TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatik IX
Computer Vision and Pattern Recognition

Representation, Acquisition and Use of Expectations for Domestic Service Robots

Michael Karg

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Darius Burschka
Prüfer der Dissertation:	1. Univ.-Prof. Dr. Daniel Cremers
	2. Jun.-Prof. Dr. Alexandra Kirsch, Eberhard Karls Universität Tübingen

Die Dissertation wurde am 16.06.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 03.11.2014 angenommen.

Abstract

Robots that are supposed to work together with people in everyday environments are confronted with a wide variety of situations. Not all such situations can be taken into account by a programmer when implementing the system. This is why robots often show strange behavior when encountering situations that their programmer(s) did not expect. We propose a knowledge-based approach to explicitly represent *expectations* in robot control programs. Comparing such expectations to the current situation allows a robot to detect unusual situations on its own and react appropriately. Our framework can incorporate expectations from different knowledge sources and allows to flexibly combine different expectations. We focus our work on the generation of expectations about habitual human behavior. By using models of human activities and a human activity recognition approach, we enable a robot to generate expectations about human task performance, which can be combined with various other expectations in the proposed framework. The validation of such expectations allows a domestic service robot to detect unexpected situations by combining all modeled expectations. We demonstrate the feasibility of the approach in simulated as well as in real-world experiments. We furthermore show how models of habitual human activities can be generated by a robot and how such models can be used in combination with activity recognition to generate and validate expectations about human behavior. The proposed framework can easily be integrated into a existing systems. It thus presents a first step towards equipping domestic service robots with a situational awareness, which enables them to better adapt their actions to the situation and their users.

Zusammenfassung

Roboter, die zusammen mit Menschen in Alltagsumgebungen arbeiten, sind mit einer Vielzahl von Situationen konfrontiert, von denen lediglich ein Bruchteil schon bei der Programmierung der Roboter berücksichtigt wurden. Deshalb zeigen solche Roboter in vielen solcher unvorhersehbaren Situationen oft ein für Menschen seltsam wirkendes Verhalten. In dieser Arbeit stellen wir einen wissensbasierten Ansatz vor, der Robotern die explizite Modellierung von Erwartungen ermöglicht. Der Abgleich solcher Erwartungen mit der aktuellen Situation erlaubt es einem Roboter, unerwartete Situationen zu erkennen, um angemessen darauf reagieren zu können. Unser allgemeiner Ansatz erlaubt die Integration und Kombination verschiedenster Erwartungen aus einer Vielzahl von Wissensquellen, wobei wir uns in dieser Arbeit speziell auf Erwartungen basierend auf menschlichem Gewohnheitsverhalten konzentrieren. Dazu stellen wir auch ein Verfahren zur Erstellung von Modellen von menschlichem Gewohnheitsverhalten vor und beschreiben, wie solche Modelle zusammen mit einem Ansatz zur Erkennung von Aktivitäten verwendet werden können. Diese Erwartungen können mit dem vorgestellten Framework mit anderen Erwartungen kombiniert und validiert werden, um einem Roboter die Erkennung von unerwarteten Situationen zu ermöglichen. Wir demonstrieren die Anwendbarkeit des Ansatzes sowohl in simulierten als auch in realen Experimenten und zeigen, wie Roboter durch das vorgeschlagene Framework mit der Fähigkeit ausgestattet werden können, verschiedene Situationen zu bewerten und unerwartete Situationen zu erkennen. Das vorgestellte Framework kann einfach in ein bereits vorhandenes Planungssystem für Roboter integriert werden und von heutigen und zukünftigen Robotersystemen verwendet werden um ihr Handeln besser an die Situation und die Menschen in ihrer Umgebung anzupassen.

Acknowledgements

This work would not have been possible without the help of many people who I want to thank on this page. First, I thank Juniorprof.Dr. Alexandra Kirsch who provided me with advice, encouragement and opportunities and at the same time gave me the possibility to freely follow my own ideas. I thank Prof.Dr. Daniel Cremers for his help and advice. I furthermore thank Prof.Dr. Michael Beetz and Dr. Jürgen Sturm for helpful advice whenever I had questions. I am very thankful for the great work with my colleagues Thibault Kruse and Christina Lichtenthäler from the Human Centered Artificial Intelligence Group. Further thanks go to the Intelligent Autonomous Systems Group. In particular, I thank Lorenz Mösenlechner, Tom Rühr, Alexis Maldonado, Uli Klank, Moritz Tenorth, Nico Blodow, Daniel Nyga, Mihai Dolha, Lars Kunze, Ingo Kresse, Dominik Jain and Karinne Amaro. I also thank the members of the CVPR group, in particular: Martin Oswald, Mohamed Souiai, Jan Stühmer, Evgeny Strekalovskiy, Frank Steinbrückner, Rudolph Triebel, Julia Diebold, Thomas Windheuser, Youngwook Kee, Claudia Nieuwenhuis and Eno Töppe. I thank all people involved in the MORSE simulator, specifically Séverin Lemaignan, Gilberto Echeverría, Arnaud Degroote, Sebastian Schmidt, Felix Ruess, and Florian Lier. Thanks go to Dr. Martin Sachenbacher, Sabine Wagner, Wibke Borngesser, Quirin Lohr my colleagues from LT-Z-4 at BMW Research and Technology as well as my former flatmate Martin for inspiring me with the surfing flatmate example. My deepest gratitude goes to my family for their love and support during my whole life. Finally, I thank Dani for her love during all these years.

This work received funding from the Institute for Advanced Study and the Cluster for Excellence Cognition for Technical Systems, funded by the German Excellence Initiative.

Contents

Chapter 1

Introduction

Intelligent robots that assist persons with unpleasant chores in their homes have been in the dreams of many people for a long time. Rapid progress in the areas of Artificial Intelligence and Robotics inspired researchers to spend a significant amount of work on making this dream a reality and bring service robots to human homes. Huge progress in the area of industrial robotics from 1973 on and especially during the last decades seemed to provide evidence that robots would enter human homes in the near future and household chores would be a relic of the distant past.

Today, more than 40 years after the first industrial robot was introduced, we still rarely see robots in areas that are inhabited by humans. Moreover, robots are still far away from being universal helpers for tasks like domestic chores. Although robots have widely succeeded in manufacturing environments, where they work separated from humans in controlled and well-defined environments, the transition towards dynamic and uncertain environments is far harder than one might expect. Designers of robots for controlled, industrial environments can foresee almost all possible events and errors that can occur during a robot's life and provide appropriate reactions to those. In dynamic environments, it is impossible to foresee all possible events that a robot might encounter. Persons living in the robot's environment introduce even more uncertainties and sooner or later, situations will occur that could not be foreseen when designing and implementing a robot. In the context of intelligent devices that interact with the real world, Norman [62] states: "We know two things about unexpected events: first, they always occur, and second, when they do occur, they are always unexpected." Norman also points out that "The same mechanisms that are so helpful

when things are normal can decrease safety, decrease comfort, and decrease accuracy when unexpected situations arise. For us, the people involved, it leads to danger and discomfort, frustration and anger." We think that those observations especially apply to domestic service robots, which we put to work in intimate spaces such as our homes. However, most of those robots today lack the ability to recognize unexpected events and react appropriately to them.

The research work in this thesis investigates a general approach to enable domestic service robots to identify unexpected events, enabling them to better react to such situations and adapt their behavior by doing the right thing at the right time. Humans are excellent in dynamically reacting to rapidly changing situations and, amongst other factors, this is due to the fact that persons make strong expectations towards all kinds of situations. Humans seem to constantly create expectations and validate them against the current situation. This not only enables us to detect and react on unexpected situations, but also to prevent failures before they even occur. In this thesis, we investigate an approach to bring such a predictive situation awareness to domestic service robots by the explicit representation and validation of expectations. We will show how expectations can enable robots to detect various kinds of unexpected events that can occur in domestic environments where robots and humans live together. Explicit models of expectations can enable robots to improve their awareness of persons and the situation in general and thus better adapt their behavior to human habits and dynamic environments. This important property can make domestic service robots more intelligent and pleasant for persons and contribute to the overall goal of bringing service robots into human homes in the near future.

1.1 Development of Service Robots

When thinking about a robot, most people think about human-like machines that move on their feet and seamlessly interact with their environment and persons. This image has been crafted by decades of robot-appearances in Hollywood movies and cartoons, where most robots are able to understand their environment and often communicate with persons or even other robots. But reality in robotics today looks quite different than what Hollywood is showing. Most of

today's commercial robots perform simple tasks in manufacturing, often without knowing about their environment or even human presence. And while robotic arms of manufacturing robots might be inspired by human arms, many robots don't have any human-like features at all. One reason for their missing perceptual capabilities and their non-human-like appearance is the fact that most of today's robots work strictly separated from humans, in environments that were designed specifically for them. As a consequence, their design has been driven purely by their functionality, accounting neither for a pleasant appearance nor advanced sensing capabilities. Such robots were never intended to work in areas created for human beings and thus do not posses any means of detecting persons or even interacting with them. But as the field of robotics is progressing, more and more research focuses on robots that are intended to work together with persons in human-like environments like medium-sized companies, hospitals or even the homes of persons.

Fortunately, not only researchers are interested in bringing robots closer to human-inhabited areas. The market for such service robots is predicted to increase dramatically within the next years as the estimated sales for the years 2011-2012 and the corresponding projection towards the year 2016 indicate. From 2011 to 2012, World Robotics (the statistical department of the International Federation of Robotics) recorded an increase of 53 % in the sales value of service robotics and predicts a huge continuing increase of sales of service robots for personnel/domestic use for the years 2013-2016[1]. While in 2012, 1.96 million domestic robots were sold, grossing a total of 697 million US$ according to the estimates, the projections for the years 2013-2016 for all types of domestic robots reach up to 15.5 million units with an estimated value of 5.6 billion US$ (see Figure 1.1).

As domestic service robots are intended to become available for a broad range of consumers, the majority of users of such robots will most likely be no experts in the area of robotics. With an increasing usage by non-expert users in areas populated by humans, the capabilities of such robots will have to grow. Today's service robots are only able to perform simple pick- and place tasks, mostly in laboratory environments after the application of a large amount of programming and expert knowledge. Tomorrow's service robots will likely be able to perform more

[1] http://www.worldrobotics.org/uploads/media/Executive_Summary_WR_2013.pdf

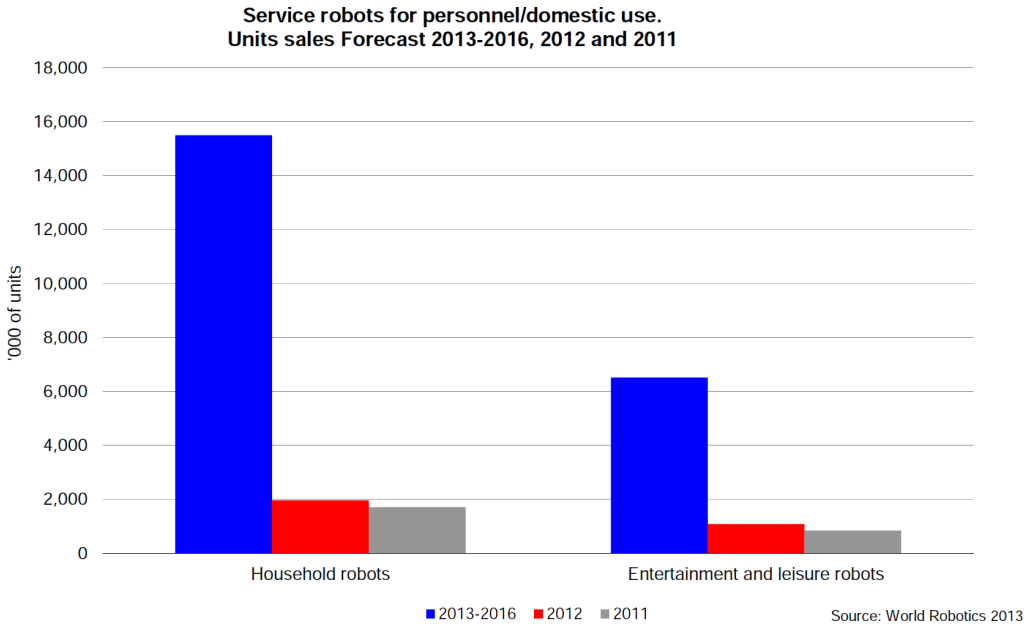


Figure 1.1.: Number of sold robots for domestic use in 2011 and 2012 and the projection of those number for the years 2013-2016. Image courtesy of World Robotics 2013

complex tasks together with persons that might not have any expert knowledge at all. Within the last four years, we have already witnessed strong development towards service robots that will someday be able to live with humans in their homes. The PR2 robot, shown in Figure 1.2 on the left, has been released in 2010 as a universal research platform that is designed to work in human-centered environments and marked a milestone in the development of personal robots. It was designed to not be harmful for persons that work close to the robot and was relatively cheap (compared to other robotic platforms at the time). In 2012, Rethink Robotics released the Baxter robot as a commercial robotics platform intended to work in direct proximity to persons and to be trained by demonstration from non-expert users. The UBR-1 robot by Unbounded Robotics (Figure 1.2 on the right) was released in 2013 to revolutionize affordable mobile manipulation by providing a cheap and reliable robotics platform that is suitable for use in domestic scenarios. The development of those robots within the last three years clearly contributed to the vision of making affordable domestic service robots a reality in a future that is not too far away.

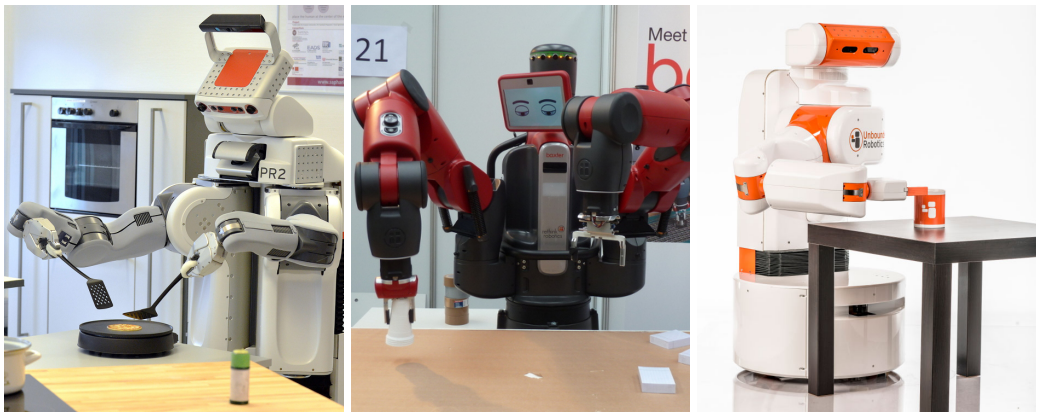


Figure 1.2.: Robots that were specifically designed to work together with humans. Left: The PR2 robot, one of the first universal robotic research platforms. Middle: The Baxter robot that is intended to provide a low-cost robot platform, which can be instructed without programming. Right: UBR-1 is a one-armed robot platform that is intended to revolutionize affordable mobile manipulation. Image courtesy of Michael Bahlo, DPA (right picture), Christina Hernàndez (middle picture), Unbounded Robotics (left picture).

1.2 Motivation

As future robotic applications will bring more robots into human homes and further progress in research will presumably enable domestic service robots to fulfill elaborate tasks, completely new possibilities for robotic systems will arise. But with new possibilities, robots will also be confronted with a wide variety of situations that have to be taken into account to guarantee that the robot's goals are achieved while constantly accounting for the safety of persons. Due to the infinite number of possible events that can happen, not all situations that a robotic system will experience can be taken into account when designing and implementing it. This fact is on the one hand caused by the unpredictable nature of life itself. On the other hand, an unpredictable situation can be produced by a human behavior that has not been foreseen. Even though persons can anticipate many events to a certain degree, no one can claim to be able to predict all situations one will encounter during his/her daily life. For a robot, such predictions, even of the simplest kind, are even harder to achieve. If a robot is confronted with a situation that has not been anticipated at design time, it will possibly keep executing its standard behavior until an error is detected. Robots that show such

behavior may look strange to persons and are not comprehensible for them, thus limiting their perceived reliability and causing problems in the mutual understanding. Imagine, for instance a robot that is intended to pass a glass of water to a person but loses the glass on its way. If the loss of the glass is not detected, the robot will possibly just reach out its empty hand to the person even if it detected a glass of water lying on the floor just before. This example shows just one possible unexpected situation that could happen at some point in a robot's life and calls for an appropriate response. Such a robot should recognize that something is not as expected when the glass of water, that is to be delivered to the person, is not in the robot's hand but lying on the floor (with a wet stain on the carpet next to it). Specifically, robots that share the environment with persons should be able to distinguish *normal* situations, where everything is as expected, from situations that deviate from the robot's expectations and possibly call for a reaction of the robot. This is due to the fact that, compared to an uninhabited area, a person introduces a high amount of dynamics into an environment. He/she will move around rather fast, possibly changing his/her directions in an unpredictable way and moving objects, leading to an increased frequency of situations that have not been anticipated. The ability to detect unexpected situations enables a robot to be of better service to the human by showing situation-aware behavior and reacting on and preventing possibly dangerous situations.

A robot that is a useful helper for humans should not only react to unexpected events caused by itself, but also have certain knowledge about human task performance to be able to take into account human behavior and react adequately to it. We think that this capability plays a key role for future robotic helpers that are to perform complex tasks in human-centered environments and is essential to enable commercial success of such robots. Imagine a household robot that is supposed to clean the kitchen after its user had dinner. Such a robot should be aware whether the human has already had dinner and not clean the table in a case where the human has not eaten yet. It should also not start vacuuming the floor when the person is watching TV or even sleeping.

In this thesis, we propose a novel approach to equip domestic service robots with the capability to detect unexpected situations by the validation of expectations. To this end, we introduce a framework that allows the modeling of various kinds

of expectations, where we focus on expectations about human behavior. We use models about human activities in combination with activity recognition to generate expectations about human task performance. We demonstrate how such expectations can be combined with a variety of other expectations to estimate the normality given situations.

1.3 Application Domain

The work in this thesis is tailored towards being applicable in any kind of environment, but we specifically focus on service robots that are intended for use in domestic areas. We believe that such a scenario represents a domain where robots can increase the quality of life of persons. Especially elderly people with minor disabilities are seen as beneficiaries of domestic robot helpers. Such robots could help them to live independently in their own home as long as possible which is being preferred to a life in retirement home by many people [81]. To gain first insights towards the acceptance of robots by elderly persons, Mitzner et al. [54] asked 21 independent living seniors for which activities they would prefer assistance rather from a robot than a human and found that this was the case for 28 out of 48 activities, which mostly involved household duties (e.g. cleaning and washing) or manual labor (e.g. gardening, mowing the lawn).

In such domestic scenarios, a robot will have to operate in a dynamic environment tailored towards the needs of persons. It will possibly collaborate with persons or at least have to take them into account when performing various activities. This calls for the need to detect and react on unexpected events instantly, thus possibly preventing dangerous situations. One simple example could be a person lying on the ground of the kitchen. In this case, a robot should detect that something is wrong and initiate a response, such as calling a doctor. One could also think of using a domestic service robot for surveillance during night times or when a person is not home. In this case, it would be beneficial if the robot could detect that a strange person, who is entering the flat at night and taking valuable things, might not be a normal situation but rather a burglary, which it should report.

1.4 Challenges

The problem of detecting unexpected events and failures is more difficult than it might look at first glance. In many robotics systems, a common way to tackle errors is to check for specific failures at specific points in time, often shortly after the robot has performed an action. But due to the unpredictable nature of the world and the multitude of errors that can occur, such an error detection results in a large number of "if ERROR then HANDLE_ERROR" constructs, even when trying to cover only a fraction of possibilities that might happen. As such error detection routines mostly have to be defined when creating a robot, errors that are not expected by the programmer at design time, will not be handled by the robot when they occur. Also, error checking at fixed points in time will lead to a robot not detecting failures that happen in between. In our scenario of a robot acting in an environment inhabited by persons, the probability of such errors will be significantly higher than in static environments. A person in the environment makes it very hard for a programmer to predict failures that might happen during the operation of the robot. Thus, a more general approach for the detection of unexpected events and failures would be desirable for service robots in general, but specifically for service robots sharing the environment with persons.

Domestic service robots that interact with persons in their environment should detect unexpected situations caused by abnormal behavior of their human partners. This implies that the robot has to be able to recognize human behavior at all. Since robots always have to deal with noisy sensor readings, current approaches in human activity recognition are mostly based on probabilistic approaches that are subject to uncertainties. Even if a robot is able to detect different activities that are executed by a person with high probability, it is still difficult to detect if the activity is carried out correctly since activities can be carried out in different ways. Thus, it is hard for a robot to know whether it detected an error in the execution of the currently observed activity or the person just did an activity in a different way. Since most robots have to rely on limited sensor data, this problem becomes even harder. Persons also tend to execute several activities at the same time, for instance cleaning some dishes while cooking, which makes it hard for a robot to distinguish between different activities.

Another important fact that a robot has to consider when working together with persons is the unpredictability of human behavior. Human behavior generally is hard to model and in many cases it is hard or even impossible to predict, even for other persons. Humans tend to sometimes quickly change their mind and thus their actions due to some thoughts or insights they gained spontaneously by observations or information sources like a television or radio. In such cases, the cause of the change in behavior is not observable and will inevitably lead to events that a robot cannot foresee but which it will possibly need to consider. But not every unexpected event necessarily has to be an error. Some events caused by unexpected human behavior might not require any actions of the robot at all, while others demand immediate action of a robot to prevent a dangerous situation. Imagine, for instance, a person unexpectedly interrupting his/her current activity because the doorbell rang. In this harmless case, a robot does not have to take any action, while in a situation where a person suddenly breaks down and falls onto the floor, an immediate action of a robot is necessary. In any of the cases, a prerequisite is that the robot is able to detect unexpected situations and unexpected behavior of persons.

The following chapters present first steps to enable the detection of unexpected events which hopefully will bring robots one step further towards a more general way of accounting for dynamic environments as well as the behavior of persons.

1.5 Contributions

This thesis demonstrates a way of enabling robots to integrate models of habitual human activities along with other types of information to create expectations about its surroundings and assess the normality of an observed situation. There are three main contributions that span the following chapters.

The first main contribution of this thesis consists of a general, modular and extensible framework enabling autonomous robots to explicitly model expectations and estimate the normality of a situation by the combination and validation of different expectations. Although, there is work using single expectations implicitly to detect anomalies, to our knowledge, so far there is no work that focuses

on the combination of a variety of different expectations to assess the normality of a situation. This capability can enable a robot to detect abnormal situations without the need of having an explicit model for each failure, which will ultimately free robot engineers from having to predict every possible failure that can happen to a robot.

The second main contribution focuses on the automatic generation of qualitative models of habitual human behavior from the observation of task performance. A robot that is able to create such models is able to distinguish different human activities from each other. Humans tend to pattern daily actions into sequences which they repeat at particular times in particular places [87]. Models of such habitual activities can serve as the input for a system for activity recognition and monitoring to ultimately enable a robot to generate and validate expectations using the aforementioned expectations framework.

The third main contribution consists of a system that performs activity recognition and monitoring in real-time in a spatially limited environment using a simple and cheap sensor setup and qualitative models of habitual human activities. The approach is able to generate predictions about likely future locations of persons performing habitual activities and integrate these into the expectations framework. This way, information about human behavior can be combined with arbitrary expectations to assess the normality of a situation, thus increasing a robot's situation awareness.

1.6 Reader's Guide

The chapters of this thesis build on each other and are thus best read in consecutive order. We first explain the theoretical foundations that were used in this work and continue with a description of a framework that enables the seamless integration of expectations into a robot's planning system. We then show how to generate models of human behavior to ultimately enable the generation of expectations. Afterwards, we propose an approach for activity recognition and monitoring that enables a robot to dynamically generate expectations using the human behavior models. Then, an overview over the datasets used for evalua-

tion is described and followed by a careful evaluation. Finally, the results of the thesis are discussed and concluded.

- *Chapter 2* features a collection of theoretical foundations that were used in this thesis.
- *Chapter 3* provides a detailed description of the expectations framework, including concepts as well as implementation details. It also describes a method for combining and validating different expectations to enable an autonomous robot to assess the normality of a situation in its planning framework.
- *Chapter 4* describes how an autonomous robot can generate models of human habitual behavior from the observation of motion tracking data.
- *Chapter 5* is about an approach for activity recognition and monitoring which enables a robot to generate predictions about human behavior based on models of habitual human behavior. We also explain how such predictions can be integrated into robot planning using the expectations framework.
- *Chapter 6* showcases existing datasets that were used for evaluation. We will furthermore describe two datasets that were created during this thesis.
- *Chapter 7* evaluates the contributions of this thesis and provides application scenarios to demonstrate the practical applicability of our work.
- *Chapter 8* discusses the results of the evaluations and assesses benefits as well as limitations of our work.
- *Chapter 9* concludes this thesis and addresses possible future directions after a short summary.

Some aspects of former versions of this work have been published in [32, 33, 31, 30, 28, 29, 27, 17, 47, 44, 34].

Chapter 2

Background

This chapter describes approaches and concepts on probabilistic reasoning and reactive planning, which the work of this thesis is based on. We assume the reader to be familiar with basic knowledge about probability theory. In particular, we follow the definitions and notation of Russel and Norvig [74] and refer the interested reader to their book. Especially chapters 13, 14 and 15 guide the reader from the basics of probability theory towards probabilistic reasoning over time, which is applied in this work. The following sections will first describe relevant concepts of probabilistic reasoning over time, in particular Hidden Markov Models, a variant of Bayesian networks. We will furthermore describe the Cognitive Robot Abstract Machine and its plan language that was used for the implementation of the approaches described in this thesis and explain recent research work in the area of autonomous semantic mapping.

2.1 Bayesian Networks

Bayesian networks define probability distributions over a directed graph where each node represents a random variable (discrete or continuous). The topology of the graph defines the conditional independence relationships between the random variables. Each node of the graph X_i has a conditional probability $P(X_i \mid Parents(X_i))$ that quantifies the effect of the parents on the node. Intuitively, an arrow or directed edge from the node X_i to node X_j means that random variable X_i has a direct influence on X_j. Given a Bayesian network, we can represent the full joint distribution $P(X_1, ...X_n)$ over all random variables X_i in the following way:

$$P(X_1, ...X_n) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i)) \tag{2.1}$$

The key advantage of modeling probability distributions using Bayesian Networks instead of the full joint distribution lies in the compact representation in most domains. Assuming, each random variable in a domain of n random variables is influenced by at most k other random variables, the complete Bayesian network can be described by $n2^k$ values in contrast to $2^n - 1$ values when using the full joint distribution. While for a fully connected graph (where $n = k$), this makes no difference, for many domains with a large number of random variables, the representation using Bayesian networks is significantly smaller. If we consider, for instance, a domain with 20 random variables, where each random variable is at most influenced by 5 other random variables, the full joint distribution has 1048575 values. A Bayesian Network of the same probability distribution can be described by only 640 values. The representation of probability distributions by Bayesian Networks thus makes the handling of domains with a large number of random variables feasible. The exemplary Bayesian Network illustrated in Figure 2.1 consists of the five random variables A,B,C,D and E. The full joint distribution $P(A, B, C, D, E)$ that is represented by the Bayesian Network can be described as the product of the probabilities of each node conditioned on its parents:

$$P(A, B, C, D, E) = P(A)P(B)P(C \mid A, B)P(D \mid C)P(E \mid C) \tag{2.2}$$

This representation enables us to describe the probability distribution by 10 values (instead of 31 values with the full joint distribution).

As the topology of a Bayesian network defines conditional independence assumptions between the nodes, we can draw several conclusions. First, a node is conditionally independent of its predecessors given its parents. In the example of Figure 2.1, for instance, the random variable D is independent of E given C. Furthermore, a node is conditionally independent of all other nodes given its Markov Blanket, which consists of the node's parents, children and all parents of

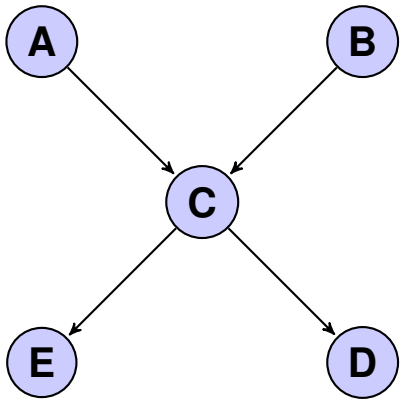


Figure 2.1.: An example of a Bayesian network consisting of the random variables A,B,C,D and E.

the children. Also, a node is conditionally independent of its non-descendants given its parents .

2.2 Hidden Markov Models

Random variables in Bayesian Networks can be related to each other over adjacent time steps. Bayesian Networks of such nature are also called *Dynamical Bayesian Networks (DBNs)*. DBNs are defined by a *transition model*. The transition model specifies the probability distribution over all state variables at time t given all previous states: $P(X_t \mid X_{0:t-1})$. $X_{0:t-1}$ is unbounded as the time t increases, which will lead to an increasing number of states at each time step and will ultimately make working with the DBN computationally unfeasible. To prevent this, we can use the Markov assumption that assumes the current state to only be dependent on a finite fixed set of predecessor states. Such processes are called Markov Chains or Markov Processes. The simplest Markov Process is a first order Markov process, where the current state only depends on one predecessor state.

For the Bayes Networks described so far, we assumed the states to be directly observable at any point in time. In many cases, the direct observation of a state is not possible. Instead, a system like a mobile robot has a variety of sensors which allow it to estimate its current state based on observations it obtained so far. The

states are not directly observable, but observations, which are dependent on the states, are.

A Hidden Markov Model (HMM) can be considered the simplest case of a DBN where each state – and thus each node of the Bayesian Network – is described by a single discrete random variable that is not directly observable. HMMs make use of the Markov assumption which allows for the following simplification of the transition model:

$$P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1}). \tag{2.3}$$

We assume the state to provide enough information to make future states conditionally independent of past states. This assumption might not hold for some systems, but it makes HMMs applicable to a large number of domains and works well for many applications. The Markov assumption relies on the assumption that changes in the world that influence our states are caused by stationary processes, meaning that the process of changes obeys laws that do not themselves change over time.

