G(s_1, s_2)$. When $G(s_1, s_2)$ is below a threshold, i.e., $G(s_1, s_2) < \gamma$, the robot will turn away from the obstacle based

on the sonar signals:

$$\Delta O = \begin{cases} -c_o & \text{if } s_1 > s_2 \\ c_o & \text{else} \end{cases} \tag{5.3}$$

where c_o is a defined turning speed of the robot (in our experiments $c_o = 0.3$. When $G(s_1, s_2) < 0.8\gamma$, the robot will walk slowly backward while turning. The reflex-like obstacle avoidance helps the robot to keep a safe distance to obstacles, where the feedback from sensors $G(s_1, s_2)$ is transferred back to the spatial memory to adapt the connection weights, which resembles an "error-based" learning of the environment. When an obstacle is unexpectedly detected, the corresponding connection will be weakened, which prevents the robot to choose this connection again in the future. Details of learning these connection weights will be described in the next section.

In order to incorporate reflex-like obstacle avoidance behavior in the simulator, we model two sonar sensors on the front side of the robot that resemble the sonar sensors of the Nao robot. Figure 5.1 shows the detection range of the simulated sonar sensor, where the green area shows the detection range that objects are close to the robot and the red area shows the detection range that objects are very close to the robot. Mathematically, a threshold γ is defined for the range detection, where the green area is active when $G(s_1, s_2) < \gamma$, and the red area is active when $G(s_1, s_2) < 0.8\gamma$. In the simulator, the sonar area can be visualized by choosing the "Sonar Sensor" option.

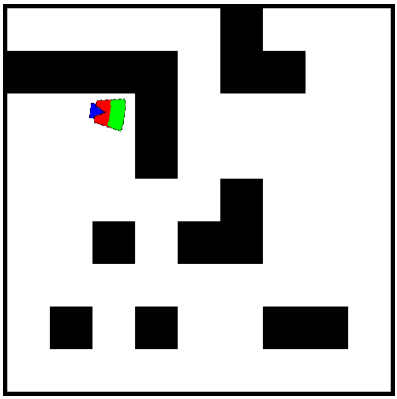


Figure 5.1: Simulated sonar sensor

5.1.2 Learning Global Connection Weights

The global connection weights represent the traversability of the static environment. When objects are moved, or a new obstacle is put into the environment, the previous traversable area will be blocked. In order to adapt the robot's path planning when the environment changes, an "error-based" learning method of the global connection weights $c^g_{i_1 i_2}$ is processed based on the feedback of the obstacle avoidance. A new obstacle is detected when the sensor feedback is smaller than a threshold, i.e., $G(s_1, s_2) < \gamma$. Then, the connection $i_1 {i_2}^*$ with the highest value of $s^r_{i_1} q^d_{i_2}$, which plays a central role in decision-making for navigation, will be updated as follows:

$$ {i_1 i_2}^* = \arg\max_{i_1, i_2} s^r_{i_1} q^d_{i_2} \tag{5.4} $$

$$ \Delta c^g_{i_1 {i_2}^*} = \tau_1 (G(s_1, s_2) - c^g_{i_1 {i_2}^*}) \tag{5.5} $$

where τ_1 is a learning rate. Here, we use the "winner-takes-all" strategy that each time only one winner connection will be weakened. When the robot approaches an obstacle, the connection weight $c_{i_1 i_2 *}$ will be decreased. The adaptation of the connection weight affects the reward spreading as reward signals will be strongly weakened through this connection (cf. Eq. (4.29) and Eq. (4.32)). By repeating this process, the contribution of this connection to the navigation becomes smaller until a new winner connection appears which guides the robot with other motion signals.

Since obstacles may also be removed from the environment, we consider the following method for recovering the connection weights in this case. When $G(s_1, s_2) > \gamma$, all the connection weights around the current robot position are adapted with Eq. (5.6):

$$\Delta c^g_{i_1 i_2} = \tau_2 (G(s_1, s_2) - c^g_{i_1 i_2}) \qquad \forall s^r_{i_1} > e \tag{5.6}$$

where e is a threshold of the distance and τ_2 is a learning rate smaller than τ_1.

5.1.3 Learning Temporary Connection Weights

Other than the global connection weights that learn through interaction with the environment, in other words learn through experience from the past, the temporary connection weights here present positions of moving robots with their active sense and help the robots to prevent collision with each other in advance. Because the others can be regarded as moving obstacles from the view of a robot (e.g., robot r), inhibitory connection weights around them will be built in the map with respect to robot r. These negative connections close to the other robots

will affect the planning of robot r so that it will avoid moving to the corresponding areas. The building of these inhibitory weights is inspired by the reward spreading while the iteration steps are constrained to keep the planning efficiency. The computation can be described as follows:

1. Initialize all weights of $c^{t_r}_{i_1 i_2}$ of the robot r as zero.

2. Determine the states of other robots i^* with the highest activity (cf. Eq. (4.26)) in the spatial layer. For each of them, we add a seed with an initial weight $c^t = -1$ where r denotes the index of the distracting robot. The index of the current state will be appended to a list $nl(0)$:

$$nl(0) \leftarrow i^* \tag{5.7}$$

3. Spread the weights with a limited iteration loop p. For each loop, run:

   (a) Update the connection weights $c^{t_r}_{i_1 i_2}$ for connections that link to neurons in nl:

$$\Delta c^{t_r}_{i_1 i_2} = c^t, \text{for } c^{t_r}_{i_1 i_2} > c^t \wedge (i_1 || i_2 \in nl(t)) \tag{5.8}$$

   (b) Update the list of nl:

$$\begin{aligned} n' &\leftarrow i \quad \text{if } i \text{ connects with neuron } j \in nl(t) \\ & \qquad \text{and } r^p_i(t+1) < r^p_j(t), i \notin nl(t) \\ nl(t+1) &= n' \end{aligned} \tag{5.9}$$

   (c) Decrease $c^t = \lambda^t c^t$ with a constant factor λ^t.

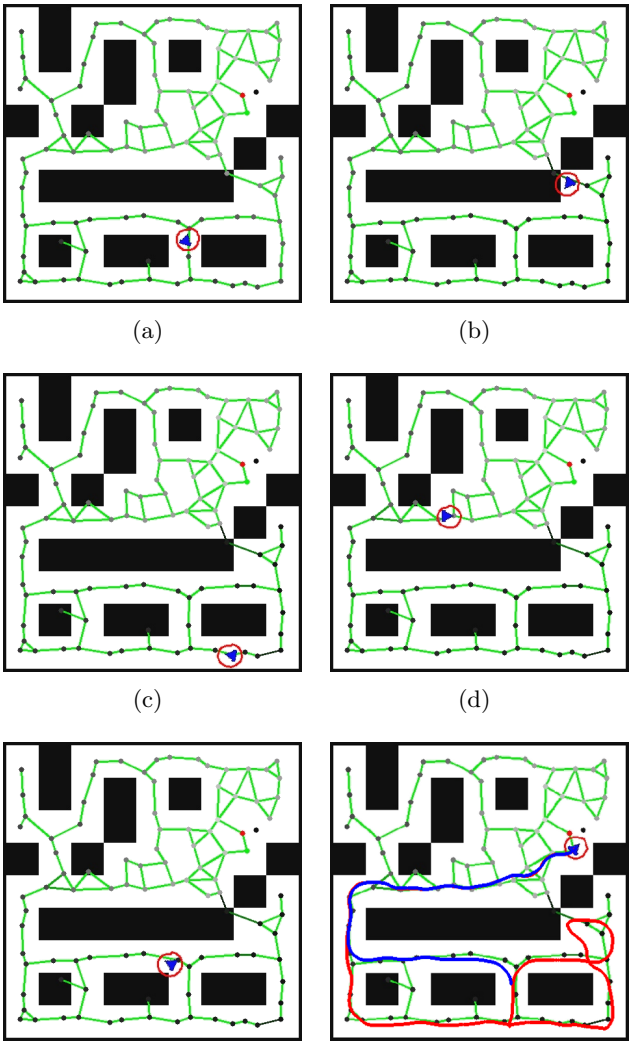
The temporary connection weights together with the global connection weights will be used for path planning and further motion control. Because the decision-making of navigation is based on dynamic planning, the robot only concerns the next action for each step. When the connection weights change, the next action will be adapted without sophisticated reconfiguration of the entire route, which helps the robot to adapt its navigation behavior efficiently.

5.2 Experimental Evaluation

In this section, we describe experiments of robot-environment interaction. Similar to the previous chapter about robot navigation, the experiments here are first conducted in the implemented simulator, then in a home-like lab environment. We first demonstrate how a robot reacts when a connection is blocked and how it adapts its behavior after the interaction. Then, we also show how robots navigate towards a target while preventing collisions with other robots actively. Experiences in a real-life scenario are being presented and evaluated at the end of this chapter.