HMMs handle the indirect state observations $y_{1:t}$ by a *sensor model* that models the probabilities of each observation being made at each state. Also, the observations can depend on previous variables as well as the current state variable. As for the state transition model, the number of states would increase with the number of time steps. To prevent the computational load from increasing with each time step, the Markov assumption is also applied to the sensor model, making the current observation solely dependent on the current state:

$$P(Y_t \mid X_{0:t}, Y_{0:t-1}) = P(Y_t \mid X_t) \tag{2.4}$$

In addition to transition and sensor models, a HMM needs a prior probability distribution π modeling the probabilities over all states at time 0. This initial state distribution can have arbitrary formats depending on prior domain knowledge, but is often defined to be uniformly distributed when no prior knowledge is available.

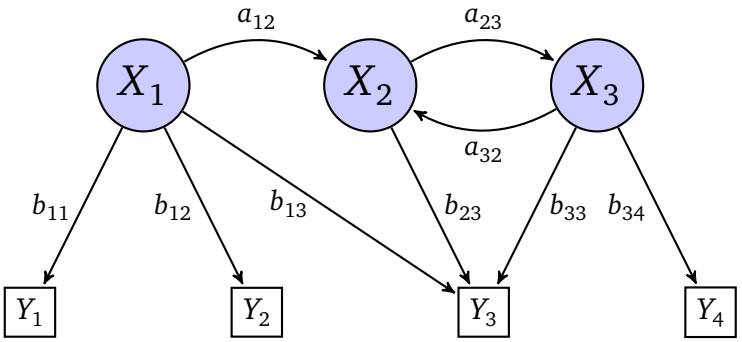


Figure 2.2.: An example of a Hidden Markov Model consisting of the random variables X_1, X_2, X_3 and observations Y_1, Y_2, Y_3, Y_4. Transition probabilities a_{ij} define the probability of a transition from node X_i to X_j while observation probabilities b_{ij} define the probability of observing Y_j while being in state X_i.

As a HMM is a variant of a Bayesian Network, we can compute the complete joint distribution over all states using equation 2.1:

$$P(X_{0:t}, Y_{1:t}) = \pi \prod_{i=1}^{t} P(X_i \mid X_{i-1}) P(Y_i \mid X_i). \tag{2.5}$$

The three terms in equation 2.5 are the prior π, the transition model $P(X_i \mid X_{i-1})$ and the sensor model $P(Y_i \mid X_i)$ and fully describe a HMM. Figure 2.2 shows an illustration of a HMM where the probabilities a_{ij} represent the probabilities of a transition from node X_i to X_j stored in the transition model $A_{ij} = P(X_t \mid X_{0:t-1})$. Probabilities b_{ij} correspond to the probability of making an observation Y_j while being at node X_i stored in the sensor model $B_{ij} = P(Y_t \mid X_t)$.

The Surfing Flatmate Example. Imagine living in a shared flat with your flatmate Martin that is a surfer. It is Wednesday evening and when you come home from work, you want to know if your flatmate has been surfing today. He is not home, so you cannot ask him directly but you can see if his wetsuit, which is hanging in the bathroom, is wet. You know that Martin has been surfing on Monday and that commonly, after he went surfing on one day, he doesn't go surfing the following day since he is exhausted. After having made a surfing-break one day, he commonly, if he has no other appointments, goes surfing the following day, since he otherwise starts feeling lazy. You can now represent such

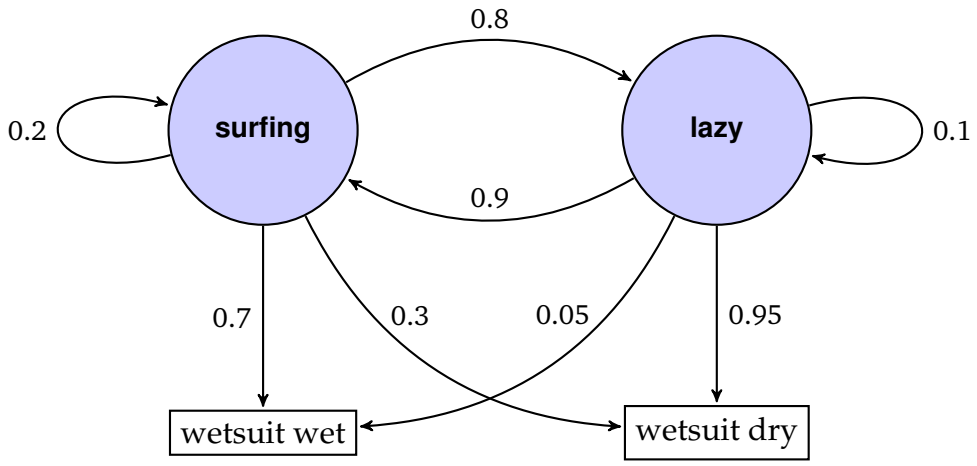


Figure 2.3.: This exemplary HMM models the dependencies of surfing days of an imaginary flatmate and the fact that his wetsuit is observed to be wet.

information in a HMM, modeling a surfing-day and a lazy-day as nodes and the transitions between surfing- and lazy-days in the transition model of the HMM. The fact that you know, your flatmate has been surfing on Monday can be represented in the initial state distribution of the surfing-days and lazy-days states. In our example, you cannot observe the states of the HMM directly and the wetsuit being wet does not guarantee that Martin was surfing. There might be warm summer days when he does not wear a wetsuit at all, so the wetsuit can be dry although Martin was surfing. Also, the wetsuit can be wet but Martin has not been surfing but washed his wetsuit. Such information can be represented in the sensor model of the HMM. Figure 2.3 illustrates the HMM that models the surfing days of the flatmate including the transition model and the sensor model with exemplary probabilities. Once this HMM defined, we can draw several conclusions about the surfing days of our imaginary flatmate, some of which will be described in the following sections. We will refer to the surfing flatmate example from time to time to provide a better understanding of the algorithms.

2.2.1 Inference in HMMs

Commonly, inference problems for temporal, probabilistic models, such as the HMM, are grouped into the following basic inference tasks:

- **Filtering:** The filtering task, also called the state estimation task, covers the calculation of the belief state, meaning the posterior distribution $P(X_t \mid y_{1:t})$ over the most recent state, given all instances of observations $y_{1:t}$ so far. Using the example of the surfing flatmate, this would estimate the probabilities of today being a surfing day or a lazy day given all observations of the wetsuit so far.

- **Prediction:** The prediction task computes the posterior distribution over a future state: $P(X_{t+k} \mid y_{1:t})$ for some $k > 0$. This task is often used to evaluate possible courses of action based on their expected outcomes. In the surfing flatmate example, this would correspond to the prediction of the probability of the flatmate to go surfing at some day in the future given all wetsuit observations so far.

- **Smoothing:** The smoothing task is about computing the posterior distribution over some past state: $P(X_k \mid y_{1:t})$ for some k such that $0 \leq k < t$. This provides a better estimate of the state than a calculation with information given at time k since more evidence is available. In the surfing flatmate example, the smoothing task would calculate the probability that the flatmate has been surfing at some day in the past given all observations of the wetsuit until the present day.

- **Most likely explanation:** Given a sequence of observations, this is the task of finding the sequence of states that is most likely to have generated the observations: $\arg\max_{x_{1:t}} P(x_{1:t} \mid y_{1:t})$. This corresponds to calculating a sequence of surfing and lazy days in the past that represents the most likely behavior of the flatmate given all wetsuit observations so far.

- **Learning:** The learning problem is essential for the generation of HMMs. It enables transition- and sensor models to be learned from observations using a variant of the Expectation Maximization (EM) algorithm [57]. An inference provides estimates about which transitions are most likely to have occurred and which states have most likely generated the sensor readings. These estimates can then be used to provide new estimates for the transition- and sensor model. The process iterates until it converges. The Baum-Welch Algorithm [73] is an instance of an EM algorithm that solves

the learning problem for HMMs and is described in section 2.2.3. In the surfing flatmate example, the learning task would provide the most likely parameters for the transitions between surfing and lazy days as well as the probabilities of the wetsuit being wet or dry given the flatmate has been surfing or not.

In the following, we will describe solutions to two of the problems that were of relevance for this thesis: The *learning task* and the *smoothing task*. A detailed description of all of the aforementioned problems lies beyond the focus of this thesis and we refer the interested reader to [74], chapter 15.

2.2.2 Forward Backward Algorithm

The Forward-Backward algorithm offers a solution to the smoothing problem and allows a more precise estimation of the posterior distribution $P(X_k \mid y_{1:t})$ than it is possible with filtering.

The Surfing Flatmate Example. The Forward-Backward algorithm in the surfing flatmate example calculates the probabilities that model if the flatmate has been surfing at one specific day given all wetsuit observations so far. For instance, on sunday we can calculate the probabilities of the flatmate having had a surfing or lazy day on Wednesday given the wetsuit observations over the whole week. Therefore, the forward step calculates the probabilities of the flatmate surfing on Wednesday by recursively filtering forward from Monday to Wednesday. Afterwards, the backward step calculates the probabilities of the flatmate having been surfing on Wednesday based on the wetsuit observations from Thursday to Sunday. The results of the forward- and backward step are combined and offer a more accurate estimation than it would have been possible when considering only observations from Monday to Wednesday.

2.2.2.1 Formal Description

The Forward-Backward Algorithm [72] calculates the posterior distribution over all past state variables $P(X_k \mid y_{1:t})$ given a sequence of observations $y_{1:t}$ for $0 \leq$

$k < t$ up to the present. The computation of $P(X_k \mid y_{1:t})$ can be split into two parts:

$$\begin{aligned} P(X_k \mid y_{1:t}) &= P(X_k \mid y_{1:k}, y_{k+1,t}) \\ &= \alpha\, P(X_k \mid y_{1:k})\, P(y_{k+1:t} \mid X_k, y_{1:k}) \\ &= \alpha\, P(X_k \mid y_{1:k})\, P(y_{k+1:t} \mid X_k) \\ &= \alpha\, f_{1:k} \times b_{k+1:t}. \end{aligned} \tag{2.6}$$

We apply Bayes' rule (see appendix, section A.4) in line two of equation 2.6 and use a conditional independence assumption between $y_{1:k}$ and $y_{k+1:t}$ given X_k in line three. α represents the normalization factor from the application of the Bayes rule. In line four, we define two new functions: The forward message $f_{1:k}$ and the backward message $b_{k+1:t}$. $\times$ represents point-wise multiplication of vectors.

Forward Step. The forward message $f_{1:k}$ can be computed by filtering forward from 1 to k. A filtering algorithm maintains and updates an estimation of the current state. It recursively computes the result for time $t+1$ from the result of the filtering up to time t and a new observation y_{t+1} for some function f:

$$P(X_{t+1} \mid y_{1:t+1}) = f(y_{t+1}, P(X_t \mid y_{1:t})). \tag{2.7}$$

This process is called recursive estimation and can be decomposed into two parts in the following way:

$$\begin{aligned} P(X_{t+1} \mid y_{1:t+1}) &= P(X_{t+1} \mid y_{1:t}, y_{t+1}) \\ &= \alpha\, P(y_{t+1} \mid X_{t+1}, y_{1:t})\, P(X_{t+1} \mid y_{1:t}) \\ &= \alpha\, P(y_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid y_{1:t}). \end{aligned} \tag{2.8}$$

We make use of the Bayes' rule in line two and the Markov assumption of the sensor model in line three. The second term of the resulting two parts is the

prediction of the next state given the observations so far $P(X_{t+1} \mid y_{1:t})$, which is updated with the new observation y_{t+1}. The first term $P(y_{t+1} \mid X_{t+1})$ is directly given by the sensor model. We can now calculate a prediction of the next state X_{t+1} conditioned on the current state X_t:

$$\begin{aligned} P(X_{t+1} \mid y_{1:t+1}) &= \alpha\, P(y_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1} \mid x_t, y_{1:t})\, P(x_t \mid y_{1:t}) \\ &= \alpha\, P(y_{t+1} \mid X_{t+1}) \sum_{x_t} P(X_{t+1} \mid x_t) P(x_t \mid y_{1:t}). \end{aligned} \tag{2.9}$$

We use the Markov assumption of the transition model in the second line. The resulting form is the desired recursive formulation where the first term of the result of equation 2.9 is obtained from the sensor model, the first part in the summation is given by the transition model and the second part in the summation is defined by the current state distribution. We define the calculation of the filtered state estimate $P(X_t \mid y_{1:t})$ as a recursive function that is propagated forward among the sequence of observations by modifying each transition and updating with each new observation:

$$f_{1:t+1} = \alpha\, FORWARD(f_{1:t}, y_{t+1}). \tag{2.10}$$

Here, $FORWARD$ represents the implementation of the filtering described in equation 2.9 (we exclude the normalization constant α here). Given, all state variables are discrete, the time for each update is constant and so is the required memory. The forward step itself is also a solution to the filtering/state estimation task since it computes the belief state over the most recent state $P(X_t \mid y_{1:t})$.

The Backward Step. The backward message $b_{k+1:t}$ in equation 2.6 is computed recursively using a process that runs backward from t:

$$
\begin{aligned}
P(y_{k+1:t} \mid X_k) &= \sum_{x_{k+1}} P(y_{k+1:t} \mid X_k, x_{k+1}) P(x_{k+1} \mid X_k) \\
&= \sum_{x_{k+1}} P(y_{k+1:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) \\
&= \sum_{x_{k+1}} P(y_{k+1}, y_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) \\
&= \sum_{x_{k+1}} P(y_{k+1} \mid x_{k+1}) P(y_{k+2:t} \mid x_{k+1}) P(x_{k+1} \mid X_k) \qquad (2.11)
\end{aligned}
$$

We condition $P(y_{k+1:t} \mid X_k)$ on x_{k+1} in the first line and use the conditional independence assumption of the observations y_{k+1} and X_k given x_{k+1} in the second line. In the last step, we use the conditional independence of y_{k+1} and $y_{k+2:t}$ given X_{k+1}. The resulting summation consists of three factors where $P(y_{k+1} \mid x_{k+1})$ is directly obtained from the sensor model and $P(x_{k+1} \mid X_k)$ is given by the transition model. The term in the middle is the recursive part of the equation. Defining the update described by equation 2.11 as *BACKWARD*, we can describe the backward step in the following recursive formulation:

$$
b_{k+1:t} = BACKWARD(b_{k+2t}, y_{k+1}). \qquad (2.12)
$$

As for the forward step, time and memory needed for each update are constant, and thus independent of t, when the states are discrete. The backward step is initialized with $b_{t+1:t} = P(y_{t+1:t} \mid X_t) = P(\mathbf{1} \mid X_t)$, where $\mathbf{1}$ is a vector of 1s. Since at initialization of the backward step, $y_{t+1:t}$ is an empty sequence, the probability of observing it is 1. Given equations 2.10 and 2.12, the two terms $f_{1:k}$ and $b_{k+1:t}$ can both be computed by recursions in time. In case of the forward step, the recursion runs forward from 1 to k and in case of the backward step, the recursion runs backward from t to $k+1$.

2.2.2.2 Implementation

Both recursions needed to calculate the forward and the backward step require a constant amount of time per step, leading to a complexity of $O(t)$ at a particular time step k. Smoothing the whole sequence by running the smoothing process for each time step would lead to a time complexity of $O(t^2)$. Using a simple application of dynamic programming can reduce the complexity for smoothing the whole sequence to $O(t)$. This is done by recording the results of the forward step over the whole sequence and reusing them in the backward recursion. Thus, we can compute the smoothed estimate at each time k from the computed backward message $b_{k+1:t}$ and the stored forward message $f_{1:k}$. In the activity recognition approach that we will introduce in chapter 5, we use an implementation of the forward-backward algorithm as illustrated by Algorithm 1.

Algorithm 1 Forward-Backward Algorithm

Require:
obs, vector of observations
prior, initial state distribution π
1: $forward[0] \leftarrow prior$
2: **for** $i = 1$ **to** t **do**
3: $forward[i] \leftarrow FORWARD(forward[i-1], obs[i])$
4: **end for**
5: **for** $i = t$ **to** 1 **do**
6: $states[i] \leftarrow NORMALIZE(forward[i] \times backward)$
7: $backward \leftarrow BACKWARD(backward, obs[i])$
8: **end for**
9: **return** *states*

The local variable *forward* represents the state distribution as a vector of state probabilities after performing the forward step. The local variable *backward* accordingly contains the state distributions after the backward step. The local variable *states* represents the state distribution after the forward and backward step as a vector of state probabilities, which is returned at the end of the algorithm. The functions $FORWARD$ and $BACKWARD$ implement the forward- and backward-step according to the definitions in equations 2.10 and 2.12. The function $NORMALIZE$ ensures that the summation of all state probabilities is 1 by calculating the normalization constant α of equation 2.6.

One should mention that the forward-backward algorithm comes with two practical drawbacks. The space complexity can become high with large state spaces and long observation sequences. Furthermore, it does not work in online settings where smoothed state estimates need to be computed for earlier times as new observations are continuously added to the end of the sequence. Those drawbacks lie beyond the scope of this thesis and we refer the interested reader to [74], chapter 15 for a discussion of solutions.

2.2.3 Baum-Welch Algorithm

The Baum-Welch algorithm [73] is named after Leonard E. Baum and Lloyd R. Welch and offers an iterative procedure to solve the learning problem by adjusting the parameters of the HMM to maximize the probability of the observation sequence given the model. It uses the Expectation Maximization (EM) algorithm [57] to calculate the maximum likelihood estimate of the parameters of the HMM given an observation sequence.

The Surfing Flatmate Example. In our example, the Baum-Welch algorithm provides an estimation about the most likely parameters of the HMM. In the surfing flatmate example, the learning task would provide the most likely parameters for the transitions between surfing and lazy days as well as the probabilities of the wetsuit being wet or dry given the flatmate has been surfing or not. Therefore, the Baum-Welch algorithm first computes the probability of the flatmate having been surfing at a specific day given all wetsuit observations so far. It uses the same forward- and backward steps as the Forward-Backward algorithm explained in section 2.2.2. For instance, it calculates the probability of the flatmate having surfed on Wednesday given the wetsuit observations of the whole week until Friday. It therefore uses initial distributions for the transition and sensor model which are either distributed uniformly or randomly if we have no prior knowledge available. They can also include prior information if we have a rough guess about the models. Next, the Baum-Welch algorithm iteratively updates the model parameters of the HMM, meaning the transition model, the sensor model and the initial distribution until it converges at a point where the model parameters are most consistent with the observations. Therefore, the algorithm uses

the observations to estimate a new transition matrix by comparing the expected total number of transitions from any state to any other stated based on the initial transition model. In the example, this would correspond to comparing our guess of the probability of a surfing day being followed by a lazy day with the probabilities estimated from the observations. Furthermore, an updated sensor model is estimated based on the expected numbers of observations at each state. The parameters of the HMM are updated with the new transition- and sensor model and the steps are repeated until the parameters reach a desired level of convergence.

2.2.3.1 Formal Description

At the beginning of the algorithm, the transition model A_{ij}, the sensor model B_{ij} and the initial distribution π are assigned random initial values if no prior information is present. If prior information is available, those parameters can also be set accordingly. The Baum-Welch algorithm can be divided into two steps: the *forward-backward step* and the *update step*.

The Forward-Backward Step. In the first step, we calculate the probabilities α_i^k of seeing a partial sequence $y_{i:k}$ and ending up in state i at time k given the current parameters θ of the HMM by recursively filtering forward as described in section 2.2.2.1.1:

$$\alpha_i^k = P(y_{1:k}, X_k = i \mid \theta). \tag{2.13}$$

Furthermore, the probability of ending the observation sequence $y_{k+1:t}$, given we started in state i at time k, is calculated using the Backward-step as explained in section 2.2.2.1.2:

$$\beta_i^k = P(y_{k+1:t} \mid X_k = i, \theta). \tag{2.14}$$

The Update Step. We define the temporary variable γ as the probability of being in state i at time k given the observation sequence $y_{1:t}$:

$$\begin{aligned}
\gamma_i(t) &= P(X_k = i \mid y_{1:t}, \theta) \\
&= \frac{P(y_{1:t} \mid X_k = i, \theta)\, P(X_k = i \mid \theta)}{P(y_{1:k} \mid \theta)} \\
&= \frac{P(y_{1:t} \mid \theta)\, P(X_k = i \mid \theta)}{P(y_{1:k} \mid \theta)} \\
&= \frac{P(y_{1:t}, X_k = i \mid \theta)}{P(y_{1:t} \mid \theta)} \\
&= \frac{P(y_{1:t}, X_k = i \mid \theta)}{\sum_{j=1}^{N} P(y_{1:t}, X_k = j \mid \theta)}.
\end{aligned} \tag{2.15}$$

We use Bayes rule in the second line of equation 2.15 and conditional independence of $y_{1:t}$ and X_k given θ in the third line. The fourth line applies the product rule (see appendix, section A.3) and the sum in the fifth line arises using the law of total probability. Given α_i^k from equation 2.13 and β_i^k from equation 2.14, we can write:

$$\begin{aligned}
\alpha_i(k)\beta_i(k) &= P(y_{1:k}, X_k = i \mid \theta)\, P(y_{k+1:t} \mid X_k = i, \theta) \\
&= P(y_{1:k}, X_k = i \mid \theta)\, P(y_{k+1:t} \mid \theta) \\
&= P(y_{1:t}, X_k = i \mid \theta).
\end{aligned} \tag{2.16}$$

We use the Markov assumption of the sensor model and the conditional independence of $y_{k+1:t}$ and X_k given X_{k+1} in the second line. Thus, equation 2.15 can be defined in terms of α_i^k and β_i^k as follows:

$$\gamma_i(k) = \frac{\alpha_i^k \beta_i^k}{\sum_{j=1}^{N} \alpha_j^k \beta_j^k}. \tag{2.17}$$

We describe this term by the temporary variable γ_i^k, which represents the probability of being in state i at time k as explained in equation 2.15. We furthermore introduce the second temporary variable ξ_{ij}^k as follows:

$$\begin{aligned} \xi_{ij}^k &= P(X_k = i, X_{k+1} = j \mid y_{1:t}, \theta) \\ &= \frac{\alpha_i^k a_{ij} \beta_j^{k+1} b_j(y_{k+1})}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m^k a_{mn} \beta_n^{k+1} b_n(y_{k+1})}. \end{aligned} \tag{2.18}$$

ξ_{ij}^k represents the probability of being in state i at time k and being in state j at time $k+1$ given the observation sequence $y_{1:t}$ and the model parameters θ. We can now start to update the model parameters $\theta = \langle \pi, A, B \rangle$ by first calculating the new initial distribution π_i^* as the expected frequency spent in state i at time 1:

$$\pi_i^* = \gamma_i(1). \tag{2.19}$$

In the next step, the expected number of transitions from state i to j compared to the expected total number of transitions away from state i is computed:

$$a_{ij}^* = \frac{\sum_{k=1}^{t-1} \xi_{ij}^k}{\sum_{k=1}^{t-1} \gamma_i^k}. \tag{2.20}$$

Also, new emission probabilities $b_i^*(k)$ are calculated for each state i at time k:

$$b_i^*(k) = \frac{\sum_{l=1}^{t} 1_{y_l = v_k} \gamma_i^l}{\sum_{l=1}^{t} \gamma_i^l}. \tag{2.21}$$

Here, $1_{y_k=v_k}$ represents an indicator function which is defined as follows:

$$1_{y_l=v_k} = \begin{cases} 1 & \text{if } y_l \equiv v_k \\ 0 & \text{otherwise.} \end{cases} \tag{2.22}$$

The steps from equation 2.17 to equation 2.22 are repeated and iteratively compute new values for the model parameters θ of the corresponding HMM until a desired level of convergence is reached, thus obtaining the most likely model parameters according to the observations. One should note that the Baum-Welch Algorithm does not guarantee to find a global maximum but rather local maxima. However, there are many applications in which the parameters computed by the Baum-Welch algorithm provide good results as the evaluation of the work of this thesis will show in chapter 7.

2.2.3.2 Implementation

One possible implementation of the Baum-Welch algorithm is illustrated in Algorithm 2. The algorithm has a vector of observations as input as well as an initial guess of the parameters of the HMM. One will probably recognize that lines 1 to 7 of the algorithm look almost the same as the initial lines of the Forward-Backward algorithm illustrated in Algorithm 1 in section 2.2.2.2. This comes as no surprise since the forward- and backward probabilities that are used for the calculation of the updated parameters are computed in exactly the same way. In line 8 to 14, the parameters of the HMM are recalculated iteratively as explained by equations 2.17 to 2.22 until a desired level of convergence is reached. The updated transition model *new_tm* is calculated as explained by equation 2.20, the updated sensor model *new_sm* is computed as explained by equation 2.21 and the updated initial distribution *new_prior* is derived as explained by equation 2.19. At each iteration, the level of convergence is computed by comparing the parameters of the HMM before the update with the updated parameters. Once, a desired level of convergence is reached, the algorithm ends and returns the parameters of the HMM.

Algorithm 2 Baum-Welch Algorithm

Require:
obs, vector of observations
prior, initial guess initial state distribution π
tm, initial guess of transition model
sm, initial guess of sensor model

1: $forward[0] \leftarrow prior$
2: **for** $i = 1$ **to** t **do**
3: $forward[i] \leftarrow FORWARD(forward[i-1], obs[i])$
4: **end for**
5: **for** $i = t$ **to** 1 **do**
6: $backward \leftarrow BACKWARD(backward, obs[i])$
7: **end for**
8: **while** *convergence* $>$ *threshhold* **do**
9: $new\_tm \leftarrow NEW\_TRANSITION\_MODEL(forward, backward)$
10: $new\_sm \leftarrow NEW\_SENSOR\_MODEL(forward, backward)$
11: $new\_prior \leftarrow NEW\_PRIOR(forward, backward)$
12: $convergence \leftarrow CONVERGENCE(tm, sm, new\_tm, new\_sm)$
13: $tm, sm, prior \leftarrow new\_tm, new\_sm, new\_prior$
14: **end while**
15: **return** $tm, sm, prior$

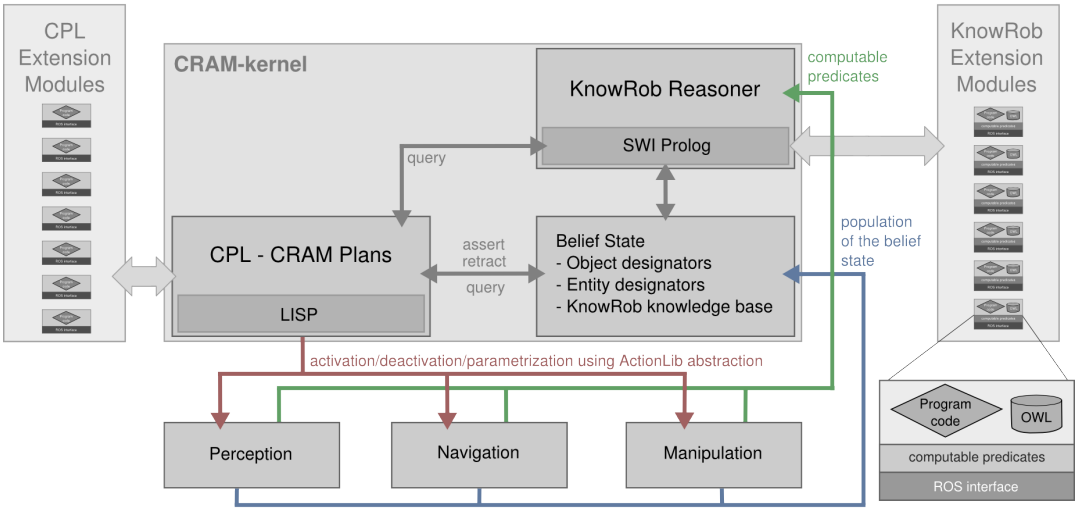


Figure 2.4.: The CRAM architecture consists out of the light-weight CRAM kernel, the KnowRob knowledge processing system as well as several modules that control the access to the robot's sensors and actuators. The CRAM kernel can easily be extended by CPL extension modules. (Image courtesy of Michael Beetz [6])

2.3 Reactive Planning

Classical approaches in planning often try to achieve correctness and optimality and therefore require a detailed model of the world.Thus, they are often subject to rather long planning times in large domains. However, if we consider robots working together with humans, such requirements can be problematic. The real world consists of an unlimited variety of situations and possible actions and cannot be captured in a restricted model. Also, a robot working together with a human will in many cases only have a limited amount of time for planning available. For many cases, a quick, possibly non-optimal reaction of a robot is preferred over an optimal reaction that the robot executes too late.

2.3.1 The Cognitive Robot Abstract Machine (CRAM)

Beetz et al. [5], introduced the Cognitive Robot Abstract Machine (CRAM), as a toolkit for the specification of complex robot behavior for cognitive mobile robots. It comes with the CRAM Plan language (CPL), which offers a reactive

planning framework and allows to program plans as modules that are stored in a plan library. CRAM is specifically tailored to the needs of autonomous robots performing complex everyday manipulation activities in human-centered environments. It has been designed to enable the effective and efficient implementation of higher-level capabilities like learning, knowledge processing and action planning into robot control programs to produce flexible, reliable and efficient behavior. The CRAM architecture, as illustrated in Figure 2.4, consists of the lightweight CRAM kernel, the KnowRob knowledge base and several modules for robot capabilities and extensions. It is based on the open-source robotics middleware ROS [71] for communication between the different modules. The CRAM kernel is defined to be lightweight, standard and configurable, extending the CommonLISP programming language. It uses LISP compilers instead of defining an own interpreter. The CRAM kernel consists of only 3000 lines of code and can easily be extended with optional modules to provide additional cognitive capabilities. One such example is the knowledge processing framework KnowRob [83]. It is implemented as an extension to the CRAM kernel and can be added or removed as needed. CRAM equips robots with lightweight reasoning mechanisms that can infer control decisions rather than requiring them to be preprogrammed. This enables robots to be much more flexible, reliable and general than robots that lack such capabilities. Furthermore, CRAM does not need the whole domain to be explicitly stated. Instead, it relies on means of grounding symbolic expressions into a robot's perception and actuation routines.

In several demonstrations of robots performing tasks of daily living in a laboratory kitchen environment, CRAM has repeatedly been used on real robots as illustrated in Figure 2.5. In particular, the two robots Rosie and James were capable to make pancakes, sausages, popcorn, a sandwich and go shopping using the CRAM architecture [4].