5.2.1 Adaptive Learning through Interaction

The algorithm of robot reaction behavior and adaptive learning of the environment is firstly experimented in our simulator (see Figure 4.6). A demonstration of map adaptation based on reactive behavior is illustrated in Figure 5.2. At the beginning of the experience, a robot (the blue triangle) is placed in the position shown in Figure 5.2(a) and the target position is defined by the position of the red dot on the map. After the map



(a) (b) (c) (d) (e) (f)

Figure 5.2: Map adaptation via interaction with the environment

building, the traversable space of the maze is represented by the spatial memory that is visualized as the GWR network. As we can see, there is a connection through the wall that makes the robot "believe" that there is a corridor through the obstacle. Obviously, this ghost connection provides a shortcut towards the target, and the robot attempts to move towards it according to the results of reward spreading. However, when the robot arrives at this area (as shown in Figure 5.2(b)), the wall will be detected by the sonar sensors which alarms the robot that this area is not blocked. The reactive behavior will then be triggered, and a maneuver action is done to avoid colliding with the wall.

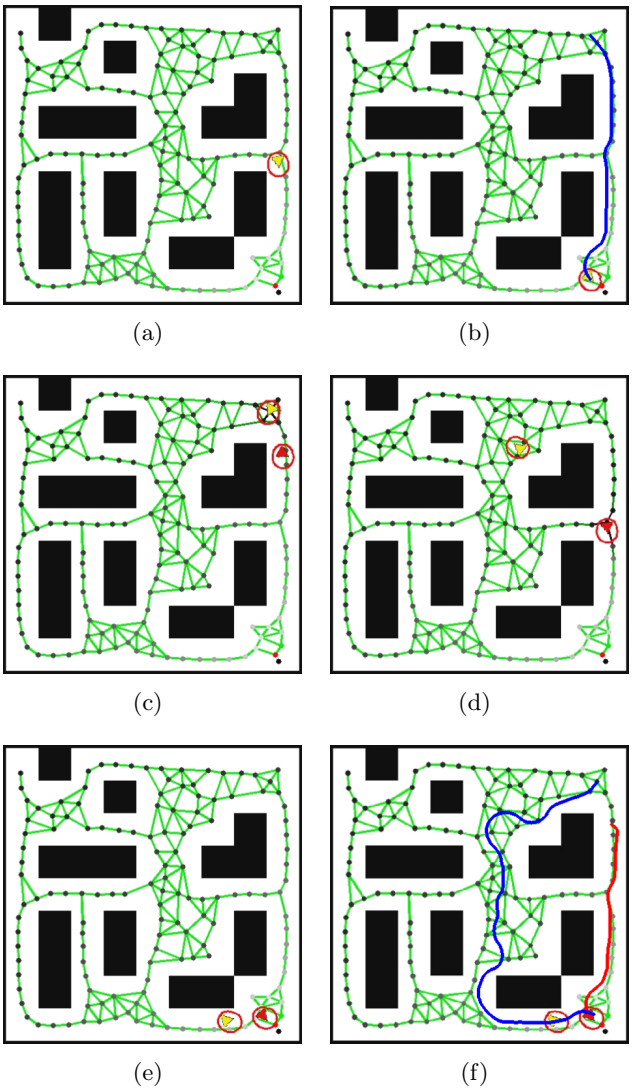
Meanwhile, based on the feedback of the sensor, the connection weights of the corresponding link through the wall is decayed (see Figure 5.2(b) that the color of the ghost connection becomes darker). The decrease of connection weights affects the reward spreading that the reward signal will be dramatically weakened by transferring through this connection hence the path planning will be reconfigured. After this change, the robot makes a detour and finds an alternative route to reach the goal position (see Figure 5.2(c) and Figure 5.2(d)). After this first experiment, we place the robot back to the starting position and test the same task again. Since the robot adapts the environment knowledge using the obtained experience from the previous trail, the ghost connection is strongly weakened to a very low value (see Figure 5.2(b) that the connections close to the wall turn into black). This connection is then "ignored", which can be seen when, after the start of the navigation (Figure 5.2(e)), the robot chooses a distinct path immediately. The trajectories of both trails are

visualized in Figure 5.2(f) where the initial trail is marked red and the second trail blue. As we can see, the robot reaches the target position much faster the second time than the first time. Consequentially, the adaptation does help the robot to avoid danger areas pro-actively and optimizes the path towards the goal.

5.2.2 Coordination of Multiple Robot Navigation

Figure 5.3 shows a test case of dynamic planning of multiple robots' navigation. In order to show the different navigation behavior, we first conduct the task with a single robot. As visualized in Figure 5.3(a), a robot (a) is placed in the starting position and navigates to the target that is marked with the red dot on the map. Based on the spatial memory, the robot walks directly southwards and reaches the target easily. The navigation trajectory for this is shown as the blue line in Figure 5.3(b). Then, we repeat the experiment and add another robot (b), which is placed on the path between the robot (a) and the goal position (see Figure 5.3(c)). If the robot (a) chooses the same route as before, it may collide with the robot (b) and trigger the reactive behavior to prevent the collision. However, in this case, our method helps both of the robots to detect this danger in advance and coordinate their path planning to avoid this situation pro-actively.

As robots can determine their positions on the map, each of them is aware of the location of the others and can treat them as moving obstacles. Using the method described in section 5.1.3, inhibitory connection weights will be added to the current map (see Figure 5.3(c) from the opinion of the red robot (b) and Figure 5.3(d) from the view of the yellow robot



(a) (b) (c) (d) (e) (f)

Figure 5.3: Navigation planning of multiple robots

(a). When the connections are occupied by the other robot, the decayed connection weights will prohibit the reward spreading through them which helps the robot to replan its navigation before the potential collision occurs. For this reason, the robot (a) chooses an alternative route towards the target. As we can see, the possible collision is solved successfully based on the learning of the temporary connection weights. The trajectories of both robots are illustrated in Figure 5.3(f) where the blue line is the trajectory of the robot (a) and the red one of the robot (b).

A further advantage of this method is its excellent real-time planning ability. The dynamic change of the environment is represented by the adaptation of connection weights, which can be processed easily based on the obtained location of the robot in the spatial memory using the localization model. After this computation, the path planning for each robot is done individually without further consideration of the other robots. The computational complexity of the path planning for multiple robots could be downscaled to the same level for a single robot planning. Using this dynamic planning during navigation, robots only need to decide the action for the next step without global trajectory replanning, which helps them to update their path according to new positions of the others quickly. Consequently, the described method provides an efficient way of multiple robots dynamic planning.

5.2.3 Experiments in a Real Environment

We use a small humanoid Nao robot for evaluating the adaptive learning of the environment during robot navigation. As shown in Figure 5.4, the robot is equipped with various sen-

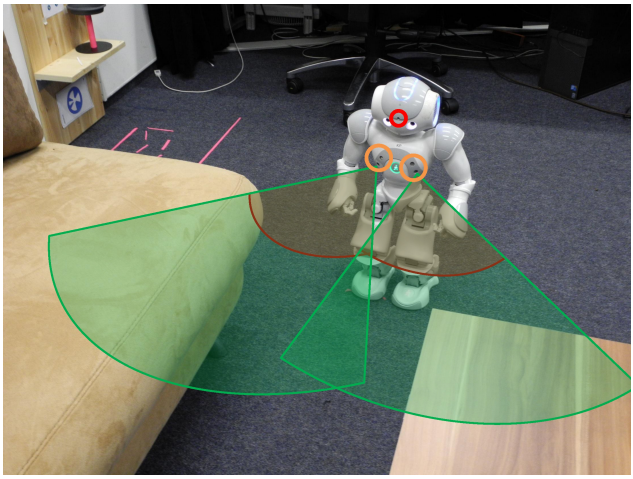


Figure 5.4: The NAO robot's sensors and their ranges

The robot uses a head camera (red circle) and two sonar sensors (orange circles on the chest). The detection ranges of the sonar sensors are illustrated in red and green. Within the dark red line, the robot only knows that an object is present. Here, the robot cannot see the open space in front of it.

sors, among them we use one camera in the head to observe the environment and the two pairs of sonar sensors for detecting obstacles during walking. Both sensors can detect the distance to obstacles robustly between 30 and 80cm. A higher sensor value indicates a larger distance. Because the robot cannot measure the distance when the obstacle is closer than 30cm (see the red area in Figure 5.4), the robot will not be able to maneuver through a narrow corridor for safety reasons.

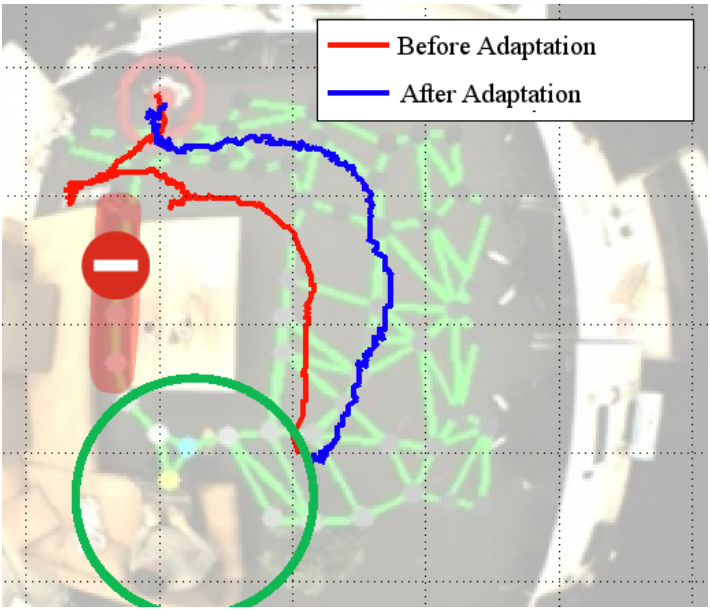


Figure 5.5: Trajectories of robot navigation before and after the map adaptation

The green circle denotes the target position. The area marked in red with a red ban sign shows the narrow path of 25 cm width, where the robot cannot walk through. The red and blue trajectories show how the robot navigates before and after the map adaptation.