2.3.2 The CRAM Plan Language

The CRAM framework comes with the CRAM Plan Language (CPL), which is an expressive behavior specification language enabling autonomous robots to not only execute their control programs but also reason about and manipulate

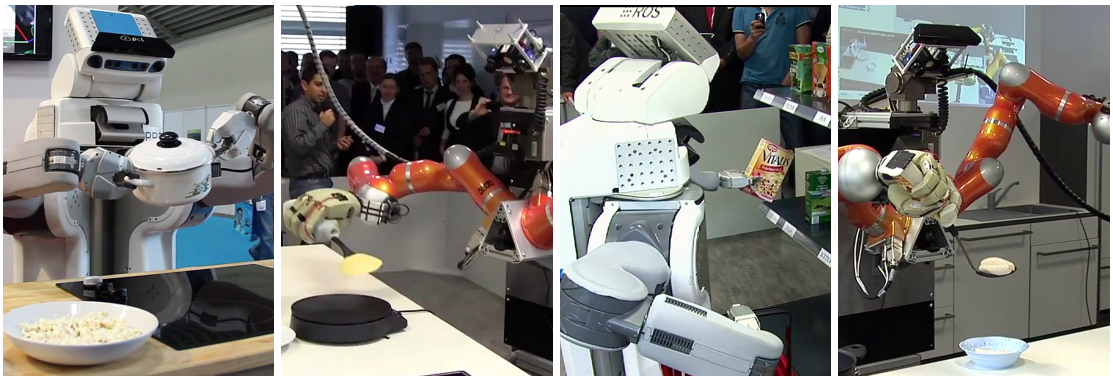


Figure 2.5.: The two robots Rosie and James performing everyday tasks in a laboratory kitchen using the CRAM architecture.

them - even during execution. It is based on the Robot Plan Language (RPL) [51] but specifically focuses on the control of real robots performing complex manipulation tasks in human-centered environments. Robot plans written in the CPL language specify concurrent, sensor-guided, reactive behavior and let us define how a robot should respond to sensory events, changes in its belief state and detected plan failures. The desired world-state has an explicit representation that lets a robot check if a goal already holds before trying to achieve it. Furthermore, a robot can validate if a routine is successful by perceiving the state after achievement. Instead of regarding plans as a partially ordered set of atomic actions, CRAM specifically considers concurrency, action synchronization, failure handling, loops and reactiveness. Therefore, CPL offers control structures for parallel execution, partial ordering of sub plans, sophisticated failure handling and the annotation of control programs. The multithreading library enables complex synchronization, partial orderings, blocking and in particular clean failure handling that propagates from sub-plans to parent-plans.

One important aspect of CPL is the fact that not only the goal, but also many control decisions, including complex reasoning actions, are turned into first-class objects, meaning entities that parameterize the plan and can be queried, analyzed, and reasoned upon. This is achieved by so called *designators* which describe decisions by a set of constraints that have to be satisfied. Designators represent objects, locations and actions by conjunctions of symbolic properties and are resolved only at the very last moment when the decision actually needs to be taken.

This allows the system to base a decision on all information that can be gathered until the code that needs to take the decision gets executed.

The second important concept that we use in this thesis are *fluents*. Fluents are variables that are used to describe objects and can be equipped with values e.g. derived from low-level sensor input. They can be *pulsed* which allows them to wait for changes of its values and notify the high-level layer. Commonly, fluents from ongoing sensor values are produced by *process modules* that provide the interface to the low-level sensors of the robot. The use of fluents enables a robot to react to changes without actively monitoring each object or sensor reading since the fluent itself takes care of the monitoring and notifies the robot when changes occur.

The work described in the remainder of this thesis is based on CPL and will implement an extension to CPL that allows a robot to model and validate different types of expectations to enable a robot to distinguish normal from abnormal situations as we will show in chapter 3.

2.4 Semantic Environment Maps

Any mobile robot that is intended to perform more sophisticated tasks than randomly driving around will need some representation of its environment. One of the most common kinds of such a representation are two dimensional gridmaps [58]. Two dimensional gridmaps divide the environment into a grid where each cell represents the probability of a cell being free or occupied by an obstacle. Such maps can be generated from two dimensional sensor data of a laser scanner, sonar or similar and are the representation of choice for robot localization and navigation.

However, when it comes to sophisticated tasks like manipulation tasks in human-centered environments, a two dimensional representation of the environment is often not sufficient. A robot needs an elaborate environment representation to be able to perform complex mobile manipulation activities. Such a representation should include the three dimensional structure of its surroundings as well as semantically annotated knowledge about their environment like information about

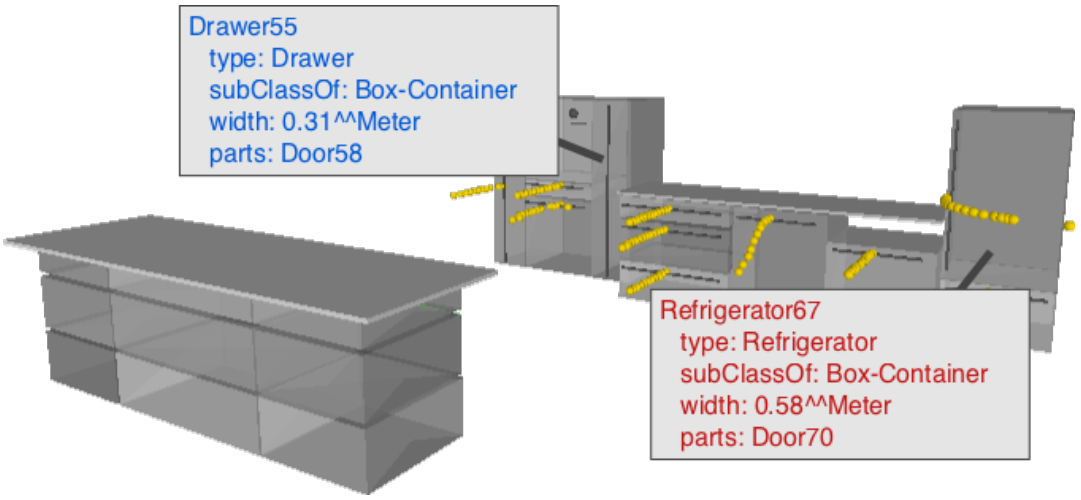


Figure 2.6.: Semantic object map of a kitchen including three dimensional models of furniture objects as well as articulation models for the opening of doors (illustrated by the yellow points). (Image courtesy of Moritz Tenorth [65])

objects, object-types, doors and so on. Recent development in autonomous semantic mapping allows robots to have semantically annotated knowledge about their environment by autonomously generating *semantic maps* using laser scans, 3D-pointclouds and cameras. While some approaches focus on detecting room categories [96], others include 3D CAD models that categorize objects, geometric information about objects, articulation models for opening doors and functional components like the handle of doors [9]. These semantic maps can widely be generated autonomously by the calculation of next best view poses. From these, 3D pointclouds are gathered and the application of various segmentation methods generate initial hypotheses for furniture, drawers and doors as illustrated in Figure 2.6. Tenorth et al. [85] offer a way to create knowledge-linked semantic object maps that combine information of semantic object maps with common sense knowledge of publicly available data bases. This enables a robot to do reasoning about the objects in its environment, like querying its database for common locations of objects in its specific environment.

In this thesis, we will use semantic object maps to obtain knowledge about the location, dimensions and types of furniture objects like cupboards and drawers. This information will help us to estimate a spatial model about where persons generally are located during pick- and place actions from container objects as we

will show in chapter 4. Semantic object maps are integrated into KnowRob and can thus be conveniently queried from within the CRAM architecture.

Chapter 3

A Framework for Expectations

In this chapter, we will introduce a framework that generally enables a robot to represent different kinds of expectations and provide a situational awareness based on various sources of information. This capability can enable a robot to distinguish normal from abnormal situations and help it to ultimately adapt its planning to react on or even prevent possible failures.

Since it is hard to find a general definition of a failure, we propose a general notion of *expectations* that can be met to certain degrees in a situation. Such expectations can be represented explicitly in the robot program using different knowledge sources to describe facts that the robot expects in the current situation. Figure 3.1 illustrates some categories of expectations and gives an impression about how expectations can be used to classify situations. The image on the left shows objects that are floating in the air, which violates the laws of physics and would contradict a continuous naive physics reasoner. A human lying on the floor obeys the laws of physics but is unusual behavior as long as it is not observed at the beach or in similar context. The right image shows a less surprising situation, although a pile of boxes on a table might not be very common in a typical household (while they would be expected in places like a warehouse). The causes of unexpected situations can be of various kinds. They might be caused by failures in the robot program (for instance due to unreliable action execution of inaccurate sensors), unexpected human behavior like dropping objects or forgetting things (the latter being specially relevant in the care of people with dementia), or completely different events like the emergence of a storm.

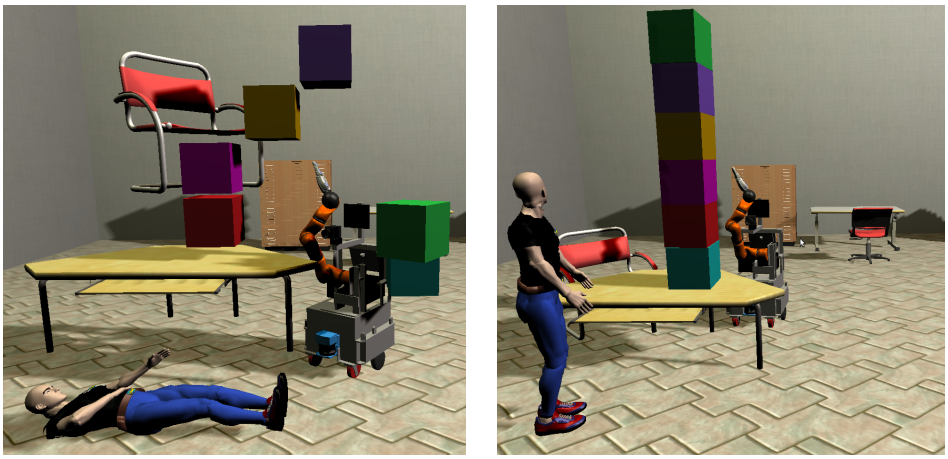


Figure 3.1.: Left: An "abnormal" simulated scene in human robot interaction. Right: A "normal" simulated scene in human robot interaction.

As Orthony et al. [64] stated as early as in 1987, the registration of and the reaction to failed expectations, failed assumptions and unanticipated events are crucial components of general intelligence. This chapter introduces a framework that enables a robot to combine different kinds of knowledge to explicitly represent expectations. These expectations can be of different kinds according to the type of knowledge that is available, and the requirements of specific applications. Promising candidates to provide such knowledge are naive physics reasoning [1], simulation-based projection [40], geometric reasoning [59] or knowledge-based inference [41]. Each of these approaches use single expectations in a rather implicit way to model specific parts of situations and detect anomalies, but none of them offers the possibility of combining several expectations to enable a combined, general representation of the normality of situations. Figure 3.2 shows an overview about the use of expectations for estimating of the normality of a situation. Different knowledge sources generate expectations that are maintained in the Expectations Framework. An estimation about the normality of the current situation is returned from the Expectations Framework by the validation of the available expectations.

An unexpected situation is not necessarily an error or a failure and some unexpected events may not require a reaction of the robot at all. Imagine a person and a robot cooking together in the kitchen when suddenly the doorbell rings and the human leaves the room. In this case, the doorbell might not have been

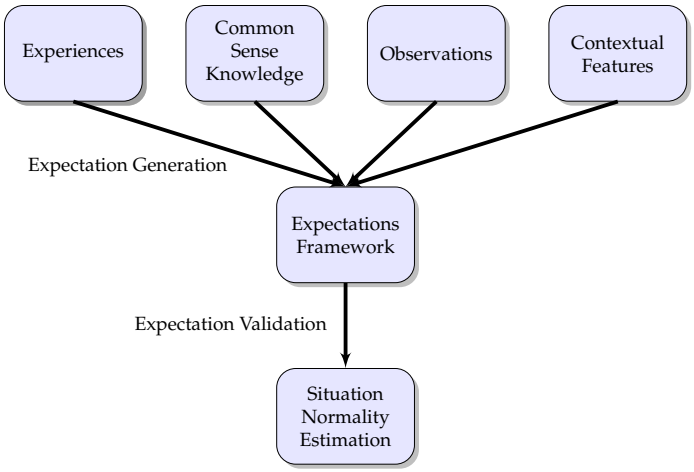


Figure 3.2.: An overview of our system for using expectation to estimate the normality of situations. Different expectations are generated and validated to enable a robot to obtain an impression about the normality of a given situation.

expected but it might not represent a situation which requires a response of the robot. Other cases like a person lying on the ground or the detection of smoke in the living room may require clarification actions of the robot such as asking the human for assistance or locating the source of smoke.

In this work, we do not deal with the reactions to unexpected situations, but rather offer a general and modular framework that enables robots to represent normal situations by a combination of different types of expectations. Our framework offers the possibility to detect unexpected situations and their degree of unusualness by validating expectations and gives hints for possible solutions by explicitly representing which expectations are currently violated.

3.1 System overview

We define an expectation as any piece of knowledge that describes normal mechanisms of the world, such as physical laws, robot behavior or habitual human behavior. The *normality* of a situation is assessed by validating the available expectations. The validation of an expectation can be seen as a comparison between what has been expected and what has really happened. It returns a value indicating to what degree the corresponding expectation has been fulfilled. The values

of all expectations are combined in one overall value that expresses the average normality of the current situation.

To keep the handling of expectations as simple as possible, our framework offers the same interface for all expectations regardless of their type. From an implementation point of view, this interface is defined by an abstract class *expectation* that defines the interface. Specific expectations are created by sub-classing the abstract expectation class. Each expectation class has to implement a validation method that provides a measure to which degree the corresponding expectation has been fulfilled by returning a value between 0 and 1. We will call this value the *normality* of an expectation. If the validation of an expectation returns a normality of 1, the expectation has been perfectly fulfilled, whereas a value of 0 means that the expectation has not been fulfilled at all. Since the validation of different types of expectations can be necessary at different points in time, each expectation can trigger the validation by a validation hook, which is implemented depending on the type of expectation. The validation hook must be created by the client application and ensures that expectations are generated and updated at the correct point in time. Consider, for instance, expectations about general physical laws in contrast to expectations about possible future locations of a person. While expectations about physical laws should probably be validated on a regular basis, expectations about possible future locations of a person can only be validated as soon as the next location of the person is known. If we expect a person to go to the dining table in the next step, knowing that he/she just finished preparing a meal, we can only validate this expectation as soon as we know where the person actually went, while we can constantly validate if objects obey physical laws like gravity.

The expectations framework also offers the possibility to group expectations, creating categories of expectations that are treated as composite objects and offer the same interface as single expectations, similar to the *Composite* pattern from object oriented software design patterns [21]. Composite expectations also implement a validation method, which recursively validates all expectations in the category and combines their normality values to one single value. This combined value expresses to what degree the expectations of the corresponding category have been fulfilled. One could, for instance, think about the creation of an expecta-

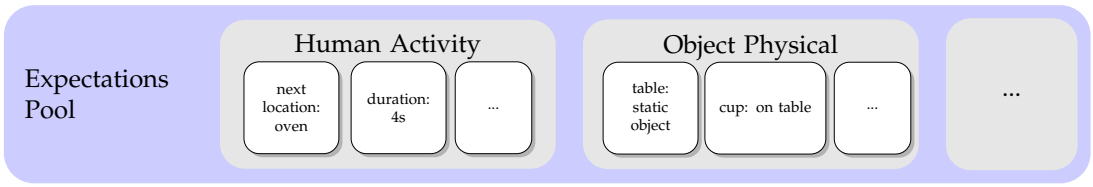


Figure 3.3.: Illustration of an expectations pool including expectation categories. The white squares represent single expectations grouped in different expectations categories which are illustrated as light blue squares.

tions category "Object Physical" to group expectations about physical properties of objects in the environment. Due to the recursive structure of expectation categories, it is also possible to create categories of expectations that include other categories of expectations, thus creating nested structures. One might for example use an expectations category "Object Physical" as in the example before and at the same time have a group of expectations "Object Locations", modeling likely locations of objects. Those two expectation categories can themselves be aggregated in a category called "Object Expectations".

All expectations are maintained in an *expectations pool* where they can dynamically be added and removed depending on the context of the current situation. If a robot's activity recognition, for instance, detects that a human just finished an activity like preparing a meal and will probably start eating the prepared meal, the robot would remove the expectation that it expects the human near the oven, and add a new expectation expecting the human to be at the dining table. An exemplary expectations pool containing different types of expectations grouped in categories is depicted in Figure 3.3.

The introduction of expectation categories as well as the different types of expectations themselves strongly depend on the scenario of application. Figure 3.4 illustrates the class structure of single expectations and expectation categories. While all expectation classes share the same interface defined by the *Expectation* class, only classes of *ExpectationCategory* implement the methods that are specific to categories like *Add*, *Remove* and *GetChildren*. This class structure allows us to represent the expectation pool as a tree structure where expectations represent nodes of the tree. Single expectations correspond to leaf nodes, while composite expectations represent non-leaf nodes. Due to the way expectation categories are

implemented, we can recursively define structures of expectations with an arbitrary number of levels. However for our experiments, we limited the number of expectations levels to less than three.

3.1.1 Expectation Models

Using the expectations framework, arbitrary types of expectations can be created as long as they conform to the interface defined in the class structure. For our application of a robot helper in a typical human household, we introduce three different types of expectations: *Logical Expectations*, *Temporal Expectations* and *Probabilistic Expectations*.

3.1.1.1 Logical Expectations

Logical expectations are composed of logical propositions that either hold in the current situation or are violated. The validation of a logical expectation will return 1 if the proposition holds and 0 otherwise. An example of a logical expectation is: *Cup_1 on the table*, which expresses that the robot would expect the object called *Cup_1* being located on the *table*. This can either be true or false. Generally, logical expectations can be seen as binary logical propositions that return their truth-value when they are validated.

3.1.1.2 Temporal Expectations

Temporal expectations only hold for a certain duration. They are an extension of logical expectations and consist of logical propositions and a duration during which the expectation is expected to hold. When a temporal expectation is validated, the duration during which a fact holds is compared with the duration the fact is expected to hold. As long as the measured duration of a fact is smaller than the expected duration, the situation is considered to be normal and the validation returns 0. When the measured duration exceeds the expected duration, the normality decreases with time until it reaches 0. By default, we define the

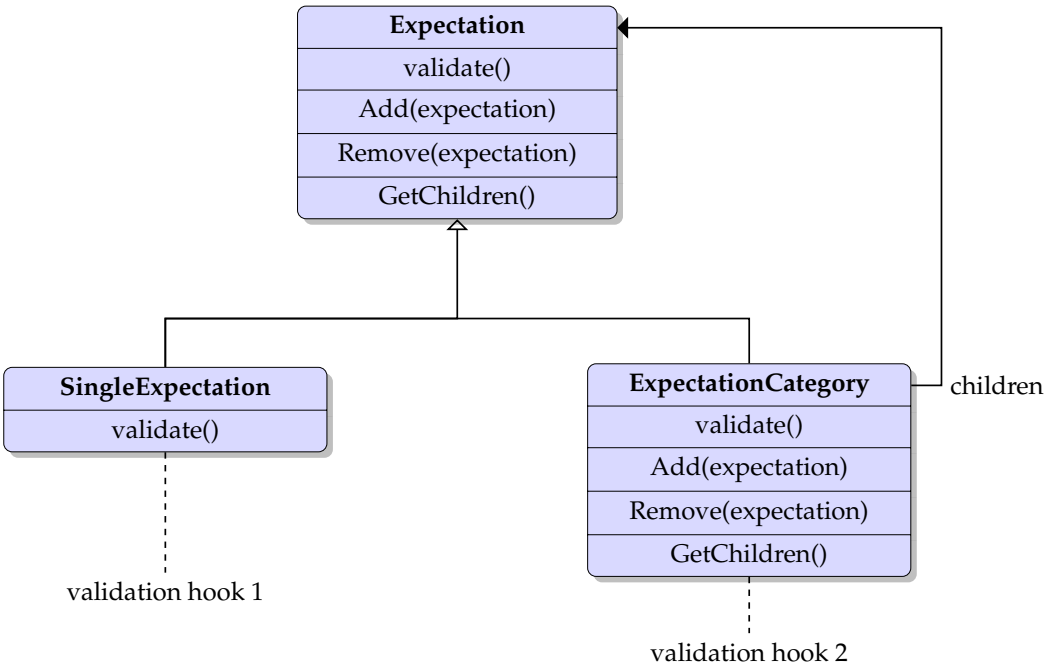


Figure 3.4.: The class structure of our implementation of expectations. The abstract class *Expectation* defines the common interface that all concrete expectations implement. Each expectation consists of a validation method returning a value between 0 and 1, indicating the degree to which the expectation has been fulfilled. *SingleExpectation* classes can directly model expectations about any type of information that can be perceived by the robot. *ExpectationCategory* classes implement compositions of expectations that enable the client to group expectations in categories. In addition to a validation method, *ExpectationCategory* classes also implement functions for adding and removing expectations as well as accessing the expectations in the category. Validation of an *ExpectationCategory* is done by recursively validating all expectations in the category.

decrease of the normality linearly but other functions are possible as well depending on the application. An example of such an expectation is the duration that a human spends at certain locations when performing activities. One might for example expect a person to usually be located approximately 4 seconds in front of a cupboard when picking up objects from it. When a person is standing longer in front of the cupboard, it becomes more unlikely that he/she is actually performing a pick- and place action.

3.1.1.3 Probabilistic Expectations

Not in all situations we are certain about single events or actions that we expect to happen. Sometimes we cannot clearly state what will happen next, instead there will be several possibilities for future events or actions of which we are aware, possibly some more probable than others. In such cases, we can use *Probabilistic Expectations* which have probability distributions over random variables instead of binary propositions assigned to them. Using *Probabilistic Expectations*, we can describe a set of events that we expect to happen and assign each event a probability, expressing how likely we think the corresponding event is to occur. For example, a robot might observe a person being at the table, and, given it knows about the typical behavior of that person, the person will commonly clean the table after breakfast, before going to work. But there is also the possibility that he/she will directly go to work without cleaning the table (e.g. when he/she is late) or go to the bathroom (e.g. when he/she is late but not too late to brush his teeth). Such multiple possibilities of expected events can be modeled with a *probabilistic expectation*. The validation of a probabilistic expectation returns a value between 0 and 1 corresponding to the probability assigned to the random variable (representing one event) of the expectation that is observed to be true. Possible implications of this validation method are discussed in section 8.1.

3.1.2 Expectation Validation

Each single expectation has a validation function which, when invoked, expresses to what degree an expectation has been fulfilled by returning a value between 0

and 1. We call this value the *normality* n_i of an expectation e_i. But we are not only interested in the normality value of single expectations, but want to use the concept of expectations to describe the normality of a situation that consists of several different facts and events by combining different expectations. So we combine the normality values of all available expectations in the current situation. Therefore, we construct a *normality tree* from all expectations in the expectations pool. The normality tree contains single normalities that result from the validation of single expectations as well as combined normalities that are generated by the validation of expectation categories. The combined normalities are computed by recursively validating each expectation in the expectations category with the validation function $g_{validate}$ and combining the corresponding values as shown in equation 3.1.

$$n_i = \begin{cases} g_{validate}(e_i), & \text{if } e_i \text{ is single expectation} \\ f_{combine}(g_{validate}(e_i)), & \text{else} \end{cases} \tag{3.1}$$

To this end, the way all normalities n_i are combined is defined by a combination function $f_{combine}$. For our experiments, we defined $f_{combine}$ to simply be the average of all expectations returned from the validation of a combined expectation as shown in equation 3.2, but one could as well think about other techniques, such as weighted sums, thresholds or maximum.

$$f_{combine} = \frac{\sum_{i=1}^{n} n_i}{n} \tag{3.2}$$

The overall normality is calculated by combining all expectations in the expectations pool. We therefore calculate the overall normality of the situations – which is also the single value on the highest level in the normality tree – according to equation 3.1 and 3.2.

The structure of the normality tree is defined by the structure of the expectations, so the number of levels of the expectations tree is basically unlimited. However, in the experiments we conducted, we mostly used three layers as shown in the exemplary normality tree in Figure 3.5. In this example, a situation is observed that mostly fulfills the robot's expectations, resulting in a high overall normality

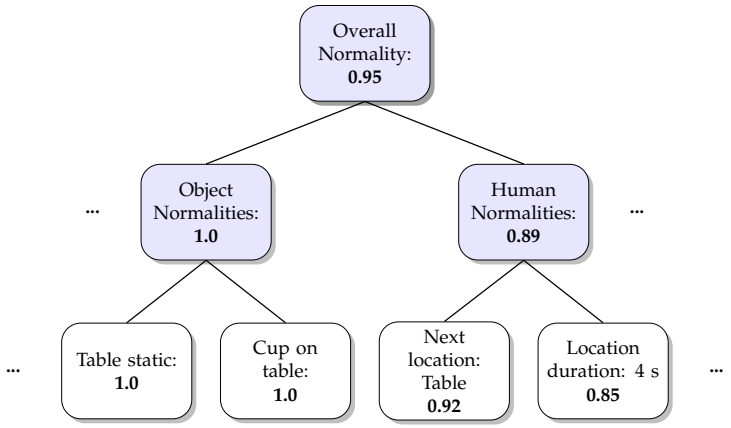


Figure 3.5.: An exemplary normality tree generated by the implementation of different expectations of a household robot. The highest level represents the overall normality of the situation by combining the normalities of all expectations in the expectations pool. The lower levels allow for a more fine-grained representation of specific expectations. Blue nodes represent composite expectations that contain several single expectations, which are illustrated as white nodes.

of 0.95. The Figure shows two expectation categories, “Human normalities” and “Object normalities”, which consist of two single expectations each. The robot observes the table to not have moved and the cup being on the table, so the two logical object expectations hold, both returning a normality of 1. Accordingly, the category ”Object normalities“ has a combined normality of 1. The probabilistic normality ”Next locations: Table“ returns a value of 0.92 indicating that the robot correctly expected the human to move to the table next with a probability of 0.92. The temporal expectations ”Location duration: 4s“ returns the value 0.85 meaning that the human is staying at the current location for a little longer than 4 seconds already. Averaging over the ”Human normalities“, the composite expectation has a normality value of 0.89 and combining all expectations, we obtain an overall normality of 0.95 indicating that in general, the current situation mostly fulfills the robot’s expectations and we consider the situation as quite normal. As soon as one of the normalities changes it value, the normality tree is updated ensuring that the robot maintains a constant situational awareness that is up-to-date.

The representation of the normality of situations using the normality tree provides a limited but general way of finding the cause of unexpected events. A

Figure 3.6.: A simulated robot assistant in a kitchen environment.

robot could, for instance, consider to take action when the overall normality drops below a certain threshold. It could also traverse the normality tree to find out which expectation has not been fulfilled and use this information as a hint for finding the cause of surprise and take actions (like clarifying the situation, possibly with human help).

3.2 Application: Expectation classes for Human Robot Interaction

The following scenario demonstrates how a domestic service robot can benefit from expectations using the proposed framework. We will assume to have a robot assistant that is expected to work in the kitchen, executing tasks that are inconvenient for its human owner, like setting the table or cleaning the table after meals. A simulation of such a scenario is illustrated in Figure 3.6. Using knowledge about typical places of objects like tableware required for breakfast, the robot can generate expectations about these objects and validate them. Such information can be learned from observations or inferred from common sense knowledge as we will discuss in section 8. Reasonable expectations can, for in-

stance, include likely storage locations of objects before, during and after the meals, or simply the fact that commonly, objects like tableware or cutlery are expected to be on the table for breakfast. Such a robot could, for instance, generate logical expectations expressing that we expect an object to be located on the table after a table-setting activity. To generate such an expectation, we create a *logical-expectation* expressing an expected location for an object. Objects can be described by fluents of a type that include the position of the object as explained in section 2.3.2. The modeling of objects as a fluent has the advantage that their position is updated as soon as it changes, thus the expectation can be validated each time a change is detected. The validation function checks if the object is located on the expected location e.g. either simply realized using a comparison of two dimensional coordinates or using more elaborate approaches like a naive reasoner. The validation of the logical expectation returns 1 if the object is at the expected location and 0 in the other case. Similarly, we can define a *temporal expectation* modeling the duration that a human is expected to be standing at a certain location using models of human task performance as we will explain in chapter 4. Furthermore, a robot can generate probabilistic predictions about likely future locations of a person performing certain activities of daily living and use such predictions to generate a *probabilistic expectation*.

To evaluate the applicability of our expectations framework, we set up several different exemplary environments for a domestic robot helper and equip the robot with different types of expectations similar to the ones shown in this chapter. We furthermore set up a real world scenario (using a dataset described in chapter 6) that combines expectations about human task execution and an approach for activity recognition (described in chapter 5) to dynamically generate and validate expectations depending on the context of the situation. The goal of these experiments is to enable a robot to distinguish normal situations from unexpected situations by validating different types of expectations about the environment, the human, the robot, etc. The experiments as well as the results of the validation and the insights we gained are described in chapter 7.

3.3 Related Work

The proposed expectations framework follows the aim of offering a general approach for failure detection for autonomous robots. Research in the areas of model-based programming [78] and discrete control theory [35] follow similar objectives, where mostly technical systems are described by formal models such as state automata with probabilistic transitions or Petri nets. Failure states are often modeled explicitly, enabling a system to avoid them or diagnose reasons for failures when they are encountered.

In the field of robotics, such approaches are commonly used for diagnosing internal faults of system components of robots. Gerald Steinbauer [77] introduces *observers* to perform model based diagnosis on robot components without affecting the control system, while Kuhn et al. [39] propose the paradigm of pervasive diagnosis, simultaneously enabling active diagnosis and model based control. Using model based autonomy, Wiliams et al. [92] enable autonomous systems to be aware of their states and possible failures in uncertain environments. Utilizing models of nominal system behavior as well as models of common failures, they perform extensive reasoning to recognize and recover from errors. The approaches mentioned so far consider only errors in components that are part of the robot, so called *internal faults* of technical systems. Akhtar and Kuestenmacher [1] use qualitative reasoning on naive physics concepts for diagnosis for the prediction of errors that do not occur in system parts of the robot itself, but in its environment, so called *external faults* of autonomous systems.