Figure 5.5 illustrates a test scenario of adaptive map learning through interaction with the environment. The robot should navigate to the target position based on the map learned from observing the movement of the person. Among the connections in the map, the route marked red seems better for

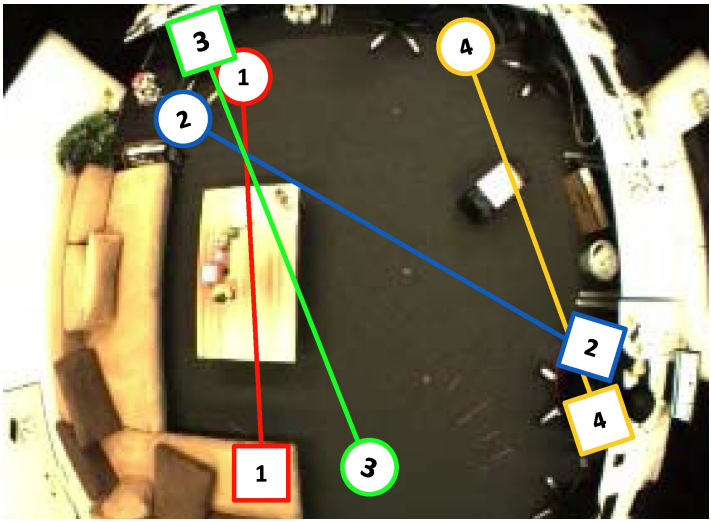


Figure 5.6: A subset of experiments for map adaptation

navigation, and the robot attempts to choose this way. However, some routes learned from observing the person may be difficult for the robot to walk through. For example, although the narrow path with a width of only 25cm (shown in Figure 5.4 and Figure 5.5 between the sofa and the coffee table) is no problem for a person to walk through, it is hard for the Nao robot because of the detection range of the sonar sensor. When the robot arrives at this area, obstacles will be detected, which triggers the reactive behavior of the robot and the corresponding connection weights in the spatial memory is decreased. At a certain point, when the connection weight is small enough, the robot's behavior is changed and the newly generated plan guides the robot with an alternative route. As the trajectory

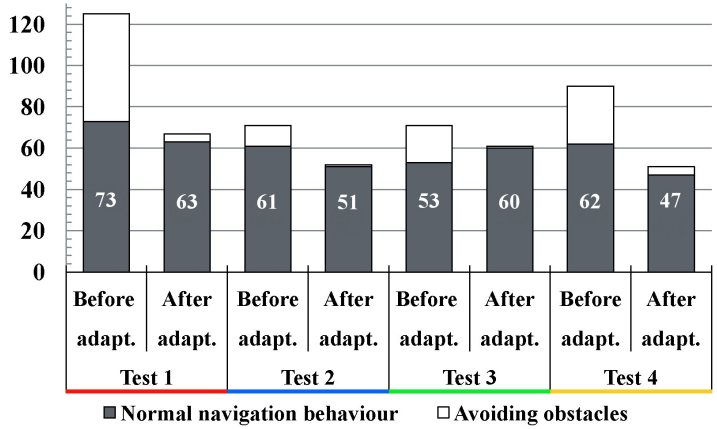


Figure 5.7: Results of the map adaptation

in Figure 5.5 shows (see the red curve), the robot walks around the table, and finally reaches the target person. After the first experiment, we repeat the task with the adapted spatial memory. As the new trajectory (in blue) in Figure 5.5 shows, the robot chooses directly the alternative path after starting to avoid obstacles pro-actively and approaches the person with a smoother curve. The difference in behavior shows that the map adaptation helps the robot to find a suitable trajectory to approach the target position successfully.

We evaluate the performance of the navigation from different positions, and a subset of the experiments is displayed in Figure 5.6. A map is built first using the person's localization for each test, and then we let the robot navigate from its position (shown as circles) to the person's position (shown as squares). During the robot navigation, obstacles are being

detected by sonar sensors that help the robot to keep a certain distance to the barriers. As the robot and the person may have a different motion behavior, the planned navigation paths are often blocked by corners of furniture that are easy for a person to avoid (see Test 1, 2 and 3). Moreover, new obstacles could be placed in the room that was not present during mapping (cf. Test 4 that a black box is placed between the start and the target position). During the robot's exploration, the robot should detect the new object and adapt the map by deactivating the connections close to them automatically.

The results of the navigation time analysis are shown in Figure 5.7. Comparing the navigation time before and after adaptation, the time consumed for avoiding obstacles decreases dramatically. During the first trail of navigation, the robot detects the obstacles and adapts the map by deactivating the connections close to them. After the adaptation, when starting the same navigation task, the robot could choose a safer route for approaching the target and less reactive behavior would be activated. In some cases (e.g., Test 3) the time for normal navigation behavior increases, because the robot may choose a longer path to avoid obstacles. However, regarding safety as well as general performance, the map adaptation helps to improve the navigation behavior significantly.

5.3 Discussion

In this chapter, we describe the developed method for adaptive learning of the navigation behavior by adjusting the connection weights of the cognitive map. Because the mobility of a person is different from a robot's mobility, the map learned

by observing the person's movement may not be suitable for robots. Therefore, we approach a reflex-like behavior to avoid obstacles in real time and adapt the environment based on the feedback from the robot sensors. This adaptation helps the robot to memorize the environment change and avoid the same obstacle pro-actively when the robot revisits the position. From the biological point of view, the learning through interaction resembles the principle of error-based learning of sensorimotor map (Wolpert et al., 2011). Because the connection weights can decay but also grow again, the regenerated path planning based on the updated connection weights allows the robot to learn the environment perceptually which is required in the dynamic environment.

A temporary connection weight is imported to coordinate the navigation behavior of multiple robots. A robot is aware of the position of other robots and manages to adjust its motion pro-actively during navigation. As here only the location of robots is needed, and the reconfiguration is integrated naturally in the path planning phase, the replanning is efficient and is capable of coordinating motions of robots in real time. We validate this model in a real environment and show that the combined reflex-based and path-based obstacle avoidance provides an efficient and robust way of navigation in a dynamic environment.

Chapter 6

Conclusion

In this thesis, we present a neurocognitive architecture for indoor localization and robot navigation using computational neuron models. We considered the requirements and challenges of developing socially assistive robots to serve users in their daily lives. Unlike the laboratory experiments that are conducted in a well-constructed environment, the real robotic application in daily life has to confront an unknown, cluttered, and dynamic environment. Due to the complexity of this situation, which is hard to predict perfectly, an intelligent learning ability is essential to allow the system to adapt its behavior to the new environment automatically. For this reason, our research focused on developing adaptive learning mechanisms for localization and navigation. We consider that humans can accomplish various cognitive tasks easily, for example, object recognition, environment perception, etc. The mechanism of information processing in the brain could be a potentially relevant inspiration to improve the performance and functionality

of the robot. Therefore, our research took inspiration from the neural models and developed neural models for person localization using multiple-cue integration and robot navigation based on sensorimotor map learning.

The localization model is based on a hybrid neural and probabilistic framework using multiple visual cues. While purely robotic models of navigation use sophisticated sensors such as laser scanners, our approach uses a simple ambient sensor, i.e., a ceiling-mounted camera, to localize the robot and the target person. The ceiling camera presents a simple, low-cost, and efficient solution to localize a person anywhere in a larger room. It enables the robot to locate the position of the target person as well as to know when (s)he is out of the detection range of the robot's sensors (for example when the robot cannot see the person from its camera). We employed a Sigma-Pi network to integrate the output of different cues together with their corresponding reliability factors, which adjusts the importance of different cues and improves the tracking robustness. The model is to some extent indicative of a human's ability to recognize objects based on different features. When some of the features are strongly disturbed, detection recovers by the integration of other features. The particle filter parallels an active attention selection mechanism that allocates most processing resources to the potential positions where the target may be. Therefore, the system has a high performance in detecting complex objects that move relatively slowly in real time. Advantages of this system are that the feature patterns used for one cue, such as the color histogram, can adapt online to provide a more robust identification of a person. With this short-term memory mechanism, the system masters the chal-

lenge of an unstructured environment and is able to detect moving objects in a home-like environment. The system is highly flexible for integrating different streams of information. Depending on the requirement, the system can be adapted easily by adding specific information channels. In principle, the more cues there are, the better tracking performance we could get. Overall, our model has potential as a robust method for object detection and tracking in complex conditions.