Such approaches are not suitable for failures that occur in high-level behavior of autonomous service robots. The definition of possible interactions between a human and a robot in a finite state machine would mean a huge modeling effort. In contrast to discrete control theory, where a formal model can often be directly generated from system specifications (usually given in a work-flow programming language), such an automaton would have to be hand coded in the case of a robotic system that interacts with humans. That means that a system designer would have to take into account all failure situations that can possibly occur when modeling the robot behavior. Even more important, in many situations it is impossible for a robot to decide whether it is in a failure state or not.

A typical failure state for a domestic service robot, that is cooperating with a human, would be "human has abandoned task" which can be indicated by the human partner suddenly leaving the room. But the fact that a person has left the room does not necessarily mean that he/she has really abandoned the task. The person could as well just fetch a necessary object or answer the doorbell. Modeling on a direct observation-level would also lead to extremely complex models, which would have to include the context of situations.

The proposed expectations framework enables autonomous robots to detect failures without an explicit model of failure states. Similar to models of nominal behavior of technical systems proposed by Wiliams et al. [92]. They model potential failures as observations that are not compliant with the robot's experience and knowledge about how the world and in particular humans should behave. The combination of different evidence and the consideration of the degree of divergence from the robot's expectations prevent the robot from being overcautious. Other work on such explicit use of expectations has been proposed by Minnen et al. [53]. They find that humans have strong prior expectations about actions in activities and propose that technical systems can use explicit expectations about high-level activities to improve activity recognition. Therefore, they use extended stochastic grammars to constrain an activity recognition approach that recognizes a human playing "Towers of Hanoi" from video data by analyzing object interaction events. Maier and Steinbach [50] make use of expectations for autonomous robots to enable a mobile robot to detect unexpected scenes from video data. They generate a dense image-based map of the robot's environment and compare luminance and chrominance values of the images at different time points. This enables a robot to detect changes in its environment and make assumptions about the uncertainties and expectations.

Lately, the field of Goal Directed Autonomy (GDA) has gained attention in the robotics community and some work includes the use of expectations. Goal directed agents generally are engaged to deliberation to select which goal among several concurrent goals they should choose to achieve and adapt their behavior accordingly. Molineaux et al. [55] use a discrepancy detector to compare the current situation with expectations and use the output of the discrepancy detector as an indicator for the creation of new goals. Kurup et al. [43] make use of ex-

pectations that are constantly generated to match them against observations and enable cognitive systems to react when inconsistencies are detected. They implement their system in the ACT-R cognitive architecture and evaluate it by creating expectations about the tracks of pedestrians that cross an intersection. Piunti et al. [69] use expectations to generate affective states, thus equipping goal directed agents with adaptive and anticipatory abilities. Therefore, expectations are modulated with emotions like excitement, disappointment, relief, etc. that influence the desirability of goals based on an anticipatory feeling.

All of the approaches work well for diagnosing faults in specific domains. But to our knowledge there is no approach that incorporates a combination of different models, including humans, to enable robots to perform diagnosis on cooperative everyday tasks. The expectations framework we presented in this chapter is designed to comprise ideas from all of the aforementioned fields and enables an autonomous robot to generalize its failure recognition by the combination and validation of explicit models of expectations.

Chapter 4

Human Activity Models

As Tapus et al. find [81], socially assistive robots that are intended to cooperate with persons in their homes must be aware of human presence and their preferences. Moreover, they should produce behavior that is not only acceptable, but also appealing for their human user. Hence, service robots that work together with persons should have an understanding about the activities of their human partners in order to better adapt to their user's behavior thus being a more efficient and user-friendly helper. A prerequisite for the capability of a robot to understand human activities and show adaptive and user-friendly behavior are models of human activities which are ideally learned from observations. We claim that robots that are intended to live together with humans can widely benefit from knowledge about human activities on which they can base their planning.

This chapter describes an approach for equipping robotic systems with models of human activities to ultimately allow a better integration of humans into planning frameworks for autonomous robots. One step towards this goal is the autonomous generation of human activity models from observations of human task performance. Such models can, for instance, be used by an activity recognition system as we will show in chapter 5 thus allowing a robot to explain observations of human motion tracking data in terms of known activities. Information about observed activities can then be included into the expectations-framework introduced in chapter 3 to generate and validate expectations. This enables a robot to distinguish common behavior of persons from unexpected behavior as we will show in chapter 5 and 7.

We will first explain, how *Context Dependent Spatial Regions (CDSRs)* can be used to provide a qualitative, human-like description of spatial areas and then present an approach for the generation of *Spatio-Temporal Plan Representations (STPRs)* from the observation of motion tracking data and semantic environment maps. Those STPRs serve as general models of typical activities that are performed by a person in his/her home. Furthermore, we will show, how STPRs can be transferred to different, but similar environments such as different kitchens and how STPRs can directly be compared based on different features to allow for a simple recognition of repeatedly executed activities.

4.1 Context Dependent Spatial Regions (CDSRs)

Most of today's robotic approaches use representations of the environment that are designed for specific purposes and are often quite different to the way humans might think of their environment. Approaches for robot navigation are mostly based on two dimensional grid maps. For object manipulation, three dimensional environment representations based on point clouds or Octrees [24] are common representations. Such representations are in most cases purely geometrical and lack qualitative, semantic information about entities in the environment. With robots more and more working together with persons, the difference between representations used by robots and humans can cause difficulties in the mutual understanding which the cartoon in Figure 4.1 illustrates in a humorous way. In contrast to many approaches in robotics, humans often think of spatial regions in terms of their functional use. Instead of relating positions in the environment to one fixed, global origin, humans tend to think of locations relative to other objects in their environment. A person might, for example, rather say: "Get the butter from the refrigerator" instead of "Go to global coordinate 2,3,0 and move your hand to point 2.5,3.5,1.4, ...". As indicated by this example, humans often seem to find furniture objects to be suitable reference points for locations, especially when referring to objects that are stored in or on the corresponding furniture objects. Robots mostly use coordinate-based representations of locations. Thereby, the coordinates usually are in reference to a specific reference frame like the origin of a two dimensional gridmap or the origin of a sensor. While such representations are perfectly suitable for a variety of basic

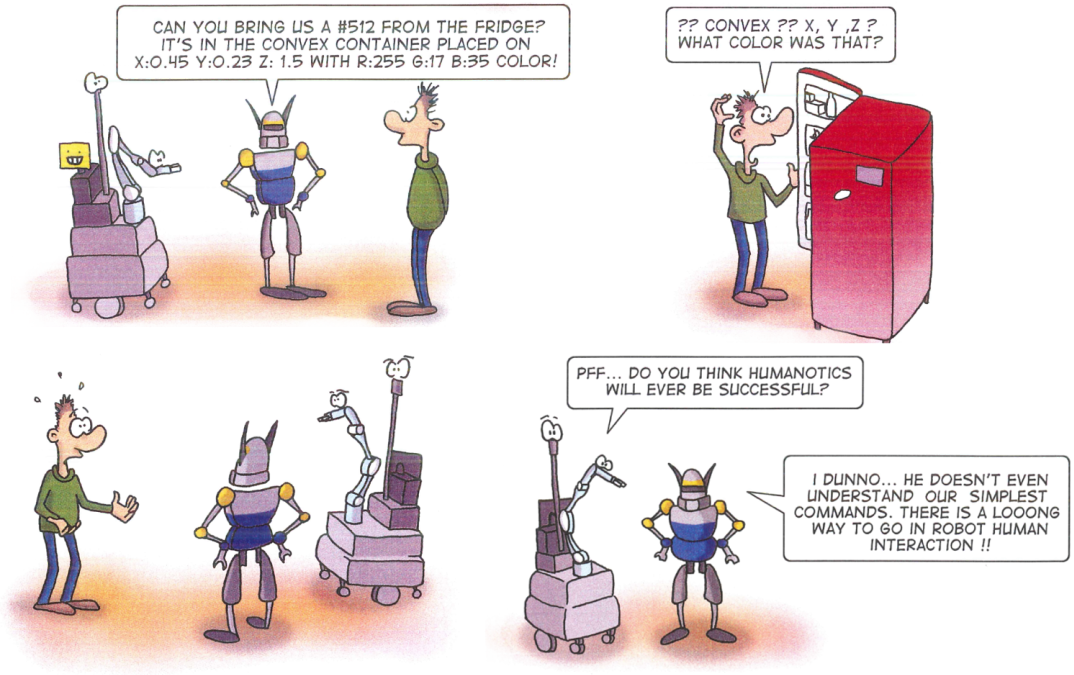


Figure 4.1.: Different representations of objects and spatial areas between humans and robots cause problems when it comes to Human-Robot Interaction. While many representations used by robots only relate to coordinates, humans rather use qualitative representations that include semantics. The differences between those representations can cause difficulties in scenarios where humans and robots work together as this cartoon illustrates in a humorous way. (Image courtesy of Mathieu Warnier and Mathias Fontmarty)

robotic tasks like mapping, localization, navigation etc., they lack the ability of equipping spatial areas with qualitative information about their context. Semantic information about the context of spatial regions becomes more and more important when robots are intended to communicate with persons since contextual information can provide a common base of information between robots and humans. Semantic environment maps, for example, extend two dimensional grid maps with knowledge about objects in the environment. They typically include the proportions of furniture objects as bounding boxes or even three dimensional models. Furthermore, they can be equipped with articulation models for opening different containers like doors, cupboards or drawers as shown by Pangercic et al. [65]. Some semantic maps offer support for object detections and allow the times and places of detected objects to be stored. It is even possible to link objects in semantic maps to a knowledge base that can provide common-sense knowledge about objects, thereby enabling a robot to reason about those objects [85]. Thus, queries like "Where do objects of daily use belong?", or "What is inside of a drawer?" can be answered using inference mechanisms of the associated knowledge base. However, most of the aforementioned approaches consider contextual information of specific objects, but do not account for information about persons that interact with those objects.

In this chapter, we enable a robot to be aware of spatial regions that are related to activities performed by persons. Therefor, we use *Context Dependent Spatial Regions (CDSRs)*, which are spatial regions that are not solely defined by their geometry but also by their context. Context in this sense can be seen as the functional use of the space by an agent, for example a bus stop, a safe zone in a battlefield [48] or a cooking place in a kitchen. Such regions are important for communication between humans and robots since they are related to the activities of persons and robots in the environment and in this work, we will specifically focus on CDSRs that are related to human task execution.

As humans think of places in terms of relative locations to entities in the environment [2], we can classify locations according to which activities or actions persons commonly perform close to specific objects. Considering, a robot detects a person performing an action at a location from which a cupboard is in reach. In this case, the performed action will likely be an action related to this cupboard

like it is the case when a person fetches an object from it. Generally, persons that pick up objects from that specific cupboard will most likely be located near that location. The addition of context to spatial areas, like the location from which humans are likely to perform pick- and place actions enables us to equip robots with a human-like representation of spatial regions. This way, a robot is able to express its knowledge in a human-readable way, facilitating communication between humans and robots. And as Kennedy et al. state, "... a system that uses representations and processes or algorithms similar to a person will be able to collaborate with a person better than a computational system that does not." [36].

4.1.1 Generation of Transferable CDSRs

A domestic robot helper might not be equipped with knowledge about CDSRs when it arrives in a new environment. Hence, it would be desirable if it could automatically learn CDSRs by observing motion tracking data of human task performance. Although, there are actions that some humans perform while moving, like grabbing an object while passing by the table, we assume that for most actions, like picking up objects from a cupboard, a person will stand still at a specific location for at least a short time. Based on this assumption, we can observe motion tracking data of human task execution, identify places where the observed person is executing an action and relate those places to furniture objects nearby.

4.1.1.1 CDSRs of Kitchen Activities

We decided to create exemplary CDSRs of typical activities that a person performs in his/her kitchen since we think that the kitchen will be one main application domain of domestic robot helpers. We therefore use motion tracking data of the TUM Kitchen Dataset that was described in section 6.1.2. Observing the two-dimensional position of the tracked person, we classify positions where the human is standing still and where interactions with objects occur. For the detection of object interactions, we use the ground truth labels of the hands and

account for pick- and place actions as well as the opening or closing of cupboards and drawers. One might as well think of using more elaborate approaches for the detection of object interactions like the analysis of full-body motion tracking data [7] or object detections based on sensor data [66]. Since this chapter is not focused on object perception, we utilize the labels provided by the dataset to limit the complexity of the experiments. Positions are extracted for the 10 impaired-person experiments of the TUM Kitchen Dataset where a human is setting the table for breakfast. The clustering of the extracted positions is performed based on the Expectation Maximization Algorithm [52] using WEKA [93] which returns the four clusters as pictured in Figure 4.2. We assume to know the number of clusters which correspond to the different storage locations of the objects that the human interacts with during task execution. In the TUM kitchen dataset, the objects used for the table setting task are stored on the oven, in the drawer and in one of the cupboards at the beginning of the table setting activity. All of the objects are located on the table when the table setting activity is finished. As we aim to equip robots with an impression about the likelihood of specific tasks being executed at certain positions, we fit a probability distribution to each cluster of points. The clusters in our example seemed to be almost equally distributed and also for reasons of simplicity, we decided to use two dimensional Gaussian distributions for the fitting. One could as well think about using more elaborate probability distributions like Stulp et al. in their work about Action Related Places (ARPlaces) [79].

These probability distributions thus represent CDSRs and equip the spatial regions with context about human activities. To build general models that are independent of specific environments, we model the CDSRs relative to the furniture objects that they relate to instead of relative to the origin of the global coordinate frame. This generalizes our model and allows the application of CDSRs not only to specific instances in our environment, but also to furniture objects that are similar. Once we know, where humans typically are located when picking up objects from a cupboard, we can, for instance, generate an assumption about where humans will be standing when picking up objects from another, similar cupboard. Therefore, we use the semantic map of our environment to locate the containers the human interacts with and put the means of the two dimensional gaussian distributions of our CDSRs into relation to them. We assume that the

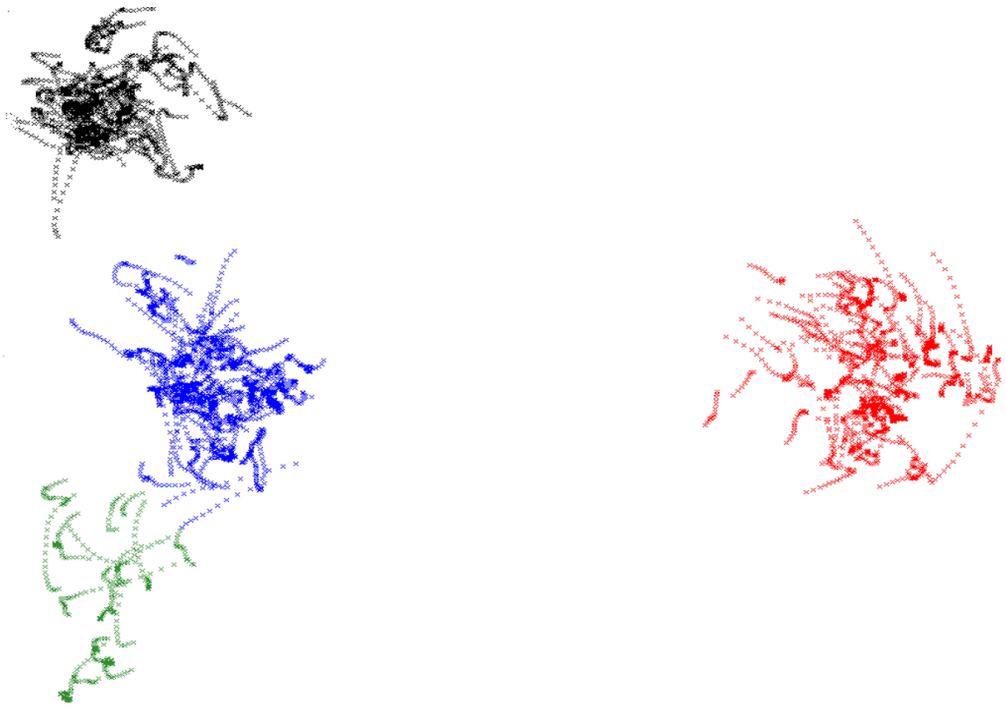


Figure 4.2.: Clustered 2D locations of 10 humans when standing still and interacting with objects during a table setting task in a kitchen. The different colors indicate the four different clusters returned from an Expectation Maximization based clustering.

position at which the human is standing while interacting with objects is dependent on the dimension and the opening direction of the corresponding container. When an object is picked up from a plane like a table, we assume the person's location being dependent on the location of the object as well as the closest edge of the plane. Given information about the type of the storage location of an object, we generate reference coordinate frames relative to these storage location using information from the semantic map, thereby distinguishing between objects that are stored in containers (e.g. a cupboard or drawer) and objects that are located on planes (e.g. table).

4.1.1.2 Objects in containers

For interactions with cupboards and drawers, we include the locations of their centroid, their depth and width into the calculation of the origin of our reference frame. For cupboards, we furthermore use information about the location of the door as well as the direction in which the door opens, depending on the location of the hinge. Such information is often annotated in the semantic maps, but can as well be extracted from the articulation models or the position of handles of the doors. We set the origin of the reference frame of a cupboard to be at the middle of the door. Thereby, the x-axis points to the direction from which objects can be fetched when the door is open in an angle of 90 degree to the closed door. The y-axis points along the door (when closed), away from the hinge as illustrated in Figure 4.3 on the left. This position can easily be calculated for each cupboard given information from the semantic map. The position of the hinge, respectively the direction to which the door opens, is important because it – among other factors – directly influences the position where a person is standing when picking up objects from a cupboard. For the case of drawers, we use the same position as origin for the reference frame, but do not account for the direction a hinge or similar (since there is none). In theory, we assume that the positions of the human relative to the drawer are distributed uniformly around one point that is offset from the middle of the drawer in the direction of the x-axis of our reference frame. In practice, however, we experienced slight deviations in direction of the y-axis. These deviations are not influenced by the drawer itself, but rather by other factors like the structure of the environment or the walking direction of the

person. In our current implementation, we see such deviations as neglectable. Further implications of such dependencies are discussed in section 8.2.

4.1.1.3 Objects on planes

In contrast to cupboards or drawers, planes do not have a general orientation that can be used to calculate the origin of a reference frame. The position where a person is standing while grabbing an object from a plane mostly depends on the location of the object and the structure of the environment. Specifically, it is influenced by other objects in the environment that limit the reachability of the object to grab. In the example of the TUM Kitchen Dataset, the placemat and the napkin are placed on the oven at the beginning of the table setting activity. In such cases, we take the position of the corresponding object into account and calculate the position on the edge of the supporting plane that corresponds to a projection of the position of the object along a line that has a 90 degree angle to the side of the supporting plane and points to the position from which objects are picked up or put down as illustrated in Figure 4.3 on the right. Unfortunately, the TUM kitchen dataset does not include detections of all of the objects that the human interacted with. Hence, we approximate the object locations using the full body motion tracking data as well as the labels of both hands. We assume consistent placing in all of the experiments and extract the positions of the hands when pick- and place actions occur. Thanks to the separate labeling of both hands in the experiments of the TUM Kitchen Dataset, we can directly approximate the object positions when one-handed pick- and place actions are performed by averaging over the hand positions of the 10 experiments. For object-interactions where the person used both hands, we assume the position of the object to be in the middle of a direct line between both hands and again average over the 10 experiments.

Having defined reference coordinate frames for the objects involved in the table-setting task, we can put the gaussian probability distributions, which we obtained from the clustering, in relation to the corresponding reference frame. This way, we obtain a set of locations l_j as tuple of a two-dimensional probability distribution P_i and a label $label_i$. l_j thus represents locations at which we ex-

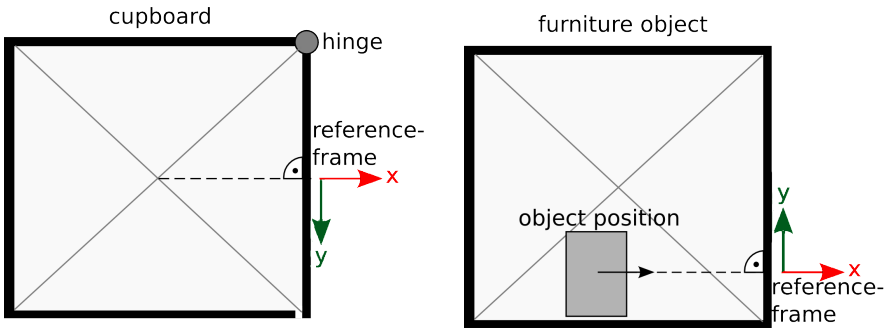


Figure 4.3.: The left figure shows how reference frames for human positions are calculated based on information about a cupboard from a semantic map. In a two-dimensional view from above, the origin of the reference frame is placed on the middle of the door of the cupboard and the x-axis is pointing towards the open side of the cupboard (when the door is open). The y-axis is pointing away from the hinge of the cupboard thus indicating the opening direction. The right picture illustrates the calculation of the reference frame for objects on planes. Here, the object position is taken into account and the origin of the reference frame is defined on the edge of the supporting plane, with the x-axis pointing to the position of the observed locations of the person.

pect pick- and place actions from the corresponding containers/planes $label_i$ to occur. We will call this set of locations the *spatial model* ψ:

$$\psi = \{l_1, l_2, ..., l_n\} \text{ with } l_i = (P_i, label_i)$$

This spatial model links two dimensional gaussians with context about human actions, in this case places where pick- and place actions occur are linked to corresponding furniture objects.

4.1.2 Transferring CDSRs

With the modeling of CDSRs relative to instances of furniture objects, we assume the models to be general enough to enable assumptions about locations related to pick- and place actions not only in the environment where we generated the models, but also in similar, but different environments. This can help a robot to

obtain a general understanding about how a table-setting task might look like in a new environment, even though, it has never observed one. To verify this hypothesis, we can transfer the CDSRs learned in one kitchen environment to another kitchen environment as we will demonstrate in section 7.2.1.2. Due to the relative description of the locations, this can easily be done be querying the semantic map of the IAS kitchen for the storage locations of the objects involved in the table setting plan (we assume those to be known) and query those containers for their type. Having obtained the storage locations, we can calculate the position of the CDSRs by generating the reference frame for each storage location. We calculate the location of the mean using the relative distances that we learned for the corresponding entity type and set the variance according to the type of container. This way, we can transfer CDSRs across different, but similar environments and thus generate a spatial model learned in one environment to another environment. This gives a robot an impression about how certain activities that were observed in one environment would look like in other environments, thus enabling it to detect those activities in environments where it never observed such an activity before.

4.2 Spatio-Temporal Plan Representations (STPRs)

So far, we described how we can enable a robot to generate a set of CDSRs that are related to activities in its environment by generating a spatial model. But the spatial model itself does not provide enough information to serve as model for different activities. For models of human activities, we want to include information about the durations that a human spends at certain locations as well as the patterns of locations that he/she visits during task execution. To achieve this goal, we use the spatial model to perform a segmentation of motion tracking data to automatically generate symbolic task descriptions that incorporate time and the patterns of visited locations. We will call such symbolic task descriptions *Spatio-Temporal Plan Representations (STPRs)*.

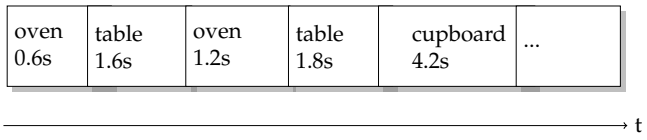


Figure 4.4.: Timeline-like visualization of an automatically generated, spatio-temporal plan representation of a table-setting task.

4.2.1 Concepts

We define a STPR $STPR_a$ of an activity a as a sequence of n tuples that consist of a location l_i and a duration t_i as elements.

$$STPR_a = \big((l_1, t_1), (l_2, t_2)..., (l_n, t_n)\big)$$

An example of a spatio-temporal plan description of a table setting task is shown in Figure 4.4. With STPRs, we use the activity related context of spatial regions to model activities not by the actions that are executed during the activity, but by the locations that are visited during the activity. The advantage of such a representation is that a robot does not have to perform action recognition, which is a complex task and is often not feasible for an autonomous robot due to a manifold of challenges like (self) occlusions of the human, limited sensor range, sensor noise, uncertainties, etc.

4.2.2 Generation of Spatio-Temporal Plan Descriptions

Once a spatial model is generated, a robot can use it for the segmentation of motion tracking data while observing human task execution and generate STPRs. To this end, the robot observes the motion tracking data and, as soon as it detected the human standing still for a short moment, the spatial model is queried to which CDSR the current position of the human most likely corresponds. Here, we consider the human "standing still" as soon as its center of mass is moving less than 25 cm withing 0.5 seconds (0.5 m/s) and a data preprocessing module is

responsible for providing positions at which this constraint is satisfied. To obtain the durations of the STPR, we measure the time which the human remains at the current location. As a result of this segmentation of the observed motion tracking data, we obtain an STPR as illustrated in a timeline-based representation in Figure 4.4. An overview of the generation of STPRs including the generation of a spatial model is illustrated in Figure 4.5.

We perform this segmentation using the data of the TUM kitchen dataset and evaluate the transferability of STPRs as well as their applicability for activity recognition in chapter 7.

4.2.3 Comparison of STPRs

STPRs can directly be compared and enable a robot to estimate the similarity between different observations of activities, which can be useful for a robot for simple activity recognition and the detection of habitual activities that are repeatedly executed in the same way. For the comparison of STPRs, we consider two different features: the durations that a human spends at certain locations and the patterns of locations that a human visits during task execution.

Durations

We assume the time a person is spending at certain locations to be a significant indication towards the activity that he/she is executing at this specific locations. Consider, for example, a robot observing a person standing at the sink. If the person is just putting down an object at the sink, he/she will most likely spend only a few seconds at this specific location. But if the person will, for instance, spend a few minutes at the sink, it is likely that he/she does not perform a pick-and place action there, but rather doing the dishes. We can exploit these dependencies between the actions and the durations and define a confidence value, that expresses how similar the durations of one activity are to the durations of

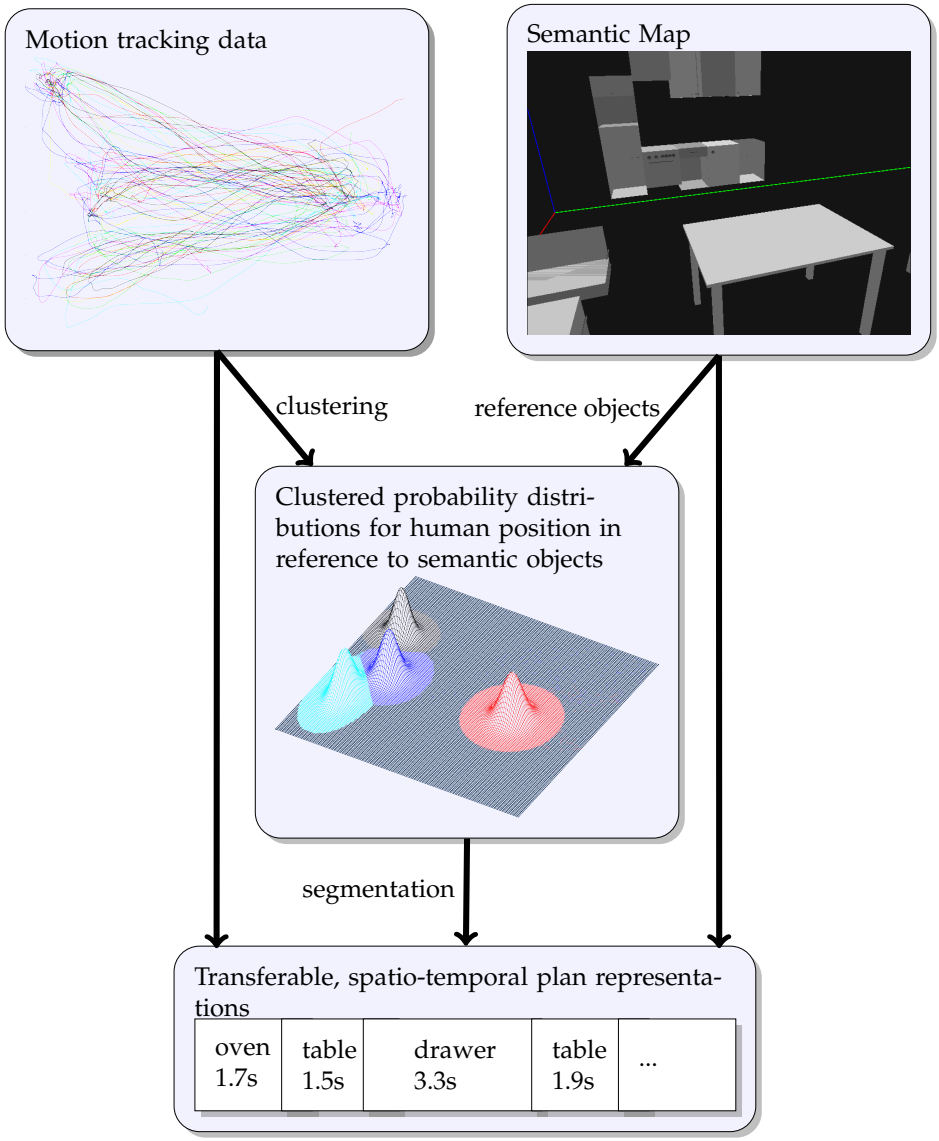


Figure 4.5.: An overview of the generation of general models about human locations during plan execution. Labeled motion tracking data is clustered for locations where the human interacts with objects. The clustered locations are then referenced to furniture objects obtained from a semantic map of the environment. This model is then used to do an automatic segmentation of the motion tracking data and create symbolic, time-line based models that are transferable to different environments.

another activity thus obtaining a measure of similarity between two activities. Therefor, we model the durations that a person generally spends at certain locations during the execution of one activity as gaussian distributions that are learned from observations:

$$\varphi(t_i)_{l_i} = \frac{1}{\sigma_{l_i}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{t_i - \mu_{l_i}}{\sigma_{l_i}}\right)^2\right).$$

Here, t_i corresponds to the duration that a person is observed to be standing at a location, $\varphi(t)_{l_i}$ represents the gaussian distribution assigned to location l_i, where μ_{l_i} and σ_{l_i} are the mean and the variance of the durations at location l_i of our reference-plan. To obtain a confidence measure c_p that describes how well the observed locations of the human fit to the model, we calculate the average confidence values over all n durations t_i of the observed locations l_i for an activity a_n as follows:

$$c_{a_n} = \frac{\sum_{i=0}^{n} \frac{\varphi(t_i)_{l_i}}{\varphi(\mu_{l_i})_{l_i}}}{n}.$$

Location patterns

As a second feature for calculating a similarity measure, we use the patterns of locations that a person visits during specific activities. We therefore assume that the same tasks are commonly executed in the same manner. This assumption is based on the work of Townsend et al. [87] who found that humans tend to pattern daily actions into sequences which they repeat at particular times in particular places. That means that the majority of activities of daily living (ADLs) are based on habits and thus are mostly carried out in the same way at the same locations.

For the comparison of plan patterns (without regarding for durations), we generate a string representation of the STPRs. We therefore use a unique acronym for each location involved in the STPR. Based on these acronyms, we can create a simplified string representation of an STPR that just considers the order of locations that a person visited during the execution of an activity. Such a string representation of the exemplary table setting activity in Figure 4.4 looks like the

following: "ADADCDBDBDBDCD". Given such a representation, we can use standard string comparison methods such as the *Generalized Levenshtein Similarity (GLS)* [95] to calculate similarity measures between two STPRs. The GLS is based on the Levenshtein distance between to strings which describes how many editing steps are needed to transform one string into another. A GLS value of 1 describes a perfect match of two strings, while a value of 0 expresses no correlation at all.

4.3 Related Work

For the generation of our CDSRs, we rely on semantic information about the robot's environment that is provided by a semantic map. Such semantic maps are of increasing interest to the robotic community and recent development in autonomous semantic mapping enables robots to gather such knowledge widely autonomously combining sensor information like laser scans, 3D point clouds and camera images. Depending on the intended application of a semantic map, the granularity between different approaches varies. Zender et al. [96], for instance focus on the detection of room categories in indoor environments according to their use and they create a hierarchy of layers that represent the environment at different levels of abstraction. They also include the possibility of using a linguistic framework to communicate with a person to include information from humans into the generation of the semantic map. Joho et al. [26] annotate environment models of supermarkets which include information about typical object arrangements to increase the efficiency of a mobile robot that is searching for those objects. Blodow et al. [9] classify different types of furniture objects and include them into a semantic map along with other information. They also learn articulation models for the opening and closing of container objects by the detection of handles of cupboards and drawers thus providing a robot with information about how to open such containers. Tenorth et al. [85] link semantic maps to a common sense knowledge base and create knowledge-linked semantic object maps thus enabling a robot to reason about objects in its environment. Such common sense knowledge can, for instance, be obtained from existing databases and converted to ontological concepts that a robot can use for inference as Kunze et

al. show [42]. This way, a robot is able to infer likely locations of different types of objects without having yet explicitly detected an object.

Although, first approaches in semantic mapping aim towards the generation of semantically enriched environment models, a majority of environmental representations in robotics purely relies on geometric information like coordinates, bounding boxes, etc.. Humans tend to classify spatial regions according to their functional use and a robot that operates in an environment populated by humans should be able to understand its surroundings in terms of human spatial concepts [96]. To achieve such a human-like understanding, Liao et al. [46] use hierarchical conditional random fields to analyze human behavior from GPS traces and learn reoccurring patterns. They recognize significant places a person visits during activities of daily live and generate labels according to their function (home, working place, home of a friend, ...). They show, how such models can be transferred to different persons and use the models to estimate the intentions of the a person. A similar approach to the generation of our CDSRs is proposed by Stulp et al. [79]. They define *ARPlaces* that equip spatial regions with information about the utility of positions in the context of mobile manipulation. Therefore, they create probability distributions that model the probability for a successful grasp of a mobile robot in reference to the pose of objects in the environment. In contrast to our approach, they consider the position of objects only and do not account for furniture or containers. Klenk et al. [37] find that "the ability to understand and reason about spatial regions is essential for cognitive systems performing tasks for humans in everyday environments". They define CDSRs that are learned from qualitative spatial representations and semantic labels and use them to identify similar environments. They also show how to transfer the CDSRs across similar but different environments.

In this chapter, we introduced models of human activities that are defined by a sequence of location at which a person is standing still for a short time. A similar concept is used in the work of Bennewitz et al. [8]. They state, that when performing everyday activities, people are not in permanent movement but rather move between *resting places* at which they are standing still. They state that mobile robots that are to work together with persons can benefit from knowledge about the locations of people in their environment and use motion patterns to

actively maintain a belief about human positions and anticipate their possible intentions. In contrast to our approach, their focus lies on trajectories between the *resting places* and they use Hidden Markov Models to estimate and anticipate positions of a person. Also Tipaldi et al. [14, 86] use models of human activities for path planning of a mobile robot. They model human activities in a probabilistic spatio-temporal gridmap using a spatial Poisson process. Luber et al. [49] find that people "... typically move and act under environmental constraints." and describe how to learn the spatial affordance map that is used in [14, 86] . In contrast to our approach, the spatial affordance map they use only includes information about the locations of persons at different points in time without providing a link to a knowledge base. Orkin et al. [63] use spatial regions to model nominal behavior of persons while playing a computer game simulating a restaurant. They learn models of nominal behavior and generate and validate expectations trying to detect exceptional situations of the players. Beetz et al. [7] propose the automated generation of probabilistic models of everyday activities that describe human activities at various levels of abstraction to enable robots to analyze actions in their situational and activity context. Therefor, they use several learning approaches to classify sequences of activities and combine symbolic knowledge with sequences of motion tracking data.

Chapter 5

Activity Recognition

Effective and socially acceptable robots should take into account human activities and adapt their behavior accordingly. We think that this capability plays a key role for future domestic helpers that are intended to perform complex tasks in unstructured environments, such as a human home, and is essential to enable commercial success of such robots. Consider a household service robot that has the duty to clean the table after a person had breakfast. Such a robot should be aware that the person already had his/her meal and clean the table afterwards. A robot that is not equipped with such a capability might clean the table before the human actually ate. Such situations could, for instance, happen when the person leaves the kitchen to answer the door after setting the table. In addition, activity recognition plays an important role for the detection of errors and anomalies. A person sleeping in the morning or late evening can be classified as normal, but a person lying down in the middle of a cooperative task might be a sign that the person is not feeling well (especially when thinking about the application of robot helpers in the domain of elderly care). Thus, the ability to perform activity recognition and distinguish different human behaviors is essential if household assistants should become useful and comfortable assistants in human environments.

While there already exists a variety of different approaches to equip robots with activity recognition, the environment has often to be equipped with loads of sensors like RFID tags and readers or extremely expensive motion tracking systems. Such setups offer precise data, but might not be available in typical human homes which are the intended operation areas of our domestic service robots. Future domestic service robots might have only inexpensive sensors mounted on

Figure 5.1.: The image shows a PR2 robot that is performing Activity Recognition in a kitchen environment using sensor data from a Kinect.

their bodies to reduce their costs and free their users from the burden of equipping their home with sensors. In this chapter, we investigate an approach for activity recognition that uses one Kinect sensor for motion tracking and allows for reliable activity recognition in spatially limited environments as illustrated in figure 5.1. Here, a PR2 robot observes a table setting activity of a person using a Kinect sensor and estimates a percentage of 95 % of having observed a table setting task.

5.1 Challenges in Activity Recognition

Given, a robot is equipped with models of human activities as we generated in chapter 4, the recognition of human behavior is still far from trivial. Approaches in activity recognition have to deal with a number of challenges, some of which are explained in the following.

- **Activities executed in parallel:** When performing tasks, humans tend to execute activities in parallel. A person might, for example, clean parts of the kitchenette while cooking or drink a glass of water in the middle of a

cleaning task. This poses challenges for the generation of models of human activities, but also when recognizing activities from sensor data since the separation of different activities is often not observable or even nonexistent.

- **High variability in the way similar/same actions can be performed:** While many activities of daily living are performed in a similar way, humans sometimes vary subtasks of activities depending on opportunities or without further reasons. A person that is setting a table might, for example, sometimes get the cutlery before a cup, but sometimes he/she might do it the other way round.

- **Cheap, non intrusive sensor setting:** Future domestic robots will presumably use rather low-cost sensors in order to be affordable for a wide range of people. Therefore, we think that they will have to rely on sensors that are subject to a certain amount of noise and low accuracy. This poses challenges for activity recognition since a system cannot rely on accurate data from its motion tracking or object detection module. The sensor setup should be as unobtrusive as possible to avoid making the human user feel uncomfortable and also keep the effort of equipping the environment with sensors as low as possible.

- **Occlusions:** Especially in confined environments, vision-based sensors have to deal with occlusions when tracking humans. Occlusions might, for example, occur when a human is standing behind doors while opening containers like the fridge or when sitting at the table. In the case of opening containers like a refrigerator, the upper part of the body is often occluded by the door of the fridge (depending on the position of the sensor) while in the case of sitting at a table, the lower parts of the body are likely to be occluded. There are several ways of dealing with occlusions depending on the environment and the sensor setting. Some approaches merge data from several cameras [82], others use model-based approaches to include occluded body parts into a model in the tracking algorithm [12].

In the remainder of this chapter, we will introduce an activity recognition approach that relies on Hierarchical Hidden Markov Models (HHMMs) and the Forward-Backward algorithm (see section 2.2.2) to deal with the aforementioned

challenges. The approach will be evaluated in chapter 7 and advantages and limitations of the approach will be discussed in chapter 8.

5.2 Hierarchical Hidden Markov Models for Activity Recognition

With the introduction of Spatio Temporal Plan Representations (STPRs) in chapter 4, we introduced a way to enable a robot to autonomously generate models of human activities. These models can be compared based on several features which can be seen as a simple variant of activity recognition. They can even be transferred across similar environments given a semantic description of its structure . Although, STPRs are a good means of modeling general human behavior that is repeatedly executed in the same way while providing a simple model for activity monitoring, they lack the ability to represent partial variations of subactivities. A person setting a table for breakfast might, for instance, first get the plate and then the cup on one day, while at another day, he/she might do it the other way around. Moreover, the order of the activities themselves might be subject to variations. While one day, a person might set the table for cereals after having a shower, he/she might set the table for bread on another day, or even leave the house without having breakfast (in case he/she is late for work). Since we model STPRs as sequences of Context Dependent Spatial Regions (CDSRs) that are stored in a spatial model, STPRs lack the ability of modeling uncertainties in the observations. The performance of activity recognition based on STPRs and a spatial model as explained in section 4.2.3 heavily relies on the correct assignment of detected locations of the human to a CDSR in the spatial model. In confined places it is often hard to distinguish between CDSRs in the spatial model since CDSRs that are close to each other might overlap. This is problematic since we base our activity recognition on CDSRs using the same spatial model that was used for the generation of STPRs. Advantages of using CDSRs instead of the detection of actions and/or objects are the simplicity of the approach as well as the inexpensive and non-intrusive sensor setting that is required. We will not need to perform complex action recognition or equip the environment with loads of sensors. Instead, we solely rely on motion tracking data of one Kinect sensor, a spatial model of the environment and human activity models as

HHMMs to enable a robot to get an impression about the activities that a person performs.

5.2.1 Hierarchical Hidden Markov Models

To represent human activities using context dependent spatial regions, we can train an Hidden Markov Model (HMM) from the sequences of observed location detections based on the spatial model of the environment. The spatial model can easily be generated from a semantically annotated map of the environment and learned relative locations of a person towards container objects in the environment as shown in section 4.1. The representation of activities using a HMM has the advantage over STPRs that, as a probabilistic model, it can account for uncertainties in the observations as well as uncertainties in state transitions. Due to the hierarchical nature of activities, we decided to use Hierarchical Hidden Markov Models (HHMM) for our activity recognition approach. A HHMM introduces the concept of hierarchies to HMMs by allowing each state of the HMM to be an HMM itself as indicated in Figure 5.2. This allows the modeling of model multi-level stochastic processes like activities that are defined by sequences of CDSRs. HHMMs differentiate between two types of states: *internal states* and *production states*. The *internal states* are hidden states that are HMMs themselves and do not emit single observation symbols but rather sequences of observations by recursive activation of one of their sub-states. In our model of activities, internal states represent single activities. States that actually emit output symbols and are located at the lowest hierarchical level are called the *production states* and correspond to CDSRs in the HHMM. The activation of a sub-state by its internal state is called a *vertical transition* in our HHMM. This corresponds to a person starting to perform a new activity. A transition between two production states is referred to as *horizontal transition* and happens when a person goes from one CDSR to another. Every sequence of production states has exactly one terminal state which, when reached, ends the process of recursive state activation and leads to a vertical transition upwards in the hierarchy. In our case, this happens when one activity is detected to be finished and the observed person starts performing another activity.

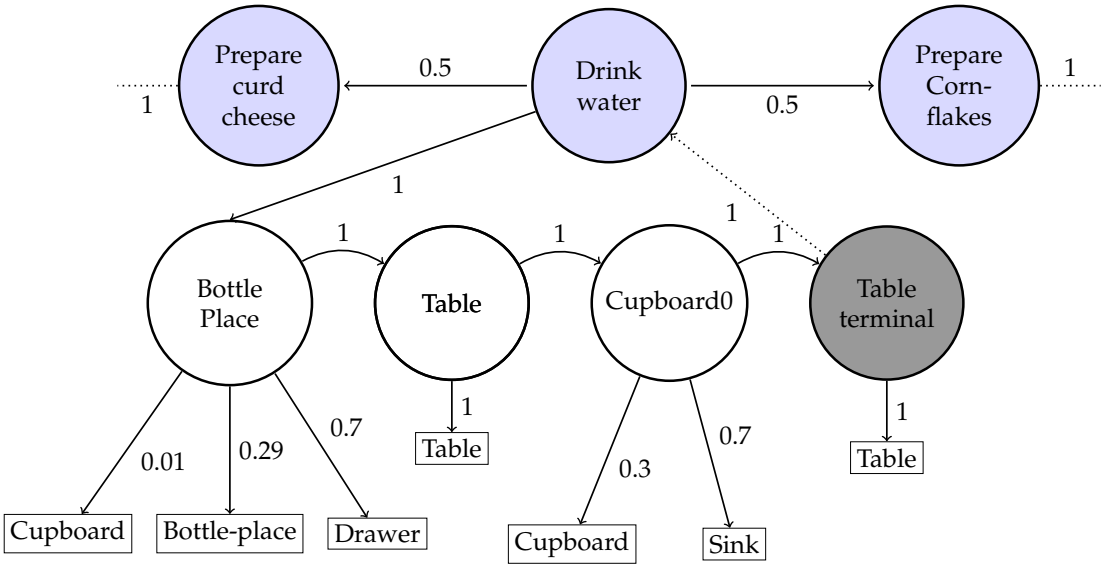


Figure 5.2.: An exemplary hierarchical HMM for activity recognition generated from a STPR. Light gray nodes represent *internal states* (activities) which are HMMs themselves. For visualization, only the "Drink Water" activity is illustrated in detail. White nodes are the *production states* which correspond to CDSRs where the observed human is standing. A dark gray node is a *terminal (production) state* which, when reached, leads to a vertical transition upwards the hierarchy of the HHMM (indicated by the dotted arrow). Rectangles represent observations that are expected at the production states. Transitions between states are labeled with their corresponding (exemplary) probabilities.

In a more formal description, a HHMM can be described as a three-tuple consisting of a topological structure ζ, an observation model Y and a set of parameters θ. The topology defines the number of levels $D = \{d_1, ..., d_n\}$ of the hierarchy, parent-child relationships between levels and the state space at each level. The observation model Y describes the set of possible observations $Y = \{y_1, ..., y_m\}$. Given ζ and Y, the set of parameters of the HHMM θ is defined by

$$\theta = \{B_{y|p}, \pi^{d,p^*}, A_{i,j}^{d,p^*}, A_{i,end}^{d,p^*} \mid \forall \left(y, p, d, p^*, i, j\right)\} \tag{5.1}$$

where $B_{y|p}$ describes the probability of an observation $y \in Y$ while being in production state p. π^{d,p^*} represents the initial distribution over all children of the internal states p^*. The transition probabilities between child nodes $i, j \in child\left(p^*\right)$ are described by $A_{i,j}^{d,p^*}$ and the probability of an internal state terminating, given its child, is the production state i is $A_{i,end}^{d,p^*}$.

5.2.2 Generation of HHMMs

To generate such a HHMM, we need to calculate values for the variables in equation 5.1 and define the topological structure ζ and the observation model Y. Since we want to use sequences of CDSRs to model human activities, we define the topological structure ζ manually in a two-level HHMM consisting of high-level activities and CDSRs. The higher level defines sequences of activities that a person executes. All states in this level are internal states p^* and do not emit single observations. The lower level consists of CDSRs that are production states p and emit observations y. The set of possible observations Y includes all CDSRs that are defined in the spatial model. For the parameters θ, we assume the initial distribution of state probabilities π^{d,p^*} and $A_{i,j}^{d,p^*}$ to be distributed uniformly at the beginning. One could as well think of biasing this initial distribution using prior knowledge like activities that are more likely to be executed before others (a person will, for instance, most likely first eat and then clean the table), but using an initial uniform state distribution is more general. The transition probabilities $A_{i,j}^{d,p^*}$ and $A_{i,end}^{d,p^*}$ can be learned by observations using the Baum-Welch Algorithm as explained in section 2.2.3.

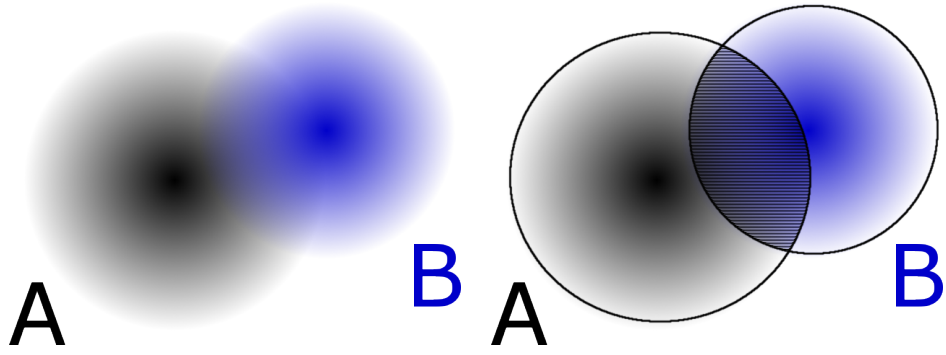


Figure 5.3.: Left: Two exemplary CDSRs that overlap. Right: The regions of CDSRs A and B consist of non-overlapping areas as well as overlapping areas. The approximate proportions of these areas can be used for the estimation of emission probabilities in the HHMM.

The emission probabilities $B_{y|p}$ can be learned from observations as well. Assuming that those probabilities are proportional to the degree of overlapping of the corresponding gaussians in the spatial model, it is also possible to estimate them from the spatial model. The advantage of the latter method is that we do not have to run separate (time-consuming) experiments for the estimation of the emission probabilities, but instead estimate them once from the spatial model. This is helpful in cases where large amounts of training data are unavailable. Two exemplary, overlapping CDSRs "A" and "B" are illustrated in Figure 5.3. Here, we assume the CDSRs to be modeled as two dimensional gaussian probability distributions and all CDSRs to be maintained in the spatial model. In the example of Figure 5.3, CDSR "A" consists of an area that is not overlapping with any other CDSR and an overlapping area which is marked hatched on the right image. But the exemplary borders that are illustrated in figure 5.3 do not really exist in continuous gaussian probability distributions. We use a variant of a sampling procedure for the estimation of the sensor model of the HHMM. To obtain the emission probability $P(y \mid p)$ for each production state p and each emission y, we repeat the following procedure for each CDSR in the spatial model:

- Randomly draw two dimensional sample points from the area of the gaussian that defines the CDSR. This is done in a way such that the probability of drawing each sample point is proportional to the probability of the point in the gaussian it is assigned to.

- For each sample point, we query the spatial model for the most likely CDSR and save the result. In the example in figure 5.3, this would return the CDSR "A" for each point that is in the non-overlapping area. If, in the first step, we drew a sample point that is located in an overlapping area of the CDSRs, the query will return the CDSR with the highest probability according to their corresponding gaussians. In the example of figure 5.3 with only two overlapping CDSRs, the query would either return "A" or "B".
- Both steps are repeated for a fixed, large number of times (around 5000 - 10.000 in our experiments seemed to suffice) and the results are kept in a table.

From the results for each CDSR, we generate a statistical estimation for $P(y|p)$ for each production state p, thus obtaining an estimate of the observation model $B_{y|p}$.

5.2.3 Activity Recognition using Forward-Backward Algorithm

Once we generated an HHMM in the aforementioned way and have a spatial model of the environment available, we can perform activity recognition using the Forward-Backward Algorithm (see section 2.2.2) as illustrated by the pseudo code in Algorithm 3. We observe a stream of motion tracking data we call *motion_stream* that we are provided by a Kinect sensor and a motion tracking software framework. Specifically, we obtain a stream of x,y and z coordinates as well as the three dimensional orientation as a quaternion for each joint. The stream is constantly monitored for situations where the tracked person is likely to stand still and execute an action. As for the generation of the CDSRs (chapter 4), we assume a person to be standing still when his/her center of mass is moving less than 25 cm withing 0.5 seconds (0.5 m/s). As we found in experiments, solely using such a detection is prone to errors, especially when relying on noisy motion tracking data, such as data from Kinect-based tracking systems. To improve recognition results, we use simple motion patterns that we assume to be characteristic for typical pick- and place actions of a person. Therefore, we assume a person to be standing at a CDSR related to a pick- and place action when

he/she is first moving towards a furniture object, staying there for a little while (meaning at least one detection of standing still) and then moves away from the furniture object. If those conditions are satisfied, we assume to have detected a person standing at a CDSR and we query the spatial model to which CDSR the coordinates of the detected two dimensional location correspond to (line 3, Algorithm 3). The query will return the CDSR that most likely corresponds to the detected location and we add this CDSR to the observations of the HHMM (line 4, Algorithm 3).

Algorithm 3 Activity Recognition using Forward-Backward Algorithm

Require: $hhmm, motion\_stream$
1: **while** $motion\_stream$ **do**
2: **if** $isStandingStill(motion\_stream) == TRUE$ **then**
3: $cdsr = querySpatialModel(motion\_stream.x, motion\_stream.y)$
4: $hmm.addObervation(cdsr)$
5: $P(p_i^*, p_i \mid y_{1:t}) = hmm.forwardBackward()$
6: $P(p_i^* \mid y_{1:t}) = getInternalStateProbabilities(hhmm)$
7: **end if**
8: **end while**

Each time, an observation is added, the Forward-Backward algorithm is used to calculate the posterior marginals $P(p_i^*, p \mid y_{1:t})$ of all states of the HHMM (line 5, Algorithm 3). Since for our activity recognition, we are interested only in the probabilities of the activities, we use a probability distribution over the internal states of the HHMM $P(p_i^* \mid y_{1:t})$ (line 5, Algorithm 3).

One should note that any HHMM can be decomposed into a standard, non-hierarchical HMM using the "flattening" method [91] which makes all algorithms that work on HMM applicable to HHMMs. There are also algorithms that exploit the hierarchical structure of HHMMs for performance improvements [19]. We use a variant of the Forward-Backward Algorithm as described by Rabiner et al. [73] on the flattened HHMM, thus obtaining the posterior marginals for all states. If we would only want to account for the most likely explanation of the observations, we could as well use the Viterbi Algorithm to find the most probable state sequence of an (H)HMM. For a more detailed formal description of HHMMs, which lies beyond the scope of this thesis, we refer the interested reader to the work of Fine et al .[19] and Bui et al. [11].

5.3 Simultaneous Plan Recognition and Monitoring (SPRAM)

Hidden Markov Models, being a special case of Dynamical Bayesian Networks, make use of the Markov assumption (see appendix A.2) to improve computational and mathematical tractability. This means they assume every state to only be dependent on a fixed number of predecessors. In our application we assume a first order Markov process and the states of the HHMM representing CDSRs that a person visits. This means that we assume the locations that a person visits to only be dependent on the one location he visited before. While this assumption will hold for activities in which each location is visited only once, it will not be true for activities where locations are visited several times. In a typical table-setting task, for instance, a person will visit several locations like the table or the fridge more than once, so the probabilities for some locations will also depend on how often specific locations have already been visited. We propose that it is beneficial for a robot to keep track of the current state of task execution, in particular, by counting how often specific locations are expected to be visited by a person and how often the person has actually been at this location. This information can be used to predict possible future locations on the one hand and on the other hand obtain an impression about the progress of the current activity. Sung et al. state, activity recognition [80] mostly is not 100 % certain about the activity the human is executing. They found that uncertainties are highest among very similar activities, even when using sensor data without occlusions. As experiments will show in chapter 7, we can confirm these findings for our application. To enable predictions based on uncertain results from activity recognition, we propose a simple, heuristic monitoring technique. It keeps track of the progress of activities even without exactly knowing which activity the observed person is currently executing. The monitoring runs in parallel to activity recognition and receives updates when a new observation is added to the HHMM of activity recognition. We will call the process Simultaneous Plan Recognition and Monitoring (SPRAM).

5.3.1 Monitoring of the activity progress

We decided to monitor activities that have high probabilities and are thus likely to be executed. In all of our experiments, the number of activities was limited to eight activities at the most, thus it was even tractable to monitor all modeled activities in real time. Given STPRs as models of human activities together with a stream of observed CDSRs and probabilities for each activity, we propose a simple heuristic monitoring technique that keeps track of the state of execution of each activity. It monitors if a location included in an activity has already been visited the expected number of times or if it is expected to be visited by a person in near future. Therefor, a *locations cache* is maintained for each activity, which keeps track of a list of locations that have been visited while the probability for the specific activity was high. When the probability for an activity falls below a certain threshold, i.e. the activity is unlikely to be executed, the locations cache of this activity is reset to an empty list. Using such a simple routine, we can distinguish between observations that are likely to belong to a specific activity and locations that do not. This property is important for our system, since, for instance, we do not want to mark the location *refrigerator* as visited in a *prepare cereals* activity when we are quite sure that the person is currently not preparing cereals. In such a situation, we would rather want to keep all locations of, for instance, the *prepare cereals* activity marked as not yet visited and keep them as possible next locations (most likely with a low weight). For each activity, we compare the locations in the locations-cache $l_i^{loc-cache}$ with the locations of the STPR $l_i^{a_i}$ of the activity a_i and simply check if each location has already been visited the expected number of times while the probability for the activity was high according to the activity recognition. This way, we generate probabilities $P_{mon}^{next}\left(l_i|a\right)$ about the location we expect a person to visit during the execution of the current activity and which locations we do not expect to be visited any more. We define the probability $p_{mon}^{next}\left(l_i|a\right)$ for a location l_i to be 1 if the number of visits according to the locations cache $l_i^{loc-cache}$ is smaller than the expected number of

times. If a location has already been visited the expected number of times, we set the probability to 0 as illustrated by the following equation:

$$p_{mon}^{next}\left(l_i|a\right) = \begin{cases} 1 & \text{if } \left|l_i^{a_i}\right| - \left|l_i^{loc-cache}\right| > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Note: This extremely simple way of monitoring is used as proof-of-concept. More elaborate approaches for monitoring are discussed in section 8.3.

5.3.2 Prediction of Likely Next Locations

One of the reasons for using activity recognition and monitoring simultaneously is the prediction of likely future locations of a person based on models of its behavior. This can be useful for a domestic robot to adapt its behavior better to a person's habits and behavior. A robot that can predict where a person will go, is able to avoid getting into a person's way when it should not disturb him/her. In case of an emergency, a robot could as well use its predictions to find the person. To enable such predictions, we can use the Forward Step of the Forward-Backward algorithm on the HHMM and calculate the probability distribution $P_{FW}^{next}\left(l_i^{t+1}|a_j^{t+1}\right)$ over the production states (the locations) l_i^{t+1}, which represents the likelihood of the locations to be visited in the next step $t+1$ conditioned on the activity a_j^{t+1}. This calculation does not produce any more additional computational effort since it is already done in the Forward-Backward algorithm that we use for activity recognition. We can use the probabilities of the monitoring step $P_{mon}^{next}\left(l_i|a\right)$ to weight plan-dependent locations in $P_{FW}^{next}\left(l_i^{t+1}|a_j^{t+1}\right)$ based on the binary probability distribution $P_{mon}^{next}\left(l_i|a\right)$ from the monitoring. This assigns CDSRs that are not expected to be visited again a lower weight which results in a merged probability distribution over plan-dependent locations that we expect to be visited next: $P_{merged}^{next}\left(l_i^{t+1}|a_j^{t+1}\right)$. For the prediction of likely next locations, we are interested in the plan-independent likely next locations of a person. If we, for instance, want to know if the observed person will got to the table next, we do not care if he/she will go to the table because he/she is setting the table or cleaning the table, especially when the activity recognition returns ambiguous results. To enable the best possible prediction given the uncertain information collected

so far, we marginalize out a_j^{t+1} from $P_{merged}^{next}\left(l_i^{t+1}|a_j^{t+1}\right)$ and obtain a probability distribution over all locations to be visited in the next step taking into account the results from activity recognition and monitoring:

$$P(l_i^{t+1}) = \int_{a_i} P\left(l_i^{t+1}|a_j^{t+1}\right) * P\left(a_j^{t+1}\right) da. \tag{5.3}$$

As we represent locations l_i by two dimensional gaussian probability distributions, we can see $P(l_i^{t+1})$ as a spatial model comprised of a set of weighted, two dimensional gaussians with their corresponding probabilities as weights. This weighted spatial model represents locations that are likely to be visited by the observed person in the next step and it is updated with every belief-update of activity recognition. This simple approach enables a robot to have an impression about where the human is likely to be in near future or which locations are unlikely to be visited by him/her. The inclusion of several activities for the prediction has the advantage that even in cases where activity recognition is uncertain, a robot can still generate a weighted spatial model with knowledge about spatial areas that are likely to be visited in near future as illustrated in Figure 5.4. In the left picture, the robot is quite certain that the observed person will go to one gaussian, which corresponds to the table in this example. In the right picture, the robot is uncertain about where the person will go next, but nevertheless, it is able to create a weighted spatial model indicating several regions where the person is likely to be next. In the case of a household robot, a robot could try to avoid regions with a high likelihood of human presence in order to not disturb its user or search those locations if it is trying to find a person.