The presented neural framework for robot navigation is inspired by the sensorimotor map learning of animals and human beings. By combining path planning with the real-time handling of obstacles, the system realizes a human-like navigation behavior that builds up an abstract navigation strategy with spatial memory while avoiding obstacles in real time based on close spatial perception. The system consists of multiple layers of neural networks that combine map building and localization with planning and navigation. A distributed neural coding is used (i.e., multiple hypotheses) to represent the positions of the robot and the target. This helps the navigation system to handle sensor and actuator noise in the real environment. Besides bringing the model closer to the spiking and redundant population coding of real neurons, this is useful for representing the locations from which the robot can see an object, which may be observed from different positions. Based on these distributed representations in the spatial memory, multiple action memories associated with state transitions combine in the competitive action layer, which yields a robust and smooth control signal for navigation.

A reflex-like behavior is developed in our model to handle the dynamics of a real environment. When an obstacle is de-

tected by the robot's sensors, the reflex-like behavior guides the robot to avoid it and reduces the corresponding action memory weights at the same time. The robot can remember the obstacles in its spatial memory of the sensorimotor map using this method and avoid them pro-actively in the future. Weight reduction and recovery, together with dynamic space representations, enable life-long model adaptation. The path planning of multiple robots can be achieved efficiently in the system based on the individual robot location in the spatial memory, which supports the system to avoid collisions with other moving robots pro-actively. This method can be also extended to avoid moving persons when the location of them is determined. A further unique feature of our approach is that by anchoring the appearance features of the environment with the states in the spatial memory, visual associations are linked to specific locations on the map, which enables a robot to find a target object by comparing its visual memory with the features of the target object.

The map learning and adaptation is inspired by the principles of sensorimotor learning (Wolpert et al., 2011): (1) observational learning that develops the map and the corresponding motor skills by watching a moving person, and (2) error-based learning during navigation that adapts the map, and hence its navigation strategy, based on interaction with obstacles. On the other hand, the robot uses the internal state representation for path planning and navigation instead of reading the sensor input directly, which resembles the property of cognitive maps introduced by Tolman (1948) in which a rat uses some form of internal spatial representation for navigation rather than following cue stimuli directly. As a result, robots can

realize complex cognitive behavior using biologically inspired methods. Through embedding of the sensorimotor map into a system architecture that integrates planning and reflex-based behavior, the robot achieves highly flexible navigation behavior and perpetual learning of the spatial knowledge of the environment.

We have tested the localization as well as the navigation model in a home-like laboratory environment. Experimental results show that our method is capable of tracking the target person and the robot in an uncontrolled environment through combining different visual cues, and the robot is able to navigate to the target person, avoid obstacles, and adapt its trajectory autonomously in a cluttered environment. It is shown that cognitive tasks such as object finding and bringing back to the target person by a robot can be realized. Nevertheless, further developments are still required to extend functionalities of the system to help a robot to assist persons in the real life. Currently, the localization model is constrained to tracking a single person in the room because the system is initially designed to serve the target person when he or she is alone at home. However, multiple persons may need to be detected in a real-life scenario, for example for robot navigation and avoiding moving persons. Therefore, further development of this model for multiple person tracking is important. Through the experiments, our system has been shown to track a specific person while other persons are in the room, which means that our hybrid system can distinguish different persons and has the potential to achieve multiple person tracking as well. Necessary improvement will be considered for future work. For example, the particle filter framework will be adapted for mul-

tiple person tracking, for example using the RJMCMC algorithm (Green, 1995).

Our navigation architecture learns a sensorimotor map by observing the movement of a person in a room in order to accelerate the mapping phase. However, it might not be suitable for a person to explore some places for safety reasons (e.g., a room with a high temperature). Accelerating the room mapping without support from a person is essential in this case. In future work, we plan to focus therefore on learning a sensorimotor map via observing the active exploration movement of a robot. The ceiling-mounted camera is an efficient external sensor that needs to be installed, but it also constrains the flexibility of the system. We will use biologically plausible methods (e.g., stereo vision) to solve this problem by modeling how information is processed in the brain. Moreover, as we have shown in the previous chapter, our system has the potential to control the trajectory planning of multiple robots based on the sensorimotor map. Therefore, it would be interesting to extend the system from a single robot navigation to the coordination of multiple robots.

In conclusion, our neural-based system provides a comprehensive method for indoor robot localization and navigation. The key achievement of this work is that intelligent functions such as indoor robot navigation, visual appearance anchoring, and adaptive learning of environmental changes are developed successfully based on the study of the brain's structure. Compared to traditional robotic approaches that rely on accurate measurements for trajectory planning, a neurally inspired method provides self-learning abilities with a redundant information representation. This results in robust, efficient navi-

gation ability and is capable of achieving cognitive functions such as environment understanding. Experimental findings in a home-like environment show that neural networks are powerful methods for helping an assistive robot accomplish complex cognitive tasks such as finding an object and approaching it automatically.

Publications

Publications which have been done as part of this research are listed below:

Yan W., Weber C., and Wermter S. A hybrid probabilistic neural model for person tracking based on a ceiling-mounted camera. *Journal of Ambient Intelligence and Smart Environments*, 3:237–252, 2011. ISSN 1876-1364.

Yan W., Torta E., van der Pol D., Meins N., Weber C., Cuijpers R. H., and Wermter S. *Robotic Vision: Technologies for Machine Learning and Vision Applications*, chapter Learning Robot Vision for Assisted Living, pages 257–280. IGI Global, 2012a. doi: 10.4018/978-1-4666-2672-0.ch015.

Yan W., Weber C., and Wermter S. Learning indoor robot navigation using visual and sensorimotor map information. *Frontiers in Neurorobotics*, 7(15), 2013. ISSN 1662-5218. doi: 10.3389/fnbot.2013.00015.

Johnson D. O., Cuijpers R. H., Juola J. F., Torta E., Simonov M., Frisiello A., Bazzani M., Yan W., Weber C., Wermter S., Meins N., Oberzaucher J., Panek P., Edel-

mayer G., Mayer P., and Beck C. Socially Assistive Robots: A Comprehensive Approach to Extending Independent Living. *International Journal of Social Robotics*, November, pages 1–17, 2013.

Yan W., Weber C., and Wermter S. Person tracking based on a hybrid neural probabilistic model. In Honkela T., Duch W., Girolami M., and Kaski S., editors, *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN 2011)*, volume 2, pages 365–372, Espoo, Finland, June 2011. Springer.

Yan W., Weber C., and Wermter S. A neural approach for robot navigation based on cognitive map learning. In *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN2012)*, pages 1146–1153, Brisbane, Australian, June 2012b. IEEE.

Yan W. Robot navigation for assisted living. *Gerontechnology*, 11(2):356, 2012.

Bibliography

Abramowitz, M. and Stegun, I. A., editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, chapter 9.6 Modified Bessel Functions I and K, pages 374–377. Dover Publications, 1965.

Adams, M., Zhang, S., and Xie, L. Particle filter based outdoor robot localization using natural features extracted from laser scanners. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1493–1498, 2004. doi: 10.1109/ROBOT.2004.1308035.

Amari, S. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–87, 1977. ISSN 0340–1200. doi: 10.1007/BF00337259.

Andersen, P., Morris, R., Amaral, D., Bliss, T., and O'Keefe, J. *The hippocampus book*. Oxford University Press, USA, 2006.

Arleo, A., del R.Millan, J., and Floreano, D. Efficient learning of variable-resolution cognitive maps for autonomous indoor navigation. *IEEE Transactions on Robotics and Au-*

tomation, 15(6):990–1000, 1999. ISSN 1042-296X. doi: 10.1109/70.817664.

Aurenhammer, F. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991. ISSN 0360-0300. doi: 10.1145/116873.116880.

Bahadori, S., Iocchi, L., Leone, G., Nardi, D., and Scozzafava, L. Real-time people localization and tracking through fixed stereo vision. *Applied Intelligence*, 26:83–97, 2007. ISSN 0924-669X. doi: 10.1007/s10489-006-0013-3.

Bahl, P. and Padmanabhan, V. N. RADAR: An in-building rf-based user location and tracking system. In *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 775–784. IEEE, 2000. doi: 10.1109/INFCOM.2000.832252.

Bailey, T., Nieto, J., Guivant, J., Stevens, M., and Nebot, E. Consistency of the EKF-SLAM Algorithm. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, Beijing, October 2006. IEEE. doi: 10.1109/IROS.2006.281644.

Balleine, B. W., Delgado, M. R., and Hikosaka, O. The role of the dorsal striatum in reward and decision-making. *The Journal of Neuroscience*, 27(31):8161–8165, 2007. doi: 10.1523/JNEUROSCI.1554-07.2007.

Barger, T., Brown, D., and Alwan, M. Health-status monitoring through analysis of behavioral patterns. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems*

and Humans, 35(1):22–27, January 2005. ISSN 1083-4427. doi: 10.1109/TSMCA.2004.838474.

Barr, A., Feigenbaum, E. A., and Cohen, P. R. *The handbook of artificial intelligence*, volume 1. Pitman, 1981.

Bauer, C., Milighetti, G., Yan, W., and Mikut, R. Human-like reflexes for robotic manipulation using leaky integrate-and-fire neurons. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2572–2577. IEEE, October 2010. doi: 10.1109/IROS.2010.5648900.

Bauer, G. and Lukowicz, P. Developing a sub room level indoor location system for wide scale deployment in assisted living systems. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 5105 of *Lecture Notes in Computer Science*, pages 1057–1064. Springer Berlin / Heidelberg, 2008. doi: 10.1007/978-3-540-70540-6\158.

Bay, H., Tuytelaars, T., and Van Gool, L. SURF: Speeded up robust features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision - ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006. doi: 10.1007/11744023\32.

Benezeth, Y., Jodoin, P., Emile, B., Laurent, H., and Rosenberger, C. Review and evaluation of commonly-implemented background subtraction algorithms. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, December 2008. doi: 10.1109/ICPR.2008.4760998.

Bennewitz, M. *Mobile Robot Navigation in Dynamic Environments*. PhD thesis, University of Freiburg, 2004.

Bernardin, K. and Stiefelhagen, R. Evaluating multiple object tracking performance: the CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008(1): 246309, 2008. ISSN 1687-5281. doi: 10.1155/2008/246309.

Bernardin, K., Gehrig, T., and Stiefelhagen, R. Multi-level particle filter fusion of features and cues for audio-visual person tracking. In Stiefelhagen, R., Bowers, R., and Fiscus, J., editors, *Multimodal Technologies for Perception of Humans*, volume 4625 of *Lecture Notes in Computer Science*, pages 70–81. Springer Berlin / Heidelberg, 2008. doi: 10.1007/978-3-540-68585-2_5.

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, 1995.

Blair, H. and Sharp, P. Anticipatory head direction signals in anterior thalamus: evidence for a thalamocortical circuit that integrates angular head motion to compute head direction. *The Journal of Neuroscience*, 15(9):6260–6270, 1995. URL `http://www.jneurosci.org/content/15/9/6260.abstract`.

Blair, H. T., Cho, J., and Sharp, P. E. Role of the lateral mammillary nucleus in the rat head direction circuit: A combined single unit recording and lesion study. *Neuron*, 21(6):1387–1397, 1998. ISSN 0896-6273. doi: 10.1016/S0896-6273(00)80657-1.

Borenstein, J. and Koren, Y. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991. ISSN 1042-296X. doi: 10.1109/70.88137.

Bradski, G. R. Conputer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1998.

Brooks, R. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986. ISSN 0882-4967. doi: 10.1109/JRA.1986.1087032.

Broxvall, M., Gritti, M., Saffiotti, A., Seo, B., and Cho, Y. PEIS Ecology: integrating robots into smart environments. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 212 –218. IEEE, May 2006. doi: 10.1109/ROBOT.2006.1641186.

Burgess, N., Maguire, E. A., and O'Keefe, J. The Human Hippocampus and Spatial and Episodic Memory. *Neuron*, 35(4):625–641, 2002a. ISSN 0896-6273. doi: 10.1016/S0896-6273(02)00830-9.

Burgess, N., Maguire, E. A., and O'Keefe, J. The human hippocampus and spatial and episodic memory. *Neuron*, 35 (4):625–641, 2002b. doi: 10.1016/S0896-6273(02)00830-9.

Canton-Ferrer, C., Salvador, J., Casas, J., and Pards, M. Multi-person tracking strategies based on voxel analysis. In Stiefelhagen, R., Bowers, R., and Fiscus, J., editors, *Multimodal Technologies for Perception of Humans*, volume 4625 of *Lecture Notes in Computer Science*, pages 91–103.

Springer Berlin Heidelberg, 2008. ISBN 978-3-540-68584-5. doi: 10.1007/978-3-540-68585-2_7.

Chen, L. L., Lin, L.-H., Green, E. J., Barnes, C. A., and McNaughton, B. L. Head-direction cells in the rat posterior cortex. *Experimental Brain Research*, 101:8–23, 1994. ISSN 0014-4819. doi: 10.1007/BF00243212.

Chen, Q., Wu, H., and Yachida, M. Face detection by fuzzy pattern matching. In *Proceedings of the 5th International Conference on Computer Vision*, pages 591–596, June 1995. doi: 10.1109/ICCV.1995.466885.

Cho, J. and Sharp, P. Head direction, place, and movement correlates for cells in the rat retrosplenial cortex. *Behavioral Neuroscience*, 115(1)(1):3–25, 2001. doi: 10.1037/0735-7044.115.1.3.

Collett, M., Collett, T. S., Bisch, S., and Wehner, R. Local and global vectors in desert ant navigation. *Nature*, 394 (6690):269–272, 1998. doi: 10.1038/28378.

Comaniciu, D., Ramesh, V., and Meer, P. Real-time tracking of non-rigid objects using mean shift. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2142–2150, Los Alamitos, CA, USA, 2000. IEEE Computer Society. doi: 10.1109/CVPR.2000.854761.

Connell, J. H. *Minimalist mobile robotics: a colony-style architecture for an artificial creature*. Academic Press Professional, Inc., San Diego, CA, USA, 1990. ISBN 0-12-185230-X.

Cox, I. J. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991. ISSN 1042-296X. doi: 10.1109/70.75902.

Cuperlier, N., Quoy, M., Laroque, P., and Gaussier, P. Transition cells and neural fields for navigation and planning. In Mira, J. and Álvarez, J., editors, *Mechanisms, Symbols, and Models Underlying Cognition*, volume 3561 of *Lecture Notes in Computer Science*, pages 147–152. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-26298-5. doi: 10.1007/11499220_36.

Cuperlier, N., Quoy, M., and Gaussier, P. Neurobiologically inspired mobile robot navigation and planning. *Frontiers in Neurorobotics*, 1(0), 2007. ISSN 1662-5218. doi: 10.3389/neuro.12.003.2007.

Demiroz, B., Ari, I., Eroglu, O., Salah, A., and Akarun, L. Feature-based tracking on a multi-omnidirectional camera dataset. In *Proceedings of the 5th International Symposium on Communications Control and Signal Processing (ISCCSP)*, pages 1–5, May 2012. doi: 10.1109/ISCCSP.2012.6217867.

Deneve, S. Bayesian inference in spiking neurons. *Advances in neural information processing systems*, 17:353–360, 2005.

Diosi, A., Taylor, G., and Kleeman, L. Interactive SLAM using Laser and Advanced Sonar. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1103–1108. IEEE, 2005. doi: 10.1109/ROBOT.2005.1570263.

Dong, G. and Xie, M. Color clustering and learning for image segmentation based on neural networks. *IEEE Transactions on Neural Networks*, 16(4):925–936, July 2005. ISSN 1045-9227. doi: 10.1109/TNN.2005.849822.

Donnart, J.-Y. and Meyer, J.-A. Learning reactive and planning rules in a motivationally autonomous animat. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(3):381–395, 1996. ISSN 1083-4419. doi: 10.1109/3477.499790.

Doucet, A., De Freitas, N., Murphy, K., and Russell, S. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the 16th conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000. URL `http://dl.acm.org/citation.cfm?id=2073946.2073968`.

Drumheller, M. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):325–332, 1987. ISSN 0162-8828. doi: 10.1109/TPAMI.1987.4767907.

Duckett, T. and Nehmzow, U. Exploration of unknown environments using a compass, topological map and neural network. In *Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 312–317. IEEE, 1999. doi: 10.1109/CIRA.1999.810067.

Durrant-Whyte, H. and Bailey, T. Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*,

13(2):99–110, 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1638022.

Elgammal, A., Harwood, D., and Davis, L. Non-parametric model for background subtraction. In Vernon, D., editor, *Computer Vision ECCV 2000*, volume 1843 of *Lecture Notes in Computer Science*, pages 751–767. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67686-7. doi: 10.1007/3-540-45053-X_48.

Eriksson, P. S., Perfilieva, E., Björk-Eriksson, T., Alborn, A.-M., Nordborg, C., Peterson, D. A., and Gage, F. H. Neurogenesis in the adult human hippocampus. *Nature medicine*, 4(11):1313–1317, 1998. doi: 10.1038/3305.

Erlhagen, W. and Bicho, E. The dynamic neural field approach to cognitive robotics. *Journal of Neural Engineering*, 3(3): R36, 2006. URL `http://stacks.iop.org/1741-2552/3/i=3/a=R02`.

Estrada, C., Neira, J., and Tardos, J. Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005. ISSN 1552-3098. doi: 10.1109/TRO.2005.844673.

Fleuret, F., Berclaz, J., Lengagne, R., and Fua, P. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1174.

Fox, D., Thrun, S., Burgard, W., and Dellaert, F. Particle filters for mobile robot localization. In Doucet, A., de Freitas,

N., and Gordon, N., editors, *Sequential Monte Carlo Methods in Practice*, pages 499–516. Springer New York, 2001. ISBN 978-1-4419-2887-0. doi: 10.1007/978-1-4757-3437-9_19.

Frintrop, S., Königs, A., Hoeller, F., and Schulz, D. A component-based approach to visual person tracking from a mobile platform. *International Journal of Social Robotics*, 2:53–62, 2010. ISSN 1875-4791. doi: 10.1007/s12369-009-0035-1.

Fritzke, B. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, volume 7, pages 625–632. MIT Press, 1995.

Gaspar, J., Winters, N., and Santos-Victor, J. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000. ISSN 1042-296X. doi: 10.1109/70.897802.

Gavrilov, A. and Lee, S. Usage of hybrid neural network model MLP-ART for navigation of mobile robot. In Huang, D.-S., Heutte, L., and Loog, M., editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 4682 of *Lecture Notes in Computer Science*, pages 182–191. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74201-2. doi: 10.1007/978-3-540-74205-0_21.