5.4 Integration of SPRAM into the Expectations Framework

Our approach of simultaneously performing activity recognition and monitoring allows the prediction of likely future locations of a person performing an activity based on models of his/her routine behavior. This means that the output of the SPRAM approach can be seen as expectations that model a typical or *normal* behavior of a person. Thus, the approach is a perfect candidate to be included into the expectations framework described in chapter 3. The predictions of a person,

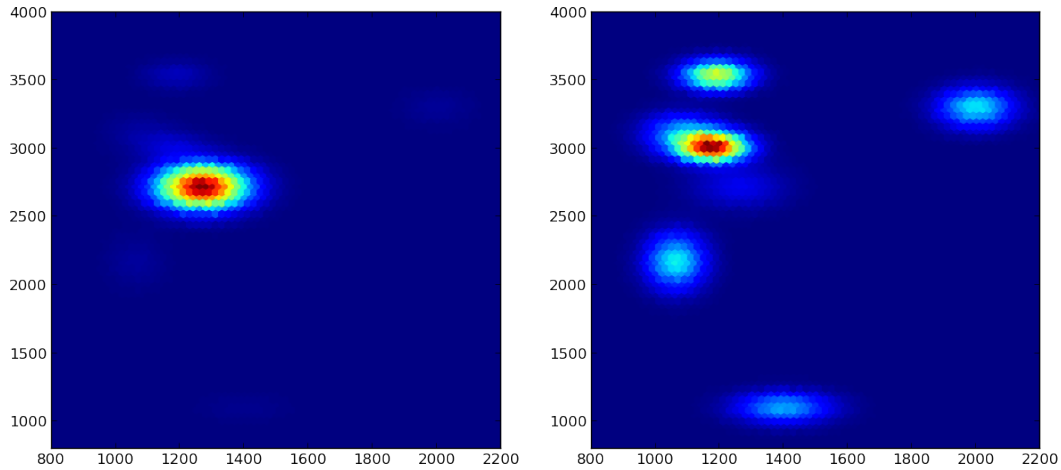


Figure 5.4.: This weighted spatial model displays probabilities about where the human is expected to go next. In the left picture, one location has a very high probability of being the next one while on the right picture, the predictions are not unique. Nevertheless, the spatial model shows likely next locations with corresponding probabilities.

for instance, about his/her likely next locations, are continually updated and depend on the complete integration of the HHMM-based activity recognition and monitoring, including the spatial model of the environment. To integrate this complete pipeline, we propose a three-layered architecture as illustrated in Figure 5.5. Here, the *Data Preprocessing Layer* directly works on the sensor data of the motion tracker to detect situations in which the tracked person is standing still for a short time and is likely to execute an action. This *Static Location Detection* therefore also includes the motion patterns described in section 5.2.3 and has a semantic map available which is queried for the coordinates of detected locations of the person. These coordinates are passed to the *SPRAM Layer* which queries its spatial model for the most likely qualitative description of the coordinates. Given models of human activities as HHMMs, the *SPRAM Layer* performs simultaneous activity recognition and monitoring as explained in the previous sections and returns a set of weighted, two-dimensional gaussian probability distributions. Those represent likely next locations of the observed person. We can map these expectations almost directly into probabilistic expectations that were described in section 3.1.1. We create a new expectation called *Next Location Expectation* and use the qualitative label of each gaussian probability distribution in the spatial model as the name of the random variable of the expectation and

assign the corresponding probability returned from the *SPRAM Layer*. Performing this procedure for each gaussian in the spatial model, we obtain a discrete probability distribution over the possible next locations of the person embedded as a probabilistic expectation in the expectations framework. The validation of this expectation is triggered each time a new observation is added to the HHMM. This integration enables a robot to distinguish typical behavior of a person from atypical behavior and to combine this information with other expectations in the expectations framework, thus including human behavior into its situational awareness. In chapter 7, we will test the fully integrated system and depict use cases in a real-world scenario as well as a simulated scenario.

5.5 Related Work

The research field of human activity recognition deals with the question of how to recognize human behavior from sensor readings using a variety of different sensors and approaches. In general, most activity recognition systems use probabilistic approaches to calculate a probability distribution over a discrete set of activities. Activities are therefore often modeled as states in discrete Markov Chains.

To enable computationally expensive probabilistic approaches to be tractable, many approaches are based on the Markov Assumption (see appendix A.2) which assumes states in a Markov Chain to be dependent only on a fixed number of predecessor states. Most commonly, a first-order Markov process is used, thus the current state is only dependent on one predecessor state. Depending on the models of states, this assumption might not hold in some cases but often these approaches produce good accuracy of the recognition while still being computationally tractable in a certain time, often even in real time. In this section, will provide an overview about different approaches and applications of activity recognition approaches that have been successfully used so far.

Many approaches in the field of activity recognition make use of HMMs to distinguish between various high level tasks. Buettner et al. [10] equip a large set of everyday objects in an apartment with RFID-based sensors and record sequences of object detections while a person performs typical everyday activities. They use

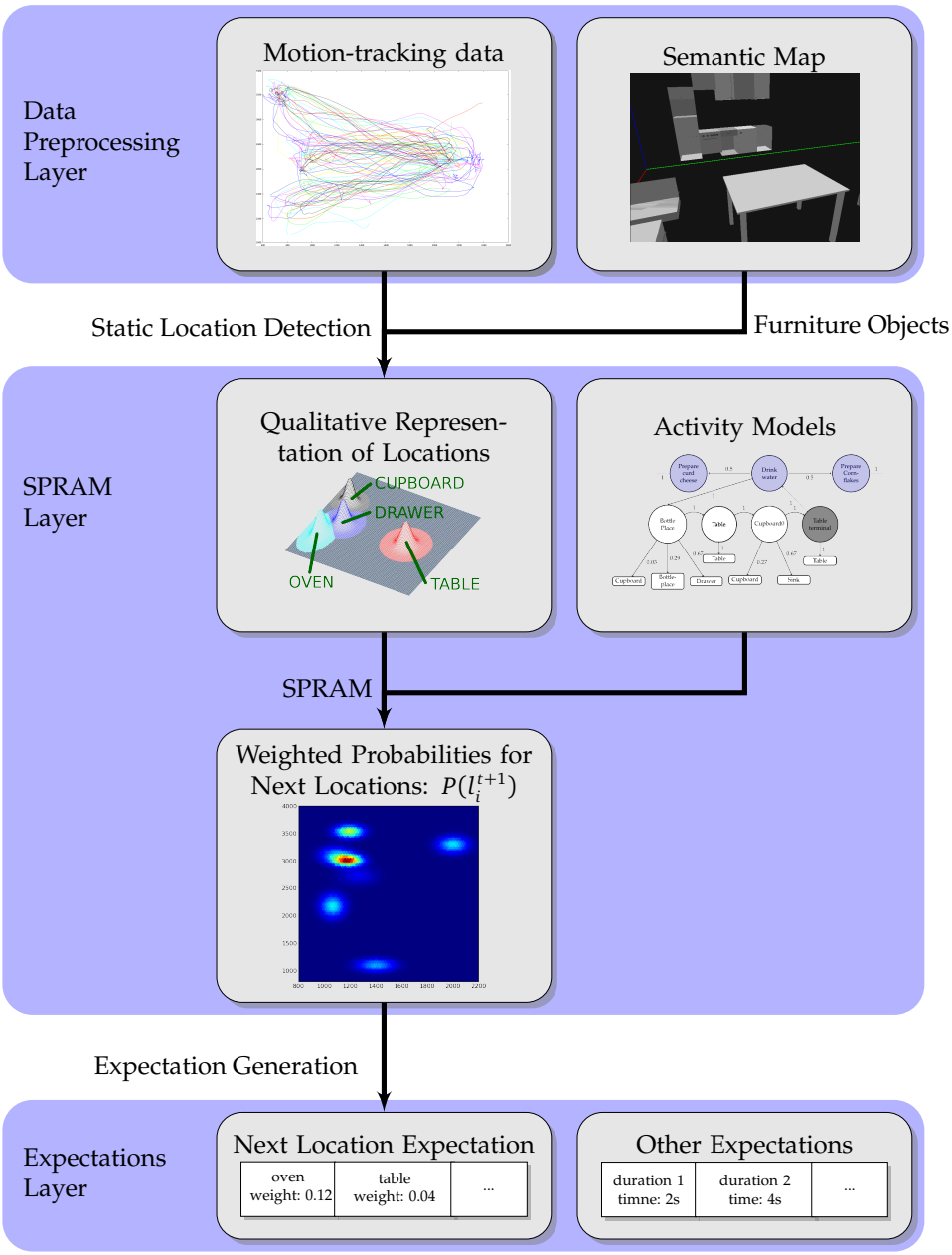


Figure 5.5.: The integration framework for human activities consists out of three layers. In the first layer, sensor data from motion tracking is preprocessed to fit the interface of the SPRAM module. Therefore, locations where the human is standing still as well as a semantic environment map is made available to the SPRAM module. The SPRAM module uses this information and its models of activities to generate weighted probabilities of likely next locations of the observed person. Those weighted probabilities are then transformed into expectations about the human and can seamlessly be integrated into the expectations framework along with other types of expectations.

these sequences of detected objects to train an HMM which they can then use for activity recognition. Nguyen et al. [60], in contrast, use manually assigned spatial regions and a multi-camera tracking system to train a HMM for recognizing high level activities. They use an Abstract Hidden Markov Model and extend it with a memory that allows them to model a richer class of context-free and state-dependent behaviors. Also other researchers use different extensions to HMMs to overcome various limitations. Nguyen et al. [61] and Bui et al. [11] introduce the general concept of hierarchies to HMMs which become Hierarchical Hidden Markov Models (HHMMs) while Duong et al. [15] propose to use different layers in the HMM to account for hierarchies and durations. But HMMs are not the only models used for activity recognition. There are also approaches using Hierarchical Conditional Random Fields (CRFs) [88, 90], Hierarchical Maximum Entropy Markov Models (MEMM) [80] or Monte-Carlo based approaches [68].

Except for Sung et al. [80], who use RGBD cameras to detect activities from human body posture, most approaches rely on the use of object detections or motion tracking systems, which produce quite reliable data but are very expensive and intrusive. Perkowitz et al. [68] equip everyday objects in a human household with RFID tags to obtain sequences of object detections while humans perform everyday activities. For object detection, the users have to wear special gloves that serve as RFID readers. To free the participants of wearing special gloves for the RFID-tags Buettner et al. [10] developed a combination of RFID tags and accelerometers, called WISPs, which make their approach less intrusive and offer reliable object detections. But still the apartment they used for testing had to be equipped with four antennas and sensors had to be attached to the 25 objects they used.

Chapter 6

Human Activity Datasets

As robotics advances towards robotic helpers working together with humans, the field of activity recognition becomes increasingly important as a key enabler for effective and acceptable human-aware robot behavior. Recently, different approaches have evolved that use diverse methodologies and algorithms to recognize human activities from sensor data like motion tracking information. The increasing number of approaches makes it necessary to have more general datasets about human activities that enable researchers to compare the quality of their techniques using similar or even the same sensor modalities. In many cases, such datasets use sensor settings that include variants of RFID sensors [68, 10] to track the positions of objects or persons, but also approaches using video cameras for human motion tracking data [45, 82] are popular. Other sensor modalities like GPS [46], pressure sensors, reed switches or water-flow sensors [20] are also used in some datasets. In our work, we are specifically interested in analyzing human motion tracking data.

In this chapter, we give an overview about human activity datasets that use motion tracking data as sensor modality and highlight their specific advantages and disadvantages. We then document two new contributed datasets that were designed to support researchers in the fields of activity recognition and learning by capturing a variety of typical activities of daily living in a domestic environment.

6.1 Available Datasets

To generate human activity models and to test our approaches, we need realistic data that a robot deployed in a real human home would find. Since we assume our robot to be equipped with sensors that allow it to perform person tracking, human motion tracking data gathered in domestic environments is most suitable for our scenario. Unfortunately, the public availability of large datasets including such motion tracking data is limited. This is due to several challenges that one has to face when collecting realistic and comprehensive datasets of human task execution, including:

- **Sensor setting.** The choice of the sensor setting is of critical importance for the dataset. Different types of sensors produce different types of data, so the choice of an appropriate sensor setting strongly depends on the envisioned application of the data. The financial component plays an important role as well since a complex and high-accuracy sensor setting can very fast exceed 100.000 €. Cheaper sensors often have to deal with low accuracies or can only record data that contains less information. But depending on the type of application, low-cost sensors can already be sufficient to produce reasonable results for some activity recognition systems.

- **Ground truth data.** To enable researchers to evaluate their approaches, it is important that not only sensor data is available, but also labels for the ground truth about e.g. the activities that the human currently performs or the positions of the tracked person. Such labels are needed to enable evaluation and offer researchers a common basis on which they can compare different approaches. However, the generation of such labels is a tedious and time-consuming task since in most cases, the data has to be labeled manually. Ideally, the labeling of the data should be performed by a third party that is not involved in the data acquisition or the later use of the data.

- **Scenario selection.** The selection an appropriate environment can be an issue as well. Having machines that constantly monitor a person's activities immediately raises privacy issues, especially when collecting data that will be available online. Recording video data in private homes and making it publicly available is a big intrusion into a person's privacy. Therefore, most

datasets are recorded in lab environments with participants that explicitly agree with the data being made available to the public.

- **Activity selection.** Once a scenario is selected, one still has to decide which activities a participant should be performing. When regarding activities of daily living, such a selection should realistically reflect typical activities that a person performs in his/her daily life. In this case, one has to take care that the set of activities will not be too artificial since persons might act differently in a laboratory than in their accustomed homes.

- **Participants.** The recording of data about human activities can be very exhausting and time consuming for the participants. Finding participants that are willing to spend such an amount of time while not getting bored can thus be a problem. A very convenient solution for this challenge are a live-in laboratories where participants are offered to live in a sensor equipped home free of cost for an extended period of time, but not every institution has the means for such an option.

The focus of our research lies in the analysis of motion tracking data generated with low-cost sensor modalities like depth cameras, so we focus on datasets that allow the extraction of information about the location of persons over time. Usually one had to put much effort in the creation of such data since the recording implied setting up complex and expensive systems that were mostly based on cameras or markers. Such setups are expensive not only regarding their cost but also their computational efforts. With the recent introduction of cheap depth cameras like the Microsoft Kinect, the cost for capturing motion tracking data significantly decreased, enabling more people to capture data with less effort. However, this does not come without cost. Motion tracking data recorded with such depth cameras suffers from inaccuracies and the tracking range is spatially limited (usually to 0.5 to 4 meters distance from the sensor).

There are already a few datasets available that try to deal with some of the challenges in different ways and offer various types of sensor settings and thus different types of data. We will discuss a few datasets that looked promising for our application and explain two contributions towards datasets for human activities in this chapter.

6.1.1 The MIT PlaceLab

The MIT PlaceLab[1] is a sensor-equipped apartment laboratory in Cambridge, MA. The apartment consists of a living room, dining area, kitchen, office, bedroom, full bath and half bath, with each room including a micro controller and up to 30 sensors to record audio-visual data. A detailed overview of the MIT PlaceLab data is given by Intille et al. [25]. The apartment has been used to perform several studies and to record several datasets in which participant were asked to perform common tasks of daily living.

6.1.1.1 The PLIA2 (PlaceLab Intensive Activity 2) Dataset

This dataset is intended to be a short test dataset of manageable size in which a researcher was asked to perform different activities of daily living during a period of four hours in the live-in laboratory. Therefore, the participant followed a set of instructions and executed activities like cooking, washing, cleaning, making the bed and light cleaning around the apartment. Unfortunately, ground truth annotations are only partially available for this dataset and some of the files like the instructions that the participant followed, are not available online any more.

6.1.1.2 The PLCouple1 (PlaceLab Couple 1) Dataset

The most interesting dataset for our research is the PLCouple dataset which consists of anonymous sensor data of a couple that has been living in the apartment for 2.5 month. The advantages of this dataset are that it was independently labeled by a third party. Since the participants were living in the live-in laboratory for a long time, the activities that were recorded can truly be regarded as routine activities of daily living.The dataset focuses specifically on activities that happen between 6 and 12 am. During this time frame, the couple performed 23 different activities of which only 11 were performed in the kitchen area. However, also

[1] http://architecture.mit.edu/house_n/data/PlaceLab/PlaceLab.htm

this dataset comes with a set of restrictions: annotations of the PLCouple dataset are partly available for the public, but we could not use the sensor data since the two participants of the experiment shared many tasks. Due to financial restrictions, only one of the persons was tracked using a RFID reader. Also, the full audio and video data are not included due to privacy issues, so sensor data of many tasks that are part of a typical human routine were missing.

6.1.2 The TUM Kitchen Dataset

The publicly available TUM kitchen dataset[2] [82] offers several recordings of humans performing a table-setting task in a typical kitchen environment in two different ways. The dataset includes video sequences from four cameras, full body motion tracking data of the human, RFID tag readings of objects and magnetic sensor readings from furniture objects like cupboards and drawers in the environment. The data has been manually labeled to provide a ground truth for motion segmentation. Labels are provided for actions of the human body in general and for each hand separately. During the table-setting task, the participants used 6 objects (placemat, napkin, fork, knife, plate, cup), that were stored at three different locations (drawer, cupboard, stove), and the table that has to be set. The dataset uses two variations of a table-setting task: For the first variation, which we will call *impaired-person* in the rest of the chapter, the participants were only allowed to transport one item at a time. This task has been performed by all 10 participants where the order in which the objects were to be picked up has been fixed. Furthermore, two of the 10 participants also executed a second table-setting task in a mode we call *able-bodied-person* , where they were allowed to carry several items simultaneously. The motion tracking data of the TUM Kitchen Dataset is obtained by a marker-less motion tracking system with four cameras mounted at the ceiling of the kitchen as sensor input. The human poses are estimated in a recursive Bayesian framework using a variant of particle filtering [3]. This sensor setup helps the handling of occlusions since, in a majority of cases, most of the body parts are visible by more than one camera resulting in a quite accurate estimation of the human poses.

[2]http://ias.cs.tum.edu/software/kitchen-activity-data

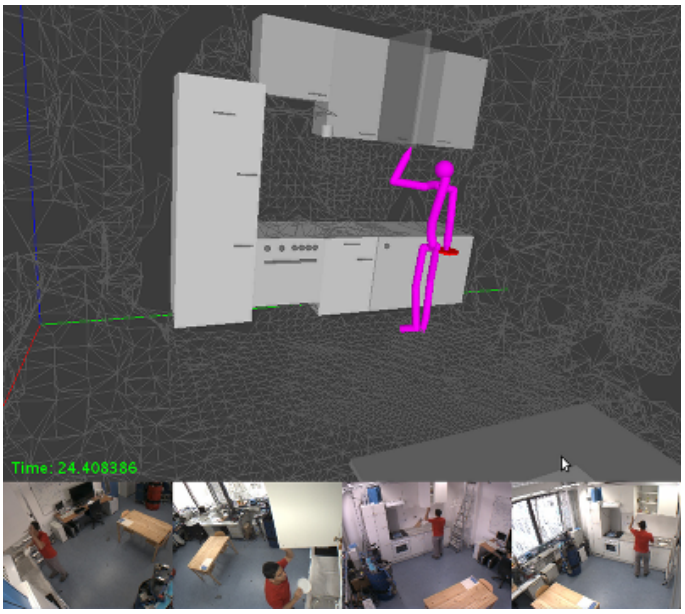


Figure 6.1.: Illustration of the data acquisition in the TUM Kitchen Dataset. The upper image illustrates 3D models of furniture objects according to a semantic map of the environment. The lower images show the output of the four video cameras that were used to obtain the human motion tracking data. Image courtesy of Moritz Tenorth [82]

.

A semantically annotated map of the environment is available in the KnowRob knowledge processing framework [84] which includes positions of the furniture, including the containers where the objects were stored. Furthermore, it includes articulation models of container objects that express how to open certain doors of cupboards and drawers. Tenorth et al. [85] even show how to create knowledge-linked semantic object maps that combine information of a the semantic map of this environment with common sense knowledge of publicly available data bases thus enabling a robot to perform reasoning e.g. about likely storage locations of different kinds of objects. A visualization of the semantic map, gathered motion tracking data and images of the four video cameras used for motion tracking, is shown in Figure 6.1

In chapter 4 we will explain how we use the TUM Kitchen Dataset to create a semantically annotated spatial model about positions that a person visits during a table setting task. Therefore, we will use the full body motion tracking data,

the ground-truth labels of the body and the hands as well as the the semantic map.

6.2 Contributed Datasets

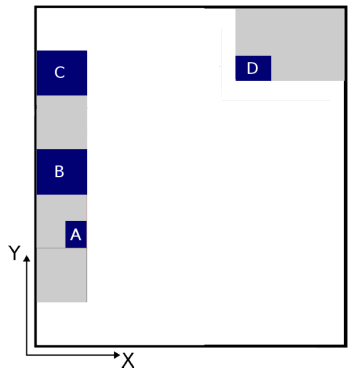
The previously explained datasets are a valuable source of information to the community and we used the TUM kitchen dataset for the generation of our models as we will explain in chapter 4. However, each of the datasets had some drawbacks that made them only partially useful for the evaluation of our anticipated activity recognition and expectations framework. To this end, we decided to create two more datasets, especially focusing on the needs of our system but also trying to keep the datasets as general as possible to make them useful for other researchers in the community.

6.2.1 TUM Kitchen Dataset 2

The TUM Kitchen dataset offers a decent set of motion tracking data featuring 10 different persons that perform two variations of a table setting task. However, the persons that were recorded only perform one task in two different ways. For our application of activity recognition, we need motion tracking data of persons performing different types of tasks to be able to test if our system can distinguish between different activities. We decided to create a new dataset and record additional motion tracking data in the same kitchen as the TUM kitchen dataset as well as in a different experimental kitchen.

6.2.1.1 Kinect Data in Garching Kitchen

This data was recorded in the same kitchen as used in the TUM kitchen dataset with the only difference that we varied the position of the table. A layout of the kitchen environment is shown in figure 6.2. We let 9 participants, which were mainly researchers of the area Computer Vision, perform the same two variants of the table setting task as in the TUM kitchen dataset, also keeping the order, in



A: Location of placemat and napkin
B: Drawer, location of cutlery
C: Cupboard, location of cup and plate
D: Table, place to be set

Figure 6.2.: The layout of the first environment used for the the TUM Kitchen Dataset 2. Blue rectangles mark the locations of dishes and objects that were used by the participants during our experiments.

which the objects had to be picked up, fixed. Furthermore, they were asked to clean the table after having a meal, including doing the dishes and putting the cleaned dishes back to where they belonged. One instance of a table-setting and cleaning activity is shown in Figure 6.3. In contrast to the TUM Kitchen Dataset, we used only one Kinect sensor instead of a complex multi-camera tracking system for human motion tracking and we recorded the 2D positions and orientation of the participants. The Kinect sensor is subject to less accurate motion tracking of human body joints as well as more occlusions than the multi-camera system of the TUM Kitchen Dataset. Another issue is the range of the Kinect sensor. The cupboard in the kitchen was located at the edge of the sensor range which caused even more inaccuracies and resulted in the motion tracking loosing track of the human in some cases. Figure 6.4 shows recorded motion tracking data for the three activities executed by five of the 9 participants.

6.2.1.2 Kinect Data in IAS Kitchen

For this data we kept the assignments of tasks of the participants the same but used another kitchen environment that differs in the layout of the furniture objects. The layout of the kitchen is shown in Figure 6.5. Again, 10 participants were asked to perform the two variations of a table setting task as in the TUM Kitchen Dataset and an additional cleaning task as in the TUM Kitchen Dataset 2 explained before. The participants were mainly researchers from the area of

Figure 6.3.: The experimental TUM kitchen that has been used in the TUM Kitchen dataset as well as for TUM Kitchen Dataset 2. Motion tracking data of three activities is collected using one Kinect sensor instead of a complex motion tracking system. The upper pictures show one participant performing a table-setting task while the lower pictures show the same participant cleaning the table and doing the dishes.

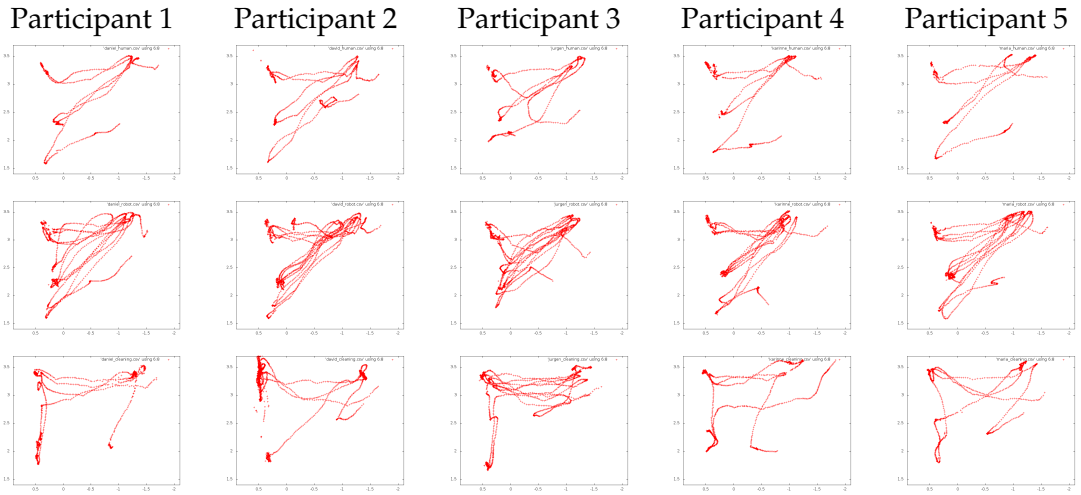
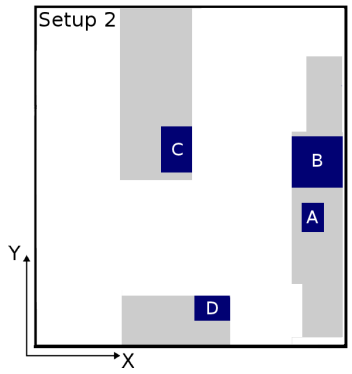


Figure 6.4.: Motion tracking data of five different participants of the experiment. The plots show 2D positions of the person tracked in the environment. The upper row consists of positions where the participants performed the able-bodied-person table setting task while in the middle row, they performed the impaired-person table setting task. The bottom row shows data recorded while the participants cleaned the table and did the dishes.



A: Location of placemat and napkin
B: Drawer, location of cutlery
C: Cupboard, location of cup and plate
D: Table, place to be set

Figure 6.5.: The layout of the second kitchen environment used for the TUM Kitchen Dataset 2. Blue rectangles mark the locations of dishes and objects that were used by the participants during our experiments.

Robotics and Artificial Intelligence. The 2D positions of five of the 10 participants performing the three different tasks are shown in Figure 6.6.

6.2.2 An Activity Dataset of a Human Morning Routine

The dataset described in the following section is one of the primary sources of information for testing and evaluation in this thesis. It has been created with the aim of capturing a typical morning routine of a single person in an apartment. In contrast to other datasets like the TUM kitchen dataset [82], which consists of several persons performing the same task, the focus for this dataset was to capture several activities of daily living performed by the same person at different days during an extended period of time. Furthermore, we specifically focused on a low cost sensor setting using only two Kinect sensors, which cost below 250 USD, in contrast to relying on complex motion tracking systems that are very costly. We decided to go for such a sensor setting since we think that a commercial service robot that will be deployed on the consumer market will use rather low cost components in order to be affordable for a wide range of people. Thus, our dataset reflects a realistic impression about how such a robot would perceive routine activities of its user when deployed in a human apartment.

The dataset has been released to the public on September 24th, 2013 and is available for download at https://hcai.in.tum.de/research/dataset.

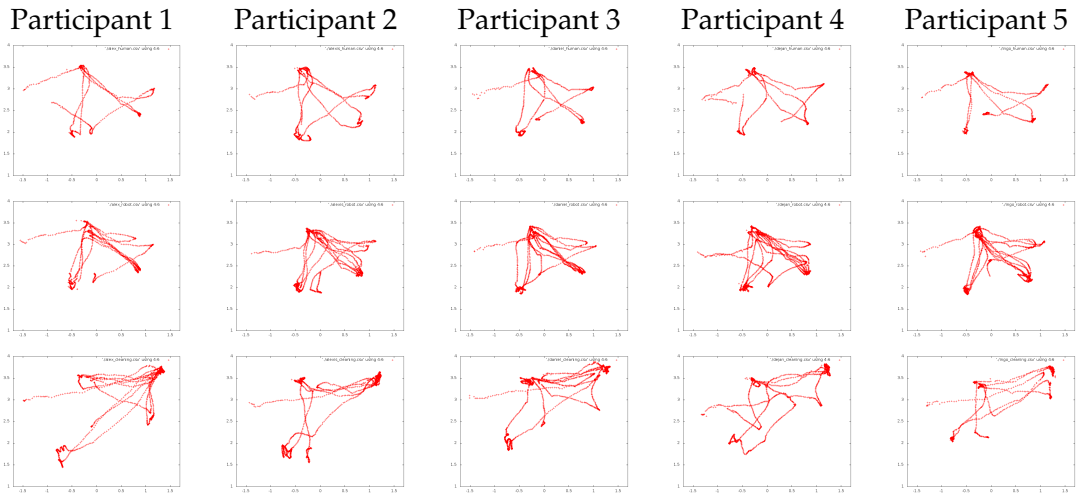


Figure 6.6.: Motion tracking data of five different participants of the experiment in the IAS kitchen. The plots consist of 2D positions of the person tracked in the environment. The upper row consists of positions where the participants performed the able-bodied-person table setting task while in the middle row, they performed the impaired-person table setting task. The bottom row shows data recorded while the participants cleaned the table and did the dishes.

6.2.2.1 Data Acquisition

Since our envisioned robot assistant is expected to detect and react on several routine activities of its human partner, we decided to record a dataset that includes such activities in a typical domestic setting. We first investigated typical morning routines of the annotated MIT PlaceLab PLCouple dataset and found that in a common morning routine of a person, the set of different activities that a human performs is very limited. To keep the already huge effort of collecting the data limited, we decided to limit the time-consuming parts of our dataset (e.g. the recording of motion tracking data) to activities that are performed in the kitchen.

We investigated a period of 14 workdays of a voluntary male test person that did not know about our system and was living in a rental apartment by himself. We asked the test person to note the activities that he performed before going to work as well as the locations at which he stands still while performing those

activities over 3 weeks. The exemplary notes of one day are shown in figure 6.7.