Giese, M. A. *Dynamic Neural Field Theory for Motion Perception*, volume 469 of *The Springer International Series*

in Engineering and Computer Science. Springer US, 1999. ISBN 0792383001. doi: 10.1007/978-1-4615-5581-0.

Gorchetchnikov, A. and Grossberg, S. Space, time and learning in the hippocampus: How fine spatial and temporal scales are expanded into population codes for behavioral control. *Neural Networks*, 20(2):182–193, 2007. ISSN 0893-6080. doi: 10.1016/j.neunet.2006.11.007.

Gordon, N., Salmond, D., and Smith, A. F. M. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113, April 1993. ISSN 0956-375X. doi: 10.1049/ip-f-2.1993.0015.

Green, P. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82 (4):711, 1995. ISSN 0006-3444. doi: 10.1093/biomet/82.4.711.

Guivant, J. E., Masson, F. R., and Nebot, E. M. Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, 40(2–3):79–90, 2002. ISSN 0921-8890. doi: 10.1016/S0921-8890(02)00233-6.

Guomundsson, S., Larsen, R., Aanaes, H., Pardas, M., and Casas, J. TOF imaging in Smart room environments towards improved people tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, 2008. doi: 10.1109/CVPRW.2008.4563154.

Hafting, T., Fyhn, M., Molden, S., Moser, M., and Moser, E. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005. doi: 10.1038/nature03721.

Hahnel, D., Burgard, W., Fox, D., and Thrun, S. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 206–211 vol.1, 2003. doi: 10.1109/IROS.2003.1250629.

Harris, C. and Stephens, M. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50, Manchester, UK, 1988.

Hartland, C. and Bredeche, N. Using echo state networks for robot navigation behavior acquisition. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 201–206, Sanya, December 2007. IEEE. doi: 10.1109/ROBIO.2007.4522160.

Hecht, F., Azad, P., and Dillmann, R. Markerless human motion tracking with a flexible model and appearance learning. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 3173–3179. IEEE, May 2009. doi: 10.1109/ROBOT.2009.5152494.

Herbort, O., Butz, M., and Pedersen, G. The SURE_REACH model for motor learning and control of a redundant arm: From modeling human behavior to applications in robotics. In Sigaud, O. and Peters, J., editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in*

Computational Intelligence, pages 85–106. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-05180-7. doi: 10.1007/978-3-642-05181-4_5.

Hootman, J. and Helmick, C. Projections of US prevalence of arthritis and associated activity limitations. *Arthritis & Rheumatism*, 54(1):226–229, 2006. ISSN 1529-0131. doi: 10.1002/art.21562.

Hsu, R.-L., Abdel-Mottaleb, M., and Jain, A. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):696–706, May 2002. ISSN 0162-8828. doi: 10.1109/34.1000242.

Hu, M.-K. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, February 1962. ISSN 0096-1000. doi: 10.1109/TIT.1962.1057692.

Huang, Y. and Rao, R. Neurons as Monte Carlo Samplers: Sequential Bayesian inference in spiking neural populations. *Frontiers in Systems Neuroscienc, Conference Abstract: Computational and systems neuroscience*, 2009. doi: 10.3389/conf.neuro.06.2009.03.048.

Hyvärinen, A. and Oja, E. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000. ISSN 0893-6080. doi: DOI:10.1016/S0893-6080(00)00026-5.

Isard, M. and MacCormick, J. BraMBLe: A Bayesian multiple-blob tracker. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 34–

41, Vancouver, BC, July 2001. IEEE. doi: 10.1109/ICCV.2001.937594.

Itti, L., Koch, C., and Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, November 1998. ISSN 0162-8828. doi: 10.1109/34.730558.

Johnson, D., Cuijpers, R., Juola, J., Torta, E., Simonov, M., Frisiello, A., Bazzani, M., Yan, W., Weber, C., Wermter, S., Meins, N., Oberzaucher, J., Panek, P., Edelmayer, G., Mayer, P., and Beck, C. Socially assistive robots: A comprehensive approach to extending independent living. *International Journal of Social Robotics*, 6(2):195–211, 2014. ISSN 1875-4791. doi: 10.1007/s12369-013-0217-8.

Jolliffe, I. *Principal Component Analysis*. John Wiley & Sons, Ltd, 2005. ISBN 9780470013199. doi: 10.1002/0470013192.bsa501.

Jordao, L., Perrone, M., Costeira, J., and Santos-Victor, J. Active face and feature tracking. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 572–576, 1999. doi: 10.1109/ICIAP.1999.797657.

Kaelbling, L., Littman, M., and Cassandra, A. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(98)00023-X.

Kakumanu, P., Makrogiannis, S., and Bourbakis, N. A survey of skin-color modeling and detection methods. *Pattern*

Recognition, 40(3):1106–1122, 2007. ISSN 0031-3203. doi: 10.1016/j.patcog.2006.06.010.

Kalal, Z., Mikolajczyk, K., and Matas, J. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.239.

Karlsson, N., Di Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P., and Munich, M. The vSLAM Algorithm for Robust Localization and Mapping. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 24–29. IEEE, 2005. doi: 10.1109/ROBOT.2005.1570091.

Kemmotsu, K., Koketsua, Y., and Iehara, M. Human behavior recognition using unconscious cameras and a visible robot in a network robot system. *Robotics and Autonomous Systems*, 56(10):857–864, 2008. doi: DOI:10.1016/j.robot.2008.06.004.

Kemper, J. and Linde, H. Challenges of passive infrared indoor localization. In *Proceedings of the 5th Workshop on Positioning, Navigation and Communication*, pages 63–70, March 2008. doi: 10.1109/WPNC.2008.4510358.

Khan, S. M. and Shah, M. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:505–519, 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.102.

Kleeman, L. and Kuc, R. Mobile robot sonar for target localization and classification. *The International Journal*

of Robotics Research, 14(4):295–318, 1995. doi: 10.1177/027836499501400401.

Klein, D. A., Schulz, D., Frintrop, S., and Cremers, A. B. Adaptive real-time video-tracking for arbitrary objects. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 772–777. IEEE, 2010. doi: 10.1109/IROS.2010.5650583.

Knoop, S., Vacek, S., and Dillmann, R. Sensor fusion for 3D human body tracking with an articulated 3D body model. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 1686–1691. IEEE, 2006. doi: 10.1109/ROBOT.2006.1641949.

Knudson, M. and Tumer, K. Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems*, 59 (6):410–420, 2011. doi: 10.1016/j.robot.2011.02.004.

Kobilarov, M., Sukhatme, G., Hyams, J., and Batavia, P. People tracking and following with mobile robot using an omnidirectional camera and a laser. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pages 557–562. IEEE, May 2006. doi: 10.1109/ROBOT.2006.1641769.

Koch, J., Wettach, J., Bloch, E., and Berns, K. Indoor localisation of humans, objects, and mobile robots with RFID infrastructure. In *Proceedings of the 7th International Conference on Hybrid Intelligent Systems*, volume 0, pages 271–276, Los Alamitos, CA, USA, 2007. IEEE Computer Society. ISBN 0-7695-2946-1. doi: 10.1109/HIS.2007.25.

Kovac, J., Peer, P., and Solina, F. Human skin color clustering for face detection. In *Proceedings of the IEEE Internetional Conference on Computer as a Tool, EUROCON*, volume 2, pages 144–148, September 2003. doi: 10.1109/EURCON.2003.1248169.

Krose, B. and Eecen, M. A self-organizing representation of sensor space for mobile robot navigation. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 9–14 vol.1, 1994. doi: 10.1109/IROS.1994.407415.

KSERA. The KSERA project (Knowledgeable SErvice Robots for Aging). http://ksera.ieis.tue.nl/.

Kuipers, B. and Beeson, P. Bootstrap learning for place recognition. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 174–180. MIT Press, 2002.

Lanz, O. and Brunelli, R. An appearance-based particle filter for visual tracking in smart rooms. In Stiefelhagen, R., Bowers, R., and Fiscus, J., editors, *Multimodal Technologies for Perception of Humans*, volume 4625 of *Lecture Notes in Computer Science*, pages 57–69. Springer Berlin / Heidelberg, 2008. doi: 10.1007/978-3-540-68585-2_4.

Latombe, J.-C. *Robot Motion Planning*, volume 124 of *The Springer International Series in Engineering and Computer Science*. Springer US, 1990. doi: 10.1007/978-1-4615-4022-9.

Lavoie, A. and Mizumori, S. Spatial, movement- and reward-sensitive discharge by medial ventral striatum neurons of

rats. *Brain Research*, 638(12):157–168, 1994. ISSN 0006-8993. doi: 10.1016/0006-8993(94)90645-9.

Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I., and Choset, H. Hierarchical simultaneous localization and mapping. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 448–453. IEEE, 2003. doi: 10.1109/IROS.2003.1250670.

Louloudi, A., Mosallam, A., Marturi, N., Janse, P., and Hernandez, V. Integration of the Humanoid Robot Nao inside a Smart Home: A Case Study. In *The Swedish AI Society Workshop*. Uppsala University, May 2010. URL `http://www.ep.liu.se/ecp/048/008/`.

Lowe, D. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999. doi: 10.1109/ICCV.1999.790410.

Lowe, D. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94.

Lozano-Pérez, T. and Wesley, M. A. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979. ISSN 0001-0782. doi: 10.1145/359156.359164.

Luber, M., Spinello, L., and Arras, K. O. People tracking in RGB-D data with on-line boosted target models. In *Pro-*

ceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3844–3849, September 2011. doi: 10.1109/IROS.2011.6095075.

Maguire, E. A., Burgess, N., Donnett, J. G., Frackowiak, R. S., Frith, C. D., and O'Keefe, J. Knowing where and getting there: a human navigation network. *Science*, 280(5365): 921–924, 1998. doi: 10.1126/science.280.5365.921.

Mao, G., Fidan, B., and Anderson, B. D. Wireless sensor network localization techniques. *Computer Networks*, 51(10): 2529–2553, 2007. ISSN 1389-1286. doi: 10.1016/j.comnet.2006.11.018.

Marsland, S., Shapiro, J., and Nehmzow, U. A self-organising network that grows when required. *Neural Networks*, 15(8–9):1041–1058, October 2002. ISSN 0893-6080. doi: 10.1016/S0893-6080(02)00078-3.

Martinet, L.-E., Sheynikhovich, D., Benchenane, K., and Arleo, A. Spatial learning and action planning in a prefrontal cortical network model. *PLoS Computational Biology*, 7(5): e1002045, 2011. doi: 10.1371/journal.pcbi.1002045.

Matsumoto, Y., Inaba, M., and Inoue, H. Visual navigation using view-sequenced route representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 83–88. IEEE, 1996. doi: 10.1109/ROBOT.1996.503577.

Mckenna, S. J., Gong, S., and Raja, Y. Modelling facial colour and identity with gaussian mixtures. *Pattern Recognition*,

31(12):1883–1892, 1998. ISSN 0031-3203. doi: 10.1016/S0031-3203(98)00066-1.

Meyer, J.-A. and Filliat, D. Map-based navigation in mobile robots:: II. A review of map-learning and path-planning strategies. *Journal of Cognitive Systems Research*, 4(4):283–317, 2003. ISSN 1389-0417. doi: 10.1016/S1389-0417(03)00007-X.

Milford, M. and Wyeth, G. Persistent navigation and mapping using a biologically inspired SLAM system. *The International Journal of Robotics Research*, 29(9):1131–1153, 2010. doi: 10.1177/0278364909340592.

Milford, M., Wyeth, G., and Prasser, D. RatSLAM: a hippocampal model for simultaneous localization and mapping. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 1, pages 403–408. IEEE, 2004. doi: 10.1109/ROBOT.2004.1307183.

Mizumori, S. and Williams, J. Directionally selective mnemonic properties of neurons in the lateral dorsal nucleus of the thalamus of rats. *The Journal of Neuroscience*, 13(9):4015–4028, 1993. URL `http://www.jneurosci.org/content/13/9/4015.abstract`.

Montello, D. Spatial cognition. *International Encyclopedia of the Social & Behavioral Sciences*, 7:14771–14775, 2001.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the 18th National*

conference on Artificial Intelligence, pages 593–598. AAAI, 2002.

Muñoz-Salinas, R., Aguirre, E., and García-Silvente, M. People detection and tracking using stereo vision and color. *Image and Vision Computing*, 25(6):995–1007, 2007. ISSN 0262-8856. doi: 10.1016/j.imavis.2006.07.012.

Murphy, R. Biological and cognitive foundations of intelligent sensor fusion. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 26(1):42–51, 1996. ISSN 1083–4427. doi: 10.1109/3468.477859.

Murphy, R. *Introduction to AI robotics*. The MIT Press, 2000.

Nait-Charif, H. and McKenna, S. Activity summarisation and fall detection in a supportive home environment. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pages 323–326, August 2004. doi: 10.1109/ICPR.2004.1333768.

Newman, P. and Ho, K. SLAM-loop closing with visually salient features. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 635–642. IEEE, 2005. doi: 10.1109/ROBOT.2005.1570189.

Nickel, K., Gehrig, T., Stiefelhagen, R., and McDonough, J. A joint particle filter for audio-visual speaker tracking. In *Proceedings of the 7th international conference on Multimodal interfaces*, ICMI '05, pages 61–68, New York, NY, USA, 2005. ACM. ISBN 1-59593-028-0. doi: 10.1145/1088463.1088477.

Novak, C. and Shafer, S. Anatomy of a color histogram. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 599–605. IEEE, June 1992. doi: 10.1109/CVPR.1992.223129.

Nummiaro, K., Koller-Meier, E., and Van Gool, L. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, 2003. doi: 10.1016/S0262-8856(02)00129-4.

O'Grady, M., Muldoon, C., Dragone, M., Tynan, R., and O'Hare, G. Towards evolutionary ambient assisted living systems. *Journal of Ambient Intelligence and Humanized Computing*, 1:15–29, 2010. ISSN 1868–5137. doi: 10.1007/s12652-009-0003-5. URL `http://dx.doi.org/10.1007/s12652-009-0003-5`.

O'Keefe, J. and Dostrovsky, J. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, 1971. ISSN 0006–8993. doi: 10.1016/0006-8993(71)90358-1.

Oriolo, G., Ulivi, G., and Vendittelli, M. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,*, 28(3):316–333, 1998. ISSN 1083-4419. doi: 10.1109/3477.678626.

Parisi, G. and Wermter, S. Hierarchical SOM-based detection of novel behavior for 3D human tracking. In *Proceedings of the 2013 International Joint Conference on Neural Networks*, pages 1380–1387, Dallas, US, August 2013. IEEE. doi: 10.1109/IJCNN.2013.6706727.

Parisi, G. I., Strahl, E., and Wermter, S. Robust fall detection with an assistive humanoid robot. 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids), November 2014. Video Session.

Pennartz, C., Ito, R., Verschure, P., Battaglia, F., and Robbins, T. The hippocampal-striatal axis in learning, prediction and goal-directed behavior. *Trends in Neurosciences*, 34(10):548–559, 2011. ISSN 0166-2236. doi: 10.1016/j.tins.2011.08.001.

Phung, S. L., Bouzerdoum, A., and Chai, D. A novel skin color model in ycbcr color space and its application to human face detection. In *Proceedings of the 2002 International Conference on Image Processing*, volume 1, pages 289–292, 2002. doi: 10.1109/ICIP.2002.1038016.

Piccardi, M. Background subtraction techniques: a review. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3099–3104. IEEE, October 2004. doi: 10.1109/ICSMC.2004.1400815.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88.

Qian, Y., Medioni, G., and Cohen, I. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *Proceedings on the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2007. doi: 10.1109/CVPR.2007.382991.

Ramanan, D., Forsyth, D., and Zisserman, A. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:65–81, 2007. ISSN 0162–8828. doi: 10.1109/TPAMI.2007.22.

Raoui, Y., Goller, M., Devy, M., Kerscher, T., Zollner, J., Dillmann, R., and Coustou, A. RFID-based topological and metrical self-localization in a structured environment. In *Proceedings on the 2009 International Conference on Advanced Robotics*, pages 1–6, June 2009.

Reinstein, M. and Hoffmann, M. Dead reckoning in a dynamic quadruped robot based on multimodal proprioceptive sensory information. *IEEE Transactions on Robotics*, 29(2): 563–571, April 2013. ISSN 1552-3098. doi: 10.1109/TRO.2012.2228309.

Rowley, H., Baluja, S., and Kanade, T. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, Janurary 1998. ISSN 0162-8828. doi: 10.1109/34.655647.

Salah, A., Morros, R., Luque, J., Segura, C., Hernando, J., Ambekar, O., Schouten, B., and Pauwels, E. Multimodal identification and localization of users in a smart environment. *Journal on Multimodal User Interfaces*, 2:75–91, 2008. ISSN 1783 - 7677. doi: 10.1007/s12193-008-0008-y.

Samsonovich, A. and McNaughton, B. L. Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of Neuroscience*, 17(15):5900–5920, 1997. URL `http://www.jneurosci.org/content/17/15/5900.abstract`.

Samsonovich, A. V. and Ascoli, G. A. A simple neural network model of the hippocampus suggesting its pathfinding role in episodic memory retrieval. *Learning & Memory*, 12(2):193–208, 2005. doi: 10.1101/lm.85205.

Seraji, H. and Howard, A. Behavior-based robot navigation on challenging terrain: A fuzzy logic approach. *IEEE Transactions on Robotics and Automation*, 18(3):308–321, 2002. ISSN 1042-296X. doi: 10.1109/TRA.2002.1019461.

Smith, K., Gatica-Perez, D., and Odobez, J. Using Particles to Track Varying Numbers of Interacting People. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 962–969, Los Alamitos, CA, USA, 2005. IEEE Computer Society. doi: 10.1109/CVPR.2005.361.

Smith, T. and Simmons, R. G. Point-based POMDP algorithms: Improved analysis and implementation. *Computing Research Repository*, abs/1207.1412, 2012. URL `http://arxiv.org/abs/1207.1412`.