A typical morning routine of our participant consisted of 12-15 activities of which 7 happened in the kitchen area. The activities that were performed in the kitchen consisted of the following:

- preparing a drink
- drinking a glass of water
- preparing breakfast
- having breakfast
- cleaning the table
- packing a bottle of water
- leaving the room with the backpack.

In accordance with the MIT PlaceLab dataset also our data shows that the set of activities that a person performs during his morning routine is quite restricted and the execution of different activities does not show much variation. While the drinking water action mostly looked alike regarding the locations where the participant was standing, he took three different kinds of breakfast at different days: 8 times the participant had cereals for breakfast, 4 times he had curd cheese and 2 times he ate bread with butter and cheese. For comparison, in the MIT PlaceLab dataset, there were 23 activities that were performed between 6 and 12 am of which 11 were performed in the kitchen area. The set of activities that a person performs in specific rooms at specific times is a key insight that can have a beneficial effect on activity recognition systems targeted towards domestic areas. Taking the current time of day, which is very easily obtained, and the rooms where the activities are performed as a prior (assuming we have a semantic representation of our environment), such a system can limit the set of activities to consider for activity recognition to only a few ones. This limitation can have an advantageous effect on both the accuracy of the activity recognition as well as the run time of such systems.

Figure 6.7.: One day of the notes the participant took showing the order of locations he visited during his morning routine. The left picture shows a schematic top view of the apartment while at the right you can see the annotated activities that the participant performed at the corresponding locations.

6.2.2.2 Recording motion tracking data

To obtain real sensor data of human motion tracking as well as detected objects, we asked our participant to reenact his morning routine according to his notes in a sensor equipped kitchen environment. Therefore, we equipped an experimental kitchen-environment with two Kinect sensors. We use one of the Kinects to perform motion tracking of the human using the OpenNI tracker[3] and the second Kinect for object detections based on visual markers using the ar_pose[4] wrapper for ARToolKit. We decided to use visual markers for object detections in order to simulate an object detection system that might be available in future while keeping the effort limited. Since the size of the visual markers was of critical importance for the detection, we could only use markers on objects that were of a certain size. We equipped the following objects with markers: Cornflakes, Milk, Backpack, Curd cheese, Bottle. The ar_pose object detection returns the approximate pose of all detected objects in reference to the Kinect as coordinate transformations in the ROS TF framework[5].

Since in previous work, we made good experiences with the testing of algorithms in simulation, we also set up a simulated environment of the same kitchen using the MORSE simulator[6] [16]. This simulator provides us with a human avatar that can be controlled by a person to perform pick- and place tasks in simulated environments like in 3D computer games as explained in section 7.1. Simulations are used by several robotics groups for making the development process more efficient [89]. In the field of human-robot interaction, Woods et al. [94] have compared user behavior in simulation and a real-world study and have come to the conclusion that simulation-based experiments are a viable approach for first evaluations of novel techniques. Figure 6.8 depicts the experimental kitchen in its simulated version as well as the real kitchen environment. Furthermore, the picture shows a visualization of the motion tracking data and object detections that are obtained by the two Kinect sensors. The objects were placed at the same

[3]http://ros.org/wiki/openni_tracker

[4]http://www.ros.org/wiki/ar_pose

[5]http://www.ros.org/wiki/tf

[6]http://morse.openrobots.org

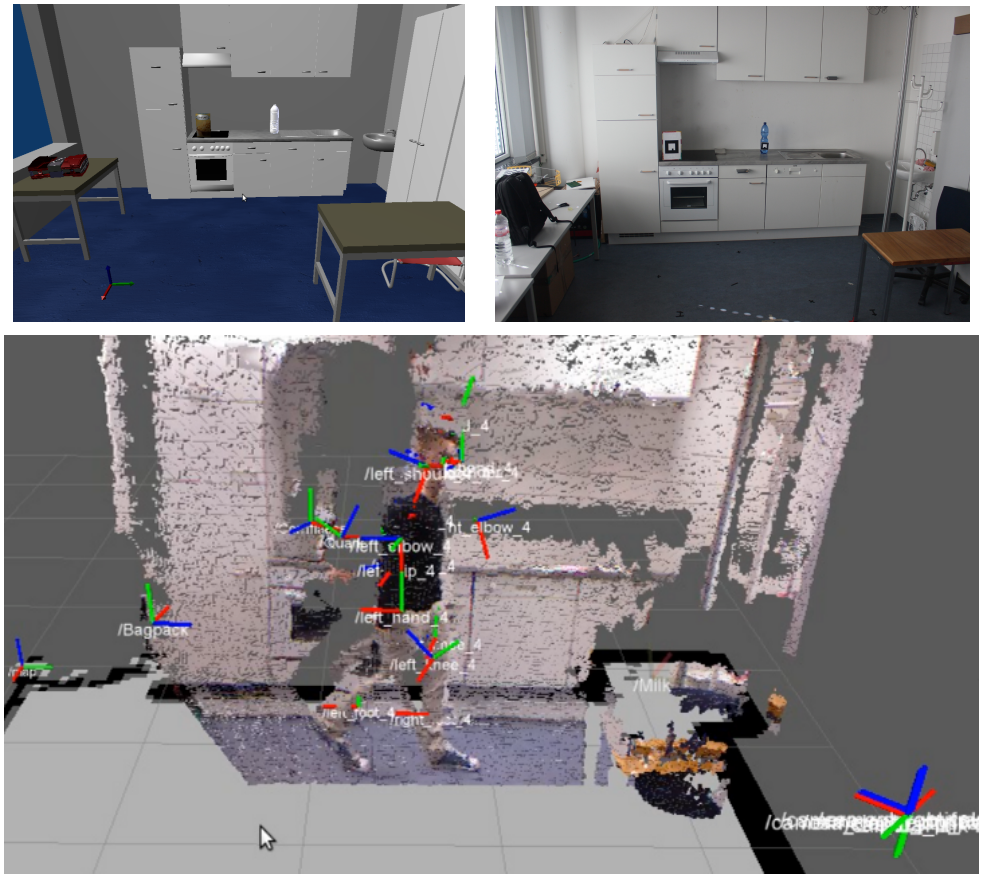


Figure 6.8.: The simulated environment in the MORSE simulator is shown on the upper left picture, while the upper right picture shows the real experimental kitchen environment. In the lower picture the sensor input of the real scenario is shown. The motion tracking returns coordinate transformations for each joint of the human while the visual marker detection returns a coordinate transform for each detected object. The map and the Pointcloud data are only shown for visualization.

typical locations in both scenarios (butter in the refrigerator, cutlery in a drawer etc.). The participant was asked to perform his typical morning routine in simulation as well as in the real kitchen in the same way as he did in the 14 days that he took notes of. A visualization of video data of parts of one activity along with the corresponding motion tracking data is shown in Figure 6.9.

In the simulated scenario, the test person used the human avatar of the MORSE simulator to open and close drawers, and move objects around. We tracked the position of all objects using a simulated object tracker and the human using a simulated person tracker. Instead of eating or drinking, the person was asked to take each object (glass, spoon) that is part if the corresponding task with the simulated human avatar for a short time and wait for a while (1 minute for drinking, 3 minutes for eating). In the real environment, we used a Microsoft Kinect for approximate human motion tracking. Although the motion tracker worked quite well while the test person was walking, it sometimes lost track of the person while sitting at the table or getting objects from the refrigerator. Especially when parts of the body are occluded, the motion tracking data is very inaccurate, which results in a "jumping" of some body joints of the tracked person. The performance of the marker-based object detection was varying in its quality quite a lot. While in many cases we had correct object detections, it often happened that objects were detected at wrong places or some detections of objects even were confused with other objects.

6.2.2.3 Comparison between Simulated and Real Data

To get an impression about how well experiments with our simulated human avatar can be compared to experiments in the real world, we investigated 2D motion tracking data of both experiments. Therefore, we plot the 2D location of the joint of the human neck over time and compare the 2D trajectories of the simulated experiments with the 2D trajectories of their corresponding real world version. We decided to use 2D trajectories as feature for the following reasons: On the one hand the activity recognition system (explained chapter 5) that we aim to implement into our expectation-based framework (explained in chapter 3) is based on 2D motion tracking data of humans. On the other hand, we don't

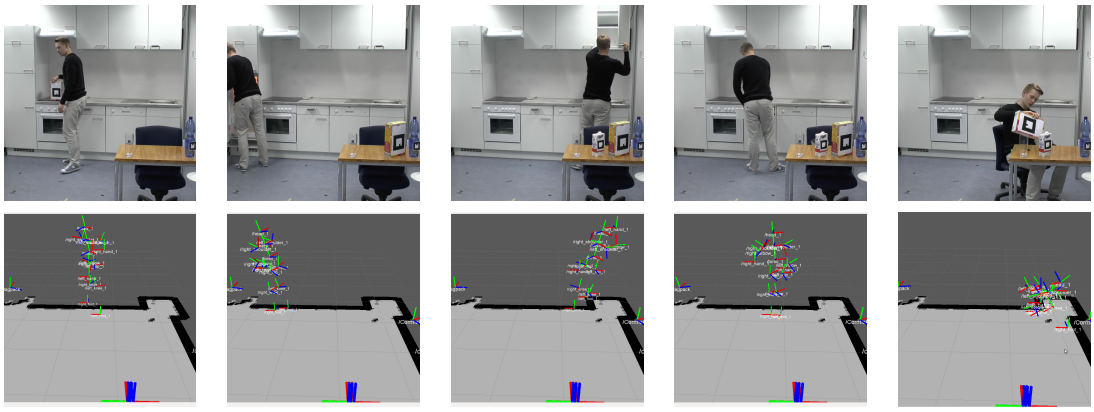


Figure 6.9.: A typical table-setting activity of the participant performed as part of his morning routine in the experimental kitchen environment. The upper images show video data of different parts of the preparation for having cereals as breakfast. The lower pictures show a visualization of the motion tracking data and object detections that were recorded using two Kinect sensors. For visualization, a 2D gridmap of the kitchen environment is shown as well. In the rightmost picture of the lower images, parts of the body of the participant were occluded by the table, resulting in inaccurate and even wrong tracking for many joints of the tracked human model.

think it makes much sense to compare full-body motion tracking data between simulation and real world since the simulated avatar in the MORSE simulator is currently not designed to produce movements that realistically mimic movements of real persons. A qualitative comparison between motion tracking data from different simulated and real world experiments is shown in Figure 6.10.

The comparison between the 2D motion tracking data of the simulated and real world experiments suggest that a human using a simulated avatar tends to perform movements in a more straight-line fashion and not following curved paths too much. We think this is due to the fact that the control of such an avatar with keyboard and mouse is rather unnatural to humans. This leads to behavior that prefers safely reaching the goal instead of optimizing paths with respect to time. In our experiments this resulted in the participant mostly turning in place and then approaching the goal in a straight line when using simulation. Another reason for the differences in the movements can be the acquisition of the human position. While in simulation, the person is more or less static regarding its joints,

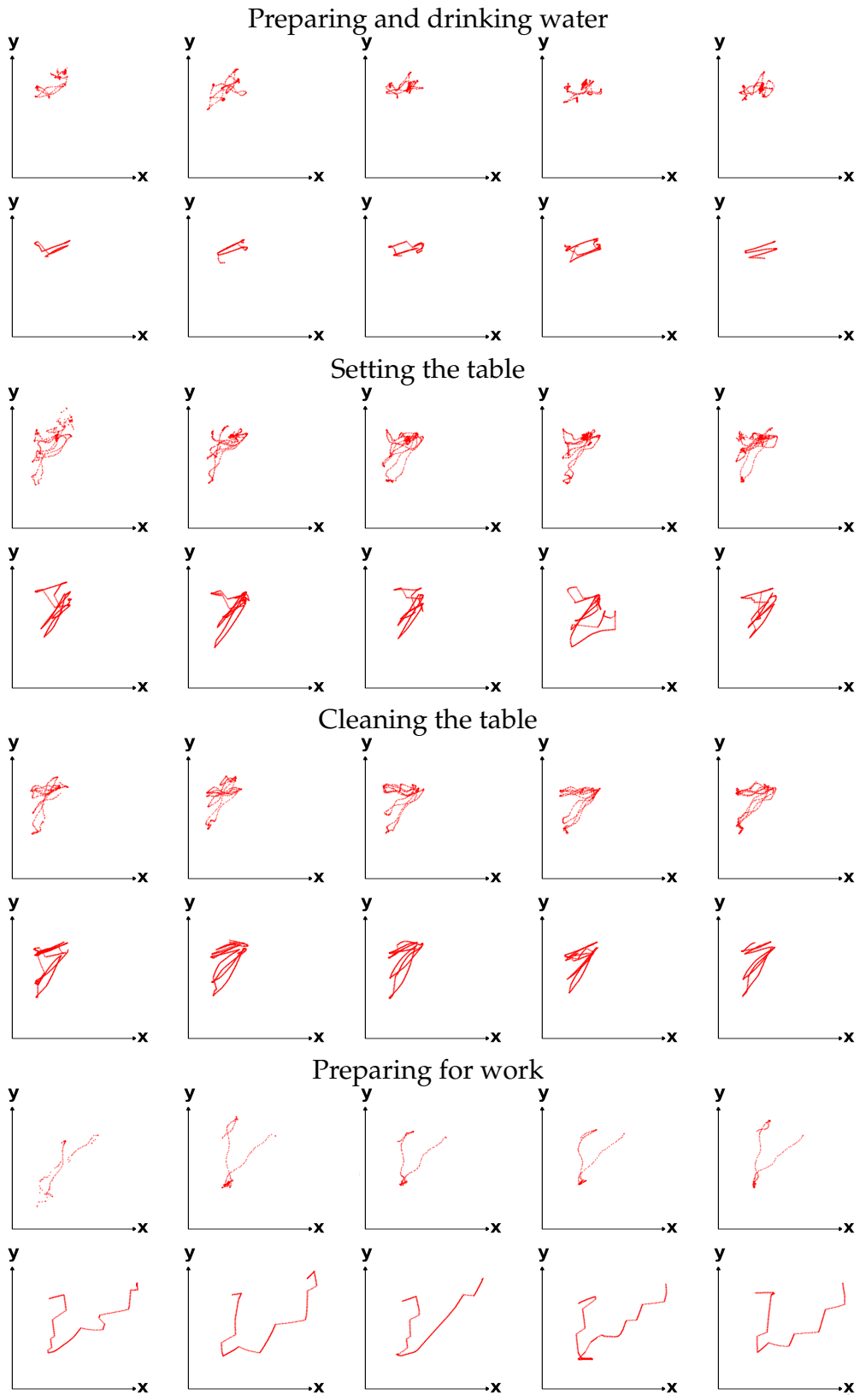


Figure 6.10.: 2D motion tracking data of different tasks in simulation (lower rows) and real world experiments (upper rows).

a real person never really has the same posture while moving. It might lean a bit forward while accelerating or sidewards when grabbing an object. These alterations in the human posture can also be the reason for the motion tracking data in the real world example looking less linear than the simulated data. An overview of positions where the human was standing still (i.e. moving less than 50cm/second) summed over all 14 experiments is shown in figure 6.11 for the real world experiments (left picture) as well as for simulation (right picture). As can be seen, those positions are in similar areas which comes as no surprise since the simulated scenario has been created as an exact model of the real kitchen and the proportions of the human avatar match roughly the proportions of a real person. This has the consequence that a human that is grabbing an object from a storage location in simulation should approximately have the same distance to that storage location as a real person would have in a real world scenario. Differences can mainly be seen at the location that corresponds to the position in front of the table (approximately at 2,3.5). At this location there is a much higher variation of positions in the real data than there is in the simulated scenario. This is due to various reasons: On the one hand, the person moves away the chair in front of the table and sits on it in the real scenario while, in simulation, the person just stands in front of the table for a fixed time. On the other hand, in the real data, the parts of the body of the participant are occluded when he is sitting at the table and eating or drinking. This has the effect that the tracker has a hard time correctly tracking the positions of the joints of the human which results in a "jumping" of some of the joints and leads to inaccurate 2D position estimations as is shown in Figure 6.12. Also the position that corresponds to the location of the door (approximately at 4,4) is different in the datasets. This is due to the limited range of the Kinect tracking. While in simulation the person was tracked until he reached the door, in the real world scenario, the door was out of the range of the Kinect tracking. In this case, the last detection of the person was mostly in the area around the 2D location at 3.5,3.5, which represents the border of the Kinect tracking range when walking towards the door.

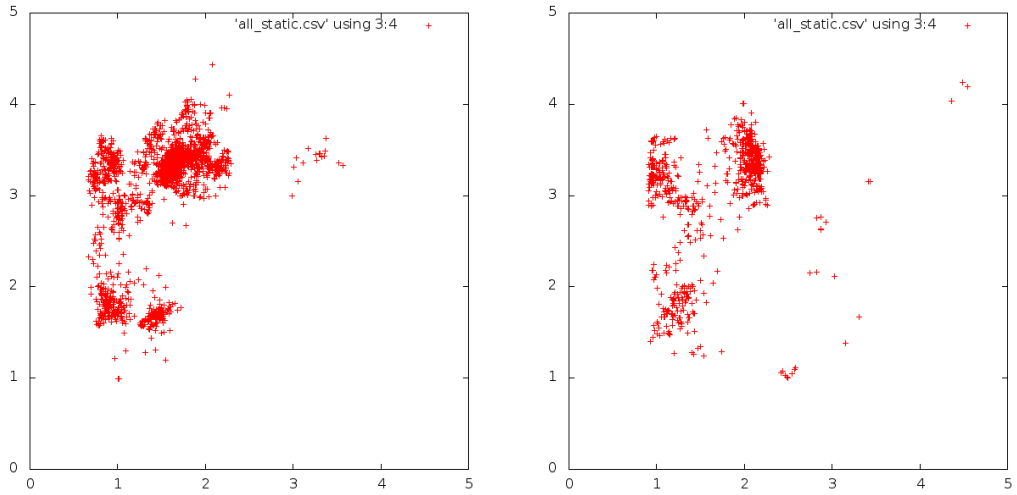


Figure 6.11.: These plots show 2D positions where the participant stood still for a little while. The left picture shows the positions for the real world experiments while the right picture shows the corresponding positions for the simulated data.

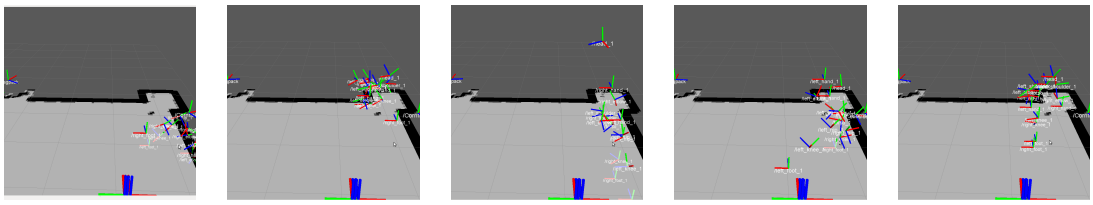


Figure 6.12.: Partial occlusions of the body of the participant give the motion tracker a hard time in estimation the positions of the joints. This results in a "jumping" of some of the joints of the person when e.g. sitting at the table. The images have been taken shortly after another when the person was sitting at a table and the legs were occluded from the Kinect sensor.

Chapter 7

Evaluation

In this chapter, we carefully evaluate the approaches that were introduced in this thesis. We will first provide evaluations of the contributions separately and finally show an exemplary application of the fully integrated system. It enables a robot to detect unexpected behavior of a person in a simulated and a real-world scenario. To this end, it will be using human activity models (as explained in chapter 4) to perform activity recognition and monitoring (as explained in chapter 5) and embed predictions about likely next locations of a person in the expectations framework (explained in chapter 3).

Parts of the evaluation and application scenarios were performed in simulation using the open-source simulator MORSE. Several contributions to this simulator were also developed throughout this thesis and were integrated into MORSE. An introduction to the MORSE simulator as well as the relevant contributions that were developed within this thesis are provided in section 7.1. Experiments in real world environments as well as simulated experiments are evaluated using the datasets described in chapter 6.

7.1 The MORSE Simulator

The availability of realistic simulation tools have drastically reduced the effort for robot testing by providing safe and readily available tools. Robot simulations are used by various robotics groups for making the development process more efficient [89]. In the field of human-robot interaction, Woods et al. [94] have compared user behavior in simulation with a real-world study and have come to

the conclusion that simulation-based experiments are a viable approach for first evaluations of novel techniques.

For testing and first evaluations of this work, we use the Modular OpenRobots Simulation Engine (MORSE)[1] [16] , which is a project of LAAS-CNRS to supply researchers with a realistic, physical simulator for a wide variety of robot tasks. This simulator is based on the free and open-source 3D modeling software Blender[2] that has the open-source physics engine Bullet[3] integrated in its game engine, which is used for realistic physics simulation. MORSE comes with different types of robots that can interact with the environment and can be controlled via robotics middlewares like ROS, Yarp, Pocolibs or MOOS. It also offers the possibility to control the degree of realism of the simulation by providing different levels of abstraction for sensors and controllers. While, for instance, somebody working in the field of perception might want to access low-level sensor data from sensors like depth cameras, someone working on high level planning might only want to deal with the positions of objects in the environment.

One unique feature of MORSE is the possibility to include an animated human avatar that can move in the world and manipulate objects in real time during the simulation. The human avatar has not only been used for testing human-aware navigation strategies before carrying out experiments with real humans [38], it has also been used to evaluate such strategies using video-based user studies as shown by Lichtenthäler et al. [47] . We have contributed to the MORSE project by adding an intuitive control for the human avatar oriented along 3D computer games [44]. The avatar enables the combination of the reactive and sometimes unpredictable behavior of a person with a simulated robot. The user takes a third-person perspective behind the human to move around as illustrated in Figure 7.1 on the left. While moving around, the camera tries to avoid objects and walls placed between the camera and the human avatar to prevent occlusions. All objects that can be interacted with can be displayed by pressing a key (also illustrated in Figure 7.1 on the left). When the user decides to interact with an

[1]http://morse.openrobots.org

[2]http://www.blender.org/

[3]http://bulletphysics.org

object, the camera switches to a first-person perspective and offers an interface showing possible actions the user can take when pointing to specific objects (as shown in Figure 7.1 on the right). Those actions include picking up and releasing objects, opening and closing drawers, cupboards and doors and switching on and off specific objects like a light or an oven.

Figure 7.1.: Left: third-person view of the human avatar that is used to navigate in the environment, displaying the names of objects that the human can interact with. Right: first-person perspective of the human avatar that indicates a possible "pick-up-action" with the bread.

The motions of the avatar are animated using Blender armatures, inverse kinematics, and predefined movement loops. The avatar can easily be controlled using the mouse and keyboard. Also a combination of the Microsoft Kinect and the Nintendo Wiimote are possible. This enables users to perform pick and place actions in a simulated world, while at the same time, simulated robots can be controlled by the supported robotic middlewares. Tracking of the simulated human can be done at two different levels of abstraction. Using the video cameras or depth sensors in combination with motion tracking approaches, the recognition of the human position and body pose can be done realistically with the associated computational cost and uncertainties. Alternatively, the avatar can directly export the position of all of its joints and feed them back to the robot, simulating a full motion capture system and avoiding the processing costs and uncertainties. For our experiments, we chose to use the directly exported positions of the human joints since human motion tracking is not the focus of this work. In the

following sections of this chapter, we will repeatedly use the MORSE simulator for testing and evaluation of the approaches described in this work.

7.2 Evaluation of Activity Recognition

In this section, we evaluate our approach on activity recognition based on Context Dependent Spatial Regions (CDSRs) as described in chapter 4 and 5. We therefore first assess the quality of distinguishing single activities using a string comparison method as explained in section 4.2 in a kitchen environment. Afterwards, the results will be compared to activity recognition results based on HHMMs as explained in chapter 5 and an evaluation of activity recognition and monitoring in real time will follow.

7.2.1 Single Activity Recognition

For this part, we use data of the TUM Kitchen Dataset and TUM Kitchen Dataset 2, which come with motion tracking data different persons performing two different types of table-setting activities as well as a cleaning activity as explained in chapter section 6.2.1. The evaluation of activity recognition will be performed based on two different features. We will first evaluate the use of durations during which a person is standing at certain locations as feature. Afterwards, we will evaluate the order in which different locations are visited as a feature for activity recognition.

7.2.1.1 Activity Recognition Based on Durations

Generally, the duration a person spends at certain locations during the execution of an activity differs from person to person to some degree. In this first experiment, we try to figure out, if the time that different persons spend at certain locations can be enough evidence to distinguish different activities. Therefore, we assume those durations to be dependent on the amount of manipulation that a person performs at a specific location. For instance, we assume a person to

spend a longer amount of time in front of a drawer when picking up objects from inside the drawer than he/she would when picking up objects from a plane like a table. We used data from the TUM Kitchen Dataset (see section 6.1.2) and the TUM Kitchen Dataset 2 (see section 6.2.1) to investigate if durations at the different locations are an adequate feature for activity recognition across different persons and environments. The datasets include motion tracking data of different persons performing two different variations of table setting activities in an experimental kitchen environment (Setup 1). Furthermore, the TUM Kitchen Dataset 2 features a third cleaning activity data in the same kitchen environment and an additional second kitchen environment (Setup 2), in which both table setting activities as well as the cleaning activity were performed by the participants. We investigate the durations that persons spend at furniture objects in both setups of the datasets. Figure 7.2 shows average durations over 10 participants of each dataset at four different types of locations for a table-setting activity. The upper picture represents data from the TUM Kitchen Dataset, whereas the lower two pictures illustrate data from TUM Kitchen Dataset 2. The locations represent positions of the objects that the participants interact with when executing the two table-setting activities and the cleaning activity. In Setup 2, there are no objects stored on a cupboard, but in another drawer instead, resulting in no values for the cupboard in the chart. Dishwasher, stove and table represent general planes, but we distinguish between the table and the other planes since the table represents the final location of the objects and we assume that the duration might also be different between picking-up and placing actions. Intuitively, picking up an object should consume less time than placing objects when e.g. setting a table since a correct placement should consume more care then a simple grasp of an object.

For this experiment, we will estimate how well we can distinguish between the different activities in the different settings by the calculation of confidence values that indicate how confident we are of having observed a specific activity. Using the (ground-truth) Spatio Temporal Plan Representations of the the impaired-person table setting task generated from the TUM Kitchen Dataset as a model, we calculate confidence values as explained in section 4.2.3 for all activities of the two experimental setups of the TUM Kitchen Dataset 2. The confidence values express a measure of similarity between the model and the observations. The

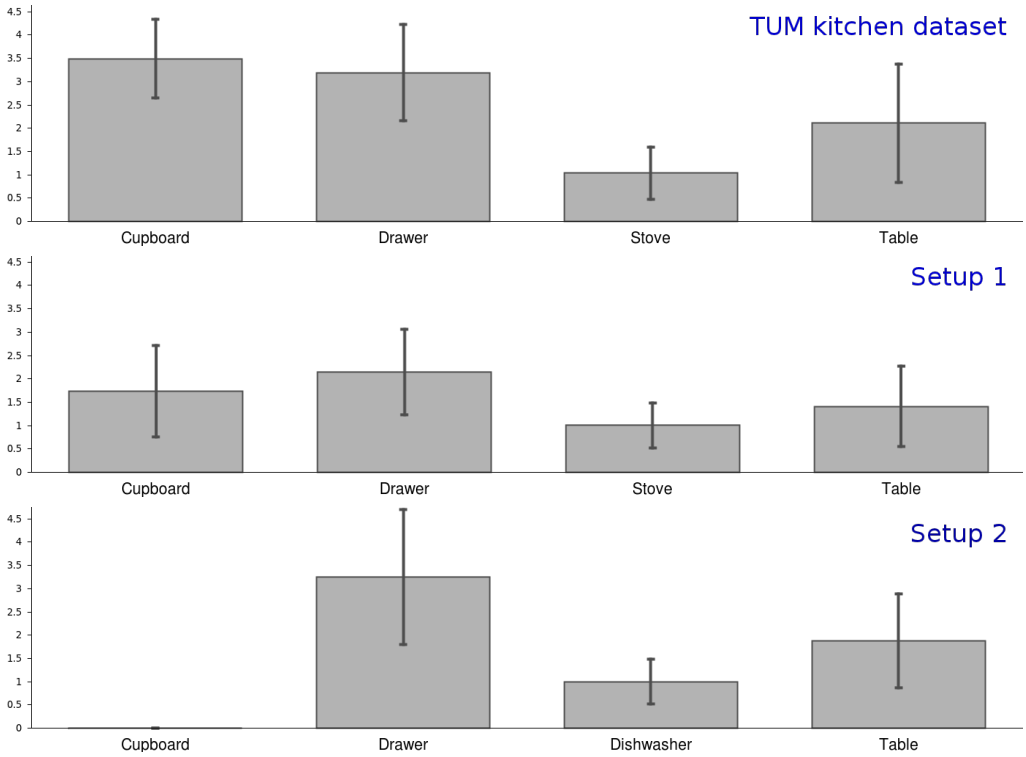


Figure 7.2.: The pictures show average durations of a person standing still at different locations during task execution, illustrated by their means and variances. The upper picture shows the data from the TUM kitchen dataset, the middle picture shows data from Setup 1 of the TUM Kitchen Dataset 2 and the lower picture represents data from Setup 2 of the TUM Kitchen Dataset 2.

higher the confidence value, the better the observations fit to the model and the more likely it is that the corresponding activity has been executed by the observed person. Table 7.1 shows the confidence values c_p for each activity in both scenarios averaged over all participants.

a) Impaired-person table setting activity

Activity	c_p Setup 1	c_p Setup 2
Impaired-person table setting	0.524	0.593
Able-bodied-person table setting	0.448	0.506
Cleaning task:	0.191	0.350

b) Able-bodied-person table setting

Activity	c_p Setup 1	c_p Setup 2
Impaired-person table setting	0.175	0.068
Able-bodied-person table setting	0.081	0.134
Cleaning task:	0.076	0.151

c) Cleaning task

Activity	c_p Setup 2
Impaired-person table setting	0.564
Able-bodied-person table setting	0.581
Cleaning task:	0.368

Table 7.1.: Confidence values for different activities of the TUM Kitchen Dataset and TUM Kitchen Dataset 2 using only durations as features.