Stauffer, C. and Grimson, W. Adaptive background mixture models for real-time tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 246–252, 1999. doi: 10.1109/CVPR.1999.784637.

Steg, H., Strese, H., Loroff, C., Hull, J., and Schmidt, S. Europe is facing a demographic challenge Ambient Assisted Living offers solutions. IST Project Report on Ambient Assisted Living, March 2006.

Stockman, G. and Shapiro, L. G. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001. ISBN 0130307963.

Swain, M. and Ballard, D. Color indexing. *International Journal of Computer Vision*, 7:11–32, 1991. ISSN 0920-5691. doi: 10.1007/BF00130487.

Taube, J. Head direction cells recorded in the anterior thalamic nuclei of freely moving rats. *The Journal of Neuroscience*, 15(1):70–86, 1995. URL `http://www.jneurosci.org/content/15/1/70.abstract`.

Taube, J., Muller, R., and Ranck, J. Head-direction cells recorded from the postsubiculum in freely moving rats. ii. effects of environmental manipulations. *The Journal of Neuroscience*, 10(2):436–447, 1990. URL `http://www.jneurosci.org/content/10/2/436.abstract`.

Taube, J. S. and Muller, R. U. Comparisons of head direction cell activity in the postsubiculum and anterior thalamus of freely moving rats. *Hippocampus*, 8(2):87–108, 1998. ISSN 1098-1063. doi: 10.1002/(SICI)1098-1063(1998)8:2⟨87::AID-HIPO1⟩3.0.CO;2-4.

Terrillon, J.-C., David, M., and Akamatsu, S. Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition.*, pages 112–117, April. 1998. doi: 10.1109/AFGR.1998.670934.

Thorpe, W. H. *Learning and instinct in animals.* Harvard University Press, 1956.

Thrun, S. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00078-7.

Thrun, S. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, 2003. ISSN 0929-5593. doi: 10.1023/A:1025584807625.

Tolman, E. Cognitive maps in rats and men. *Psychological Review*, 55(4):189, 1948. doi: 10.1037/h0061626.

Torta, E., Cuijpers, R. H., and Juola, J. F. A model of the user's proximity for Bayesian inference. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 273–274, New York, NY, USA, 2011a. ACM. ISBN 978-1-4503-0561-7. doi: 10.1145/1957656.1957767.

Torta, E., Oberzaucher, J., Werner, F., Cuijpers, R. H., and Joula, J. F. Attitudes towards socially assistive robots in intelligent homes: Results from laboratory studies and field trials. *Journal of Human-Robot Interaction*, 1:1–16, 2011b. doi: 10.5898/JHRI.1.2.Torta.

Toussaint, M. A sensorimotor map: Modulating lateral interactions for anticipation and planning. *Neural Computation*, 18:1132–1155, 2006. ISSN 0899-7667. doi: 10.1162/089976606776240995.

Triesch, J. and Malsburg, C. Democratic integration: Self-organized integration of adaptive cues. *Neural Computation*, 13(9):2049–2074, 2001. doi: 10.1162/089976601750399308.

Van Der Meer, M. A. A. and Redish, A. D. Expectancies in decision making, reinforcement learning, and ventral striatum. *Frontiers in Neuroscience*, 3(6), 2010. ISSN 1662-453X. doi: 10.3389/neuro.01.006.2010.

Varshavsky, A., Chen, M., de Lara, E., Froehlich, J., Haehnel, D., Hightower, J., LaMarca, A., Potter, F., Sohn, T., Tang, K., and Smith, I. Are gsm phones the solution for localization? In *Proceedings of the 7th IEEE Workshop on Mobile Computing Systems and Applications*, pages 34–42, 2006. doi: 10.1109/WMCSA.2006.2.

Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518, 2001. doi: 10.1109/CVPR.2001.990517.

Want, R., Hopper, A., Falcão, V., and Gibbons, J. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992. doi: 10.1145/128756.128759.

Weber, C. and Triesch, J. From exploration to planning. In Kurkov, V., Neruda, R., and Koutnk, J., editors, *Artificial Neural Networks - ICANN 2008*, volume 5163 of *Lecture Notes in Computer Science*, pages 740–749. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-87535-2. doi: 10.1007/978-3-540-87536-9_76.

Weber, C. and Wermter, S. A self-organizing map of sigma-pi units. *Neurocomputing*, 70(13-15):2552–2560, 2007. ISSN 0925-2312. doi: DOI:10.1016/j.neucom.2006.05.014.

Weiller, D., Läer, L., Engel, A., and König, P. Unsupervised learning of reflexive and action-based affordances to model adaptive navigational behavior. *Frontiers in Neurorobotics*, 4(2), 2010. ISSN 1662-5218. doi: 10.3389/fnbot.2010.00002.

Wermter, S., Palm, G., and Elshaw, M., editors. *Biomimetic neural learning for intelligent robots : intelligent systems, cognitive robotics, and neuroscience*, volume 3575 of *Lecture Notes in Computer Science*. Springer / Heidelberg, 2005. ISBN 3-540-27440-5. doi: 10.1007/b139051.

Wermter, S., Weber, C., Duch, W., Honkela, T., Koprinkova-Hristova, P., Magg, S., Palm, G., and Villa, A. E. P., editors. *Artificial Neural Networks and Machine Learning ICANN 2014*, volume 8681 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014. doi: 10.1007/978-3-319-11179-7.

West, G., Newman, C., and Greenhill, S. Using a camera to implement virtual sensors in a smart house. In Giroux, S. and Pigot, H., editors, *From Smart Homes to Smart Care*, pages 83–90. IOS Press, 2005.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009. ISSN 0921-8890. doi: 10.1016/j.robot.2009.06.010.

Wilson, H. and Cowan, J. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik*, 13(2):55–80, 1973. ISSN 0023-5946. doi: 10.1007/BF00288786.

Wilson, R. and Finkel, L. A neural implementation of the Kalman filter. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in neural information processing systems*, pages 2062–2070. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3665-a-neural-implementation-of-the-kalman-filter.pdf`.

Witkowski, M. An action-selection calculus. *Adaptive Behavior*, 15(1):73–97, 2007. doi: 10.1177/1059712306076254.

Wolpert, D. M., Diedrichsen, J., and Flanagan, J. R. Principles of sensorimotor learning. *Nature Reviews Neuroscience*, 12(12):739–751, 2011. doi: 10.1038/nrn3112.

Wyss, R., Knig, P., and Verschure, P. F. M. J. A model of the ventral visual system based on temporal stability and local memory. *PLoS Biol*, 4(5):e120, 04 2006. doi: 10.1371/journal.pbio.0040120.

Yan, W., Weber, C., and Wermter, S. A hybrid probabilistic neural model for person tracking based on a ceiling-mounted camera. *Journal of Ambient Intelligence and Smart Environments*, 3:237–252, 2011. ISSN 1876-1364. doi: 10.3233/AIS-2011-0111.

Yan, W., Torta, E., van der Pol, D., Meins, N., Weber, C., Cuijpers, R. H., and Wermter, S. *Robotic Vision: Technologies for Machine Learning and Vision Applications*, chapter 15. Learning Robot Vision for Assisted Living, pages 257–280. IGI Global, 2012a. doi: 10.4018/978-1-4666-2672-0.ch015.

Yan, W., Weber, C., and Wermter, S. A neural approach for robot navigation based on cognitive map learning. In *Proceedings of the 2012 International Joint Conference on Neural Networks*, pages 1146–1153, Brisbane, Australian, June 2012b. IEEE. doi: 10.1109/IJCNN.2012.6252522.

Yan, W., Weber, C., and Wermter, S. Learning indoor robot navigation using visual and sensorimotor map information. *Frontiers in Neurorobotics*, 7(15), 2013. ISSN 1662-5218. doi: 10.3389/fnbot.2013.00015.

Yang, A., Jafari, R., Sastry, S., and Bajcsy, R. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, 1(2):103–115, 2009. ISSN 1876-1364. doi: 10.3233/AIS-2009-0016.

Zetzsche, C., Wolter, J., Galbraith, C., and Schill, K. Representation of space: image-like or sensorimotor? *Spatial Vision*, 22(5):409–424, 2009. doi: doi:10.1163/156856809789476074.

Zhang, B. and Muhlenbein, H. Synthesis of Sigma-Pi neural networks by the breeder genetic programming. In *Proceedings of the first IEEE Conference on Evolutionary Computation*, pages 318 –323, June 1994. doi: 10.1109/ICEC.1994.349933.

Zhang, J., Yan, Y., and Lades, M. Face recognition: eigenface, elastic matching, and neural nets. *Proceedings of the IEEE*, 85(9):1423–1435, September 1997. ISSN 0018-9219. doi: 10.1109/5.628712.

Zhong, J. and Sclaroff, S. Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, volume 1, pages 44–50, October 2003. doi: 10.1109/ICCV.2003.1238312.

Zivkovic, Z. and Krose, B. An EM-like algorithm for color-histogram-based object tracking. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 798–803, June 2004. doi: 10.1109/CVPR.2004.1315113.

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den, October 5, 2015 _____
(Unterschrift)

Declaration on oath

I hereby decleare, on oath, that I have written the present dissertation by my own and have not used other than the acknowledged resources and aids.

Hamburg, October 5, 2015 _____
(Signature)