The confidence values for an observation of the impaired-person table setting task in section a) indicate that the most likely activity corresponds to a table setting task in both scenarios. However, the able-bodied-person table setting tasks also have a relatively high confidence value compared to the cleaning tasks. This can be explained by the fact that the time, a human spends at a storage location does not vary significantly between actions where he picks up one single object or several objects. One could regard the recognition of activities based solely on durations as successful in this case. Section b) of table 7.1 shows the confidence values when observing the able-bodied-person table setting task. In this case, the activity recognition does not perform well. In Setup 1, it detects the impaired-person table setting task as the most likely one and in Setup 2, it even detects the cleaning task as the one with the highest confidence. Observing the cleaning task generated from data of Setup 1 of the TUM Kitchen Dataset 2 (the

TUM Kitchen Dataset itself provides no cleaning task) leads to the confidence values shown in section c) of table 7.1. In this experiment, the confidence values identify the wrong activities to be the most likely ones.

Considering the results of all of the experiments, the durations that persons spend at certain locations might be an indicator for some activities but solely this information cannot be used as a reliable feature for activity recognition. We think, the unreliable recognition rates of the able-bodied-person table setting task and the cleaning task result from the high variances of the activity models regarding the durations. The durations that persons spend at certain locations seem to be quite different from person to person but also from time to time, since it might be influenced by factors that we don't account for in our model.

In the next sections, we will thus focus on the evaluation of activity recognition based on the patterns of locations that a person visits during task execution which seem to be a more promising candidate feature for activity recognition.

7.2.1.2 Activity Recognition Based on Location Patterns using Generalized Levenshtein Similarity

In this experiment, we evaluate how well we can distinguish different activities based on the order in which different locations are visited by a person during task performance. We therefore use data from the TUM Kitchen Dataset to generate a spatial model of the environment as explained in section 4.1. The spatial model consists of Context Dependent Spatial Regions (CDSRs) modeled as two-dimensional gaussians . It represents locations at which a person is standing still while performing an action during the execution of an activity. The spatial model generated from data of the TUM Kitchen Dataset is used for activity recognition in the same kitchen based on data of the TUM Kitchen Dataset 2 (Setup 1). We furthermore transfer the spatial model to the second scenario of TUM Kitchen Dataset 2 as explained in section 4.1.2 to test if activity recognition can be performed in a previously unknown environment based on a transferred model (Setup 2). Both spatial models are illustrated in Figure 7.3.

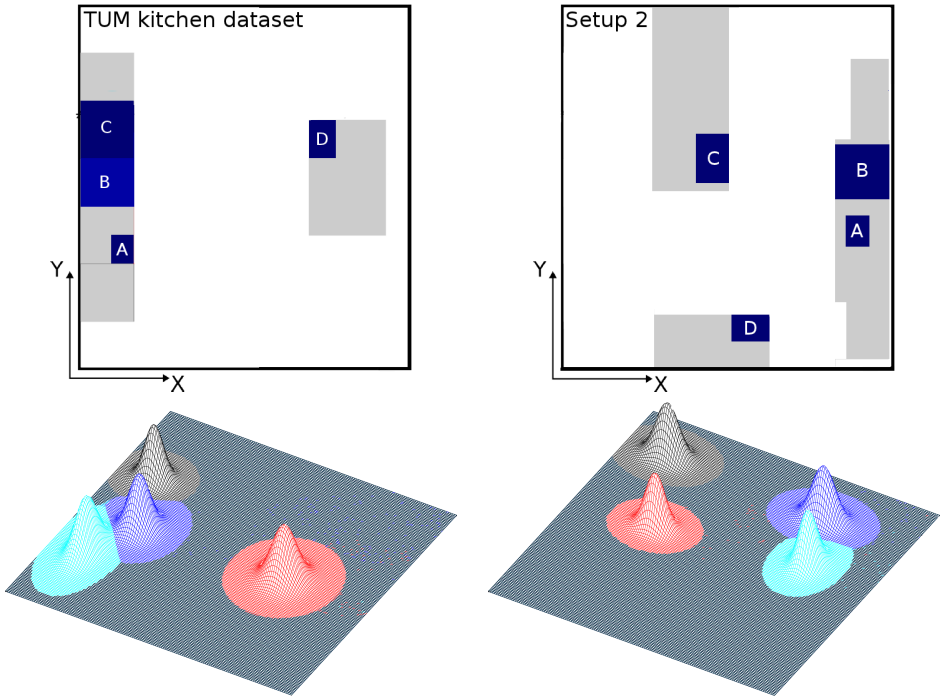


Figure 7.3.: The left picture shows the setup of the TUM Kitchen Dataset and corresponding spatial model (Setup 1). Location "A" represents the position of the place mat and napkin, "B" describes the drawer where the cutlery is stored, "C" represents the cupboard where the plate and cup are in and "D" is the table that is to be set. On the right picture, Setup 2 is shown. Here our model is transferred to the second kitchen of TUM Kitchen Dataset 2. The expected locations of the model have been adapted to the new environment resulting in the spatial model shown on the lower right picture.

Using data from the TUM Kitchen Dataset and the corresponding spatial model, we generate STPRs of the impaired-person table setting task and the able-bodied-person table setting task as explained in section 4.2. Using this STPR as a model, we calculate confidence values for each plan based on the *Generalized Levenshtein Similarity (GLS)* [95] for Setup 1 and Setup 2 as explained in section 4.2.3 These values express a measure of likelihood of the observed activity towards each modeled activity. Table 7.2 illustrates the confidence values when observing the impaired-person table setting task in section a) and the values when observing the able-bodied-person table setting task in section b). The values indicate that in our experiments consisting of three different tasks, we are able to identify the table-setting task.

a) Impaired-person table setting

Activity	GLS $_{\text{Setup 1}}$	GLS $_{\text{Setup 2}}$
Impaired-person table setting	0.982	0.943
Able-bodied-person table setting	0.429	0.429
Cleaning task:	0.357	0.340

b) Able-bodied-person table setting

Activity	GLS $_{\text{Setup 1}}$	GLS $_{\text{Setup 2}}$
Impaired-person table setting	0.429	0.500
Able-bodied-person table setting	0.929	0.964
Cleaning task:	0.714	0.714

Table 7.2.: Confidence values for different activities of the TUM Kitchen Dataset and TUM Kitchen Dataset 2 using the Generalized Levenshtein Similarity (GLS).

In contrast to the durations in section 7.2.1.1, the values of the impaired-person table setting task strongly deviate from the able-bodied-person table setting task since their patterns are quite different. Thus, a reliable distinction between the two table setting tasks is possible in this case. Unfortunately, we were not able to create a general model of the cleaning task that could have been used for activity recognition. This is due to the fact that while recording the data of TUM Kitchen Dataset 2, we did not specify the order in which the persons should clean which object. Thus, the order deviated from person to person and the strictly linear structure of STPRs does not provide the possibility to model such variations.

In the experiments conducted so far, the spatial model, which is used to map positions of a person to CDSRs, only consists of four CDSRs that furthermore were distributed around the kitchen in a way that their corresponding gaussians did not overlap. Thus, the qualitative descriptions of locations queried from the spatial model could easily be assigned correctly when the spatial model was queried with a position. But when trying to recognize several activities in confined spaces, the mapping from two dimensional coordinates to qualitative spatial regions might not always be as unique as in the examples so far. To evaluate our approach in such a setting, we set up a second scenario using the Morning Routine Dataset described in section 6.2.2, accounting for a variety of activities that are to be detected. The challenge in this environment is that the person is standing at several CDSRs that are close to each other during the performance of different tasks. When building the spatial model, this results in gaussians of the CDSRs that are close to each other and partly overlap as illustrated in Figure 7.4.

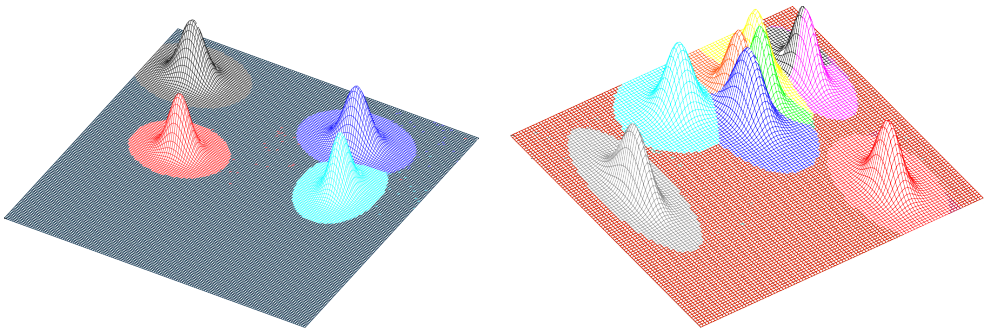


Figure 7.4.: The left picture shows the spatial of the TUM Kitchen Dataset with only a four context dependent spatial regions as used in the first experiments. In this case, it is easy to distinguish between different locations. The right picture shows the spatial model of the same kitchen environment when accounting for more CDSRs. Since some furniture objects are close to each other, some of the CDSRs overlap, making a correct assignment of human positions to a CDSR challenging.

We decided to use only 12 of the 14 days of the Morning Routine Dataset, where the participant had cereals or curd cheese for breakfast to keep the effort limited on the one hand. On the other hand, there were only 2 more days where the participant had bread for breakfast and the generation of a general model

for "Prepare Bread" and meaningful tests can hardly be done with such a small number of instances. We define the activity models from ground truth data and perform activity recognition for each of the 12 days using the GLS approach. Activity recognition has quite a hard time distinguishing between the different activities as can be seen in the exemplary case of “Prepare Cereals”. Results for this activity averaged over 8 days of the dataset (in which the participant of the dataset had cereals) can be seen in table 7.3 for the simulated and the real dataset. Here, $P(a)_{sim}$ represents the probability of activity a being observed in experiments of the simulated dataset and $P(a)_{real}$ corresponds to the same probability for experiments in the real dataset. For this evaluation, we calculated probabilities from the confidence values by normalization over all activities. This step is necessary to enable a comparison between activity recognition using GLS with activity recognition based on Hidden Markov Models as we will show in the next section. Even though, the maximum of the average values indicates that mostly “Prepare Cereals” is correctly classified as the most likely activity, variance between the mean values of all plans is small and in some cases, activity recognition is undecided or wrong. In the simulated data, still 7 of the 8 “Prepare Cereals” instances are classified correctly, but with low confidence. In one out of the 8 cases, “Prepare Cereals” has been classified wrongly as one of “Prepare curd cheese” or “Clean table after cereals” (with the same probability). For the real data, only 2 of the activities were correctly classified, in 4 cases, classification was wrong and in the remaining 2 cases, probabilities for two plans were the same (including the correct one).

Activity	$P(a)_{sim}$ (%)	$P(a)_{real}$(%)
Drink water	11.24	11.87
Prepare cereals	23.40	19.94
Prepare curd cheese	21.40	19.49
Clean table cereals	18.34	18.11
Clean table curd cheese	16.61	17.66
Prepare work	9.00	12.93

Table 7.3.: Probabilities for activities when “Prepare cereals” is performed using STPRs and GLS for activity recognition.

In contrast to the first experiment, activity recognition performs significantly worse, which we found to be caused by observations of incorrectly labeled spa-

tial regions when querying the spatial model. In the first experiment, there were only few spatial regions that did not overlap as shown in Figure 7.4 on the left. Thus, the spatial model could return one unique CDSR with high confidence when queried with a position. In the second setting, we accounted for far more spatial regions that were located close to each other, resulting in an overlapping of the corresponding gaussians illustrated in Figure 7.4 on the right. This is due to the fact that in some kitchens, cupboards are for example located above drawers or other cupboards, which results in the human standing at the almost same location when picking up an object from a drawer or a cupboard that is directly above the drawer. The overlapping of such CDSRs results in high uncertainties in the assignment of a location of a person to a CDSR. Our approach of activity recognition performs such an assignment by querying the spatial model each time the person is standing still. Since STPRs currently do not support a model of uncertainty in the observations, activity recognition using the GLS approach heavily relies on the correct assignment of positions towards CDSRs and thus performs worse in settings like this one.

7.2.1.3 Activity Recognition Based on Patterns using HHMMs

To account for the high uncertainties that come with overlapping CDSRs and also for uncertainties in the activity patterns, we found the application of Hierarchical Hidden Markov Models (HHMMs) to be a perfect fit. HMMs have already been proven to work well in the area of activity recognition. We use the same data as in the last experiment for the generation of the activity models from the ground truth data of the Morning Routine Dataset to learn the state transition model of the HHMM. The observation model of the HHMM is estimated as explained in section 5.2.2 using 10.000 iterations.

To perform activity recognition using the HHMM, we estimate probabilities for all of its internal states, the activities, at each time step using the Forward-Backward Algorithm as explained in section 5.2.3. This provides us with the posterior marginals over all internal states and thus a probability distribution over all modeled activities. Table 7.4 shows probabilities for each activity averaged over the 8 days when the observed person performs the “Prepare Cereals” activity.

Compared to the probabilities when using GLS as in table 7.3, we see that HHMM-based recognition clearly outperforms the GLS approach. The probabilities indicate that the observations fit well to one of the "Prepare Food" tasks. However, the distinction between "Prepare Cereals" and "Prepare curd cheese" is difficult, since the difference between both plans is rather small. When preparing cereals, the person goes to the ceramic glass cook top (location of the cereals) first, whereas when preparing curd-cheese, he first visits the fridge. The location of the ceramic glass cook top and the refrigerator are very close to each other (as can be seen in picture 6.8 in chapter 6), so especially when using the noisy Kinect tracking, we can hardly distinguish between those places.

Activity	$P(a)_{sim}$ (%)	$P(a)_{real}$(%)
Drink water	2.30	1.78
Prepare cereals	55.56	41.98
Prepare curd cheese	26.68	34.08
Clean table cereals	9.84	5.57
Clean table curd cheese	3.44	14.20
Prepare work	2.18	2.42

Table 7.4.: Probabilities for activities when "Prepare cereals" is performed using HHMMs for activity recognition.

This experiment shows that activity recognition based on patterns of locations using HHMMs can better cope with uncertainties in the observations. One further advantage is the possibility of using HHMMs with a continuous stream of observations, making it applicable for live-activity recognition. Since the GLS approach needs to be given the full sequence of observations to compute a similarity measure, such a live application would not be possible in the current GLS-based implementation.

7.2.2 Live Activity Recognition Using HHMMs

The experiments illustrated above show that the HHMM based approach is able to detect single activities even though some spatial regions are close to each other and thus overlapping. However, both experiments so far tried to recognize single activities, given the starting- and ending times of the activities are known. The

more interesting, although more difficult use case of our activity recognition is the observation of a person performing several activities after another. We therefore set up our HHMM based approach to perform live activity recognition in a kitchen environment. In the following experiments, we will measure how well this approach is able to detect different activities over time, including transitions between activities. We used the Morning Routine Dataset again and calculate precision and recall values for each activity a in the following way:

$$precision = \frac{\left|t_a \cap t_a^*\right|}{\left|t_a^*\right|}.$$

t_a represents the time when activity a has been executed by the participant according to the ground truth labels of the dataset and t_a^* stands for the time where the detection estimates activity a to be the most likely one. Accordingly, *recall* is calculated in the following way:

$$recall = \frac{\left|t_a \cap t_a^*\right|}{\left|t_a\right|}.$$

Furthermore, we calculate the *accuracy* which is the proportion of true classification results (true positives and true negatives) and all classification results during the whole observation period t_{obs}:

$$accuracy = \frac{\left|t_a \cap t_a^*\right| + \left|\bar{t_a} \cap \bar{t_a^*}\right|}{\left|t_{obs}\right|}.$$

$\bar{t_a}$ corresponds to the time when activity a has not been performed and $\bar{t_a^*}$ represents time periods when activity a has not been classified as the most likely activity.

The transition probabilities of the HHMM were manually generated using the ground-truth data of the Morning Routine Dataset. We did the experiment using data from simulation as well as the real data and the resulting precision, recall and accuracy values for each activity can be found in Table 7.5 a) for the simulated data and Table 7.5 b) for the real data. Although, the precision and recall rates might suggest otherwise, the "Prepare Work" activity is recognized very well in most cases as can also be seen in a qualitative evaluation in figure 7.5.

The reason for having lower precision and recall rates in our experimental setting is that, when recording the Morning Routine Dataset, we unluckily chose the position that was used for the initialization of the motion tracker (and the starting position of the human in simulation) close to a CDSR that is (uniquely) assigned to the "Prepare Work" activity. This created a strong bias towards this activity in the beginning resulting in lower precision rates of the "Prepare work" activity and lower recall rates of the "Drink water" activity.

a) Simulated data

Activity	Precision (%)	Recall (%)	Accuracy (%)
Drink water	66.3	62.5	86.8
Prepare cereals	95.1	96.6	94.4
Prepare curd cheese	63.8	46.5	62.8
Clean table cereals	87.9	64.7	94.0
Clean table curd cheese	45.2	44.7	89.5
Prepare work	44.6	68.0	92.6

b) Real world data

Activity	Precision (%)	Recall (%)	Accuracy (%)
Drink water	35.9	37.0	76.4
Prepare cereals	51.9	67.5	62.9
Prepare curd cheese	34.8	25.0	63.0
Clean table cereals	68.4	23.2	82.3
Clean table curd cheese	85.8	34.1	84.9
Prepare work	63.4	91.3	92.6

Table 7.5.: Average precision and recall for 12 experiments of the simulated and real data using only locations with HHMMs.

The values indicate that using our approach, we can distinguish between the different activities. Again, if it comes to the recognition of the different food-preparing and cleaning activities, distinction between them is rather difficult due to their similarity which seems to be a bigger problem in the noisy real-data than in simulation. To get an impression if our the approach is able to reliably recognize the different categories of activities, we performed another experiment using the real data. We merged the "prepare cereals" and "prepare curd cheese" activities into one "prepare food" activity as well as both clean table plans into one. Again, we calculated precision, recall and accuracy for this experiment, which resulted in the values shown in Table 7.6.

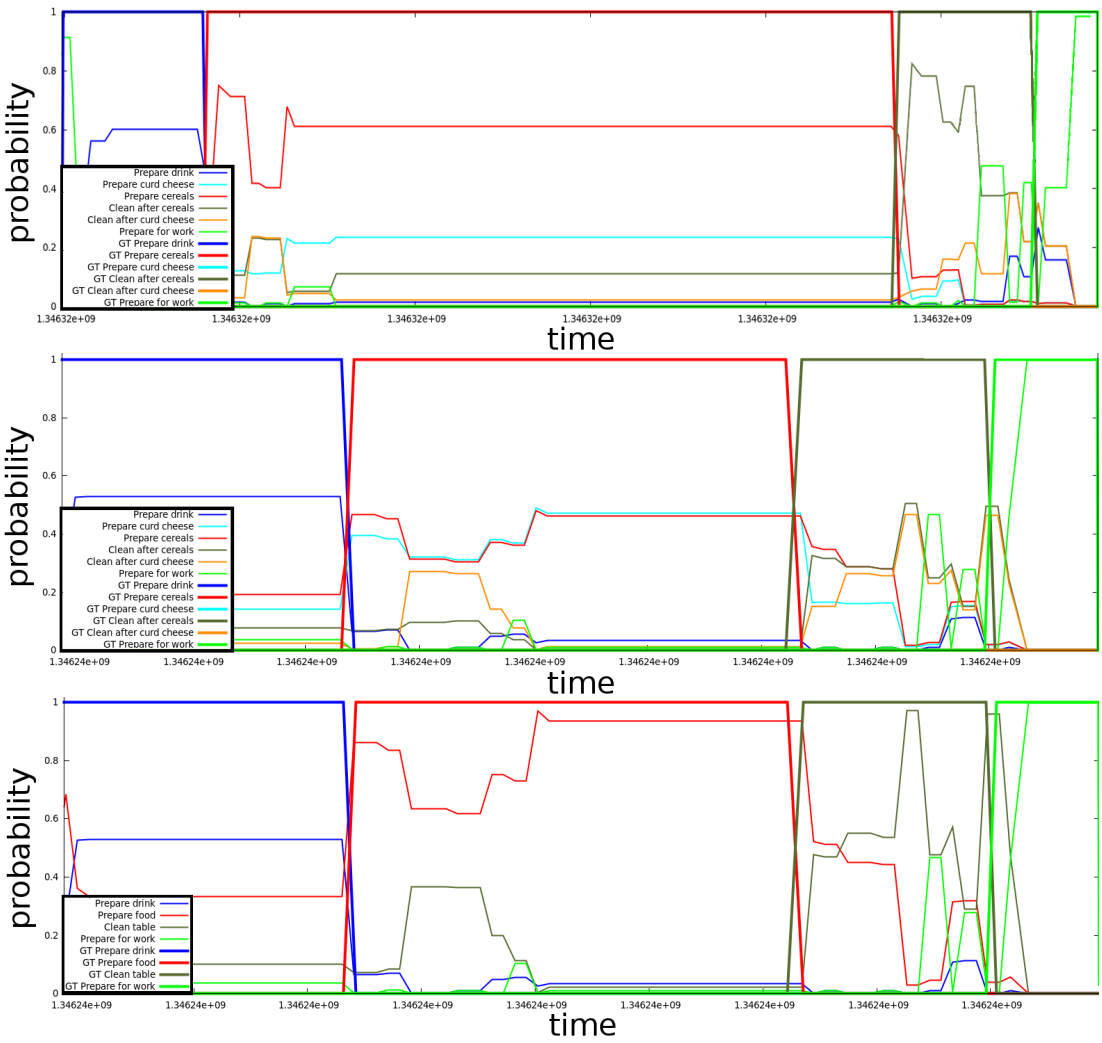


Figure 7.5.: The upper picture shows probabilities of the activities over time as well as the ground-truth of one morning routine of the simulated dataset using the online activity recognition. In the middle picture, the same plan probabilities are illustrated for the real data. Here, the activity recognition has a hard time distinguishing between the different food preparing and cleaning tasks due to their similarity. The lower picture shows plan probabilities of the same morning-routine as the middle picture, but in this case, "Prepare cereals" and "Prepare curd cheese" are merged to a "Prepare food" activity and the two clean-table activities have been merged to a single one.

A plot of plan probabilities over time for the simulated data, real data and real data with merged "prepare food" and "clean table" plans can be seen in Figure 7.5. The results suggest that most of the time, we can draw conclusions about

Activity	Precision (%)	Recall (%)	Accuracy (%)
Drink water	64.0	61.6	87.0
Prepare food	69.4	68.7	73.8
Clean table	49.2	79.5	79.9
Prepare work	90.6	48.5	94.4

Table 7.6.: Average precision, recall and accuracy for 12 experiments of the real data using only locations with HHMMs and combined plans.

different human activities using only CDSRs. Only the "Drink water" activity is not recognized at all in some cases. We think this effect is due to being the first activity the participant performed, thus only few observations are given to the HHMM. Since we initialize the probabilities of all of our states of the HHMM uniformly, it has a hard time finding the correct activity at the beginning when only one or two observations are available. To improve recognition rates at the beginning, one could think of biasing the initial state distribution (which currently is initialized uniformly) of the HHMM for some activities since, for instance, a human will most likely have breakfast and *afterwards* clean the table not the other way around.

Another way to increase recognition rates is the inclusion of object detections. Since one of our goals is to avoid equipping the environment extensively with sensors, RFID tags, etc., we investigated the use of only few object detections that we use to bias our recognition rates by only observing which objects appear in an activity at all and which don't. Therefore, we penalize the probability of an activity by a factor of 0.5 if we detect an object that is not being used in the activity and we reward the probability of an activity by a factor of 0.5 if a detected object is used in the activity. We decided to use this rather simple approach as a proof-of-concept, but there are also more elaborate systems based on sequences of object detections, for example by Buettner et al. [10, 61, 11]. We classified a detected object as "used by the human" when it was in reach of the human and changed its position. Precision, recall and accuracy for the simulated dataset with partial object detections can be seen in Table 7.7. Here, out of the 25 object interactions of the human user, on average 15 were detected. Using those 60.3 % of detected object interactions already allows us to bias the activity recognition and increase the system performance (on average: Accuracy of 94.1 %). Unfortu-

nately, for the real data, the inclusion of object detections we recorded with visual markers and a Kinect did not significantly improve our results due to rather bad recognition of the objects. Because of the small size of our markers and long distances between objects and Kinect, we obtained many wrong object detections. Also, objects that were standing still were wrongly classified as moving due high sensor noise. The accuracy values when using real data with the marker-based object detections ranged between 57.5 % and 91.0 % with an average of 76.41 %.

Activity	Precision (%)	Recall (%)	Accuracy (%)
Drink water	90.2	73.5	94.2
Prepare cereals	96.9	98.1	96.4
Prepare curd cheese	97.9	88.4	91.0
Clean table cereals	84.1	73.4	96.2
Clean table curd cheese	61.7	48.2	91.3
Prepare work	68.8	86.5	95.7

Table 7.7.: Average precision and recall for 12 experiments of the simulated data with partial object detections using HHMMs.

7.2.3 Learning Transition Probabilities Using the Baum-Welch Algorithm

In the experiments so far, we used the ground truth data to manually generate activity models that were used for the HHMM-based activity recognition. But it is also possible to directly learn the parameters of the HHMM only from observations. This is commonly done using the Baum-Welch Algorithm [73] as explained in section 2.2.3 in chapter 2. The algorithm uses an initial guess of the state-transition probabilities and the emission probabilities as input and employs a variant of Expectation Maximization [57] to calculate a maximum likelihood estimate of the state-transition model and the sensor model. If no prior data is given for the initial guess, mostly the initial probabilities are chosen to be distributed uniformly. We performed one experiment where we learned the parameters of the HHMM using 13 experiments of the Morning Routine Dataset as training data and one experiment for a qualitative evaluation. As input for learning, we use the observations, which, in our case, represent CDSRs at which

the observed person is standing still. We obtain those CDSRs by querying the spatial model for the most likely CDSR at the position where the person is standing. We assume the robot to be provided with the current activity that the person is performing at every time but it does not know about the order in which CDSRs are visited during task execution. We decided to use this simplification since the detections of transitions between activities is a non-trivial task and is not the focus of this thesis. We use emission probabilities estimated from the spatial model as explained in section 5.2.2 as initial guess for the sensor model and initialize the state transitions uniformly since we assume our robot to have no prior knowledge about the activities at the beginning. We iteratively apply the steps of the Baum-Welch algorithm until it converges and use the updated HHMM to perform activity recognition on the test data. A qualitative evaluation of the experiment is shown in Figure 7.6. We see that activity recognition for the all activities except for the "Prepare Cereals" activity performs really well. However, "Prepare Cereals" is wrongly classified as "Prepare Curd Cheese" and "Prepare Drink". This can on the one hand be explained by the similarity of "Prepare Curd Cheese" and "Prepare Cereals" as already mentioned in the experiments before. On the other hand, one should note that commonly, large sets of training data are needed by learning approaches to enable them to perform well. The Morning Routine Dataset we used in our experiments only consists out of 14 instances of data and when trying to learn a model for "Prepare Cereals", we can even only use data of the 8 days where the participant had cereals for breakfast. Splitting these data into training and testing data limits the training data for the "Prepare Cereals" activity to 7 days. In the case of "Prepare Curd Cheese" (5 days in total), it even looks worse and for "Prepare Bread", we only have two instances of data. However, this exemplary, qualitative evaluation proves that generally, the learning of the parameters of an HHMM for activity recognition is possible, but one would need a larger amount of data to obtain good results. A domestic service robot could, for instance, could collect such data when observing a person for a long time.

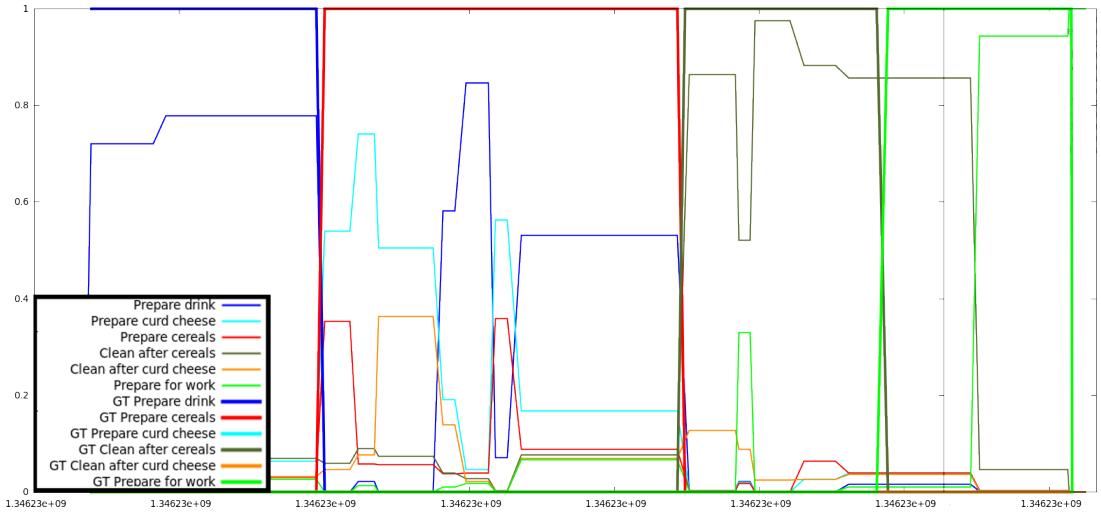


Figure 7.6.: The picture shows results of HHMM based activity recognition of the test data using parameters that were learned by the application of the Baum-Welch Algorithm.

7.3 SPRAM

Most of our experiments show that the different food-preparing and cleaning-tasks can only be distinguished with a certain amount of uncertainty. Given these uncertain results from activity recognition, predictions about future locations can be generated as visualized in Figure 7.7. We use the results of the Forward-Backward Algorithm (see section 2.2.2) that is used for activity recognition. The Forward-Backward algorithm provides us with a state distribution over all states of the HHMM, meaning not only the internal states, but also the production states, which represent CDSRs. Given this state distribution, we can use the Forward Step of the algorithm to compute predictions about likely next locations of the observed person as explained in section 5.3.2. Some of the locations (in our case the table) can be predicted very reliably resulting in only one gaussian with a high weight as shown in the second picture of the first row in Figure 7.7. The figure shows a weighted spatial model representing a probability distribution about which CDSRs the person is likely to visit next based on the results of activity recognition. In other cases, our system was not too certain which location would be visited next resulting in several gaussians with similar weights as illustrated in the first image of the first row in Figure 7.7. However, in this case, we still have a weighted spatial model which gives information about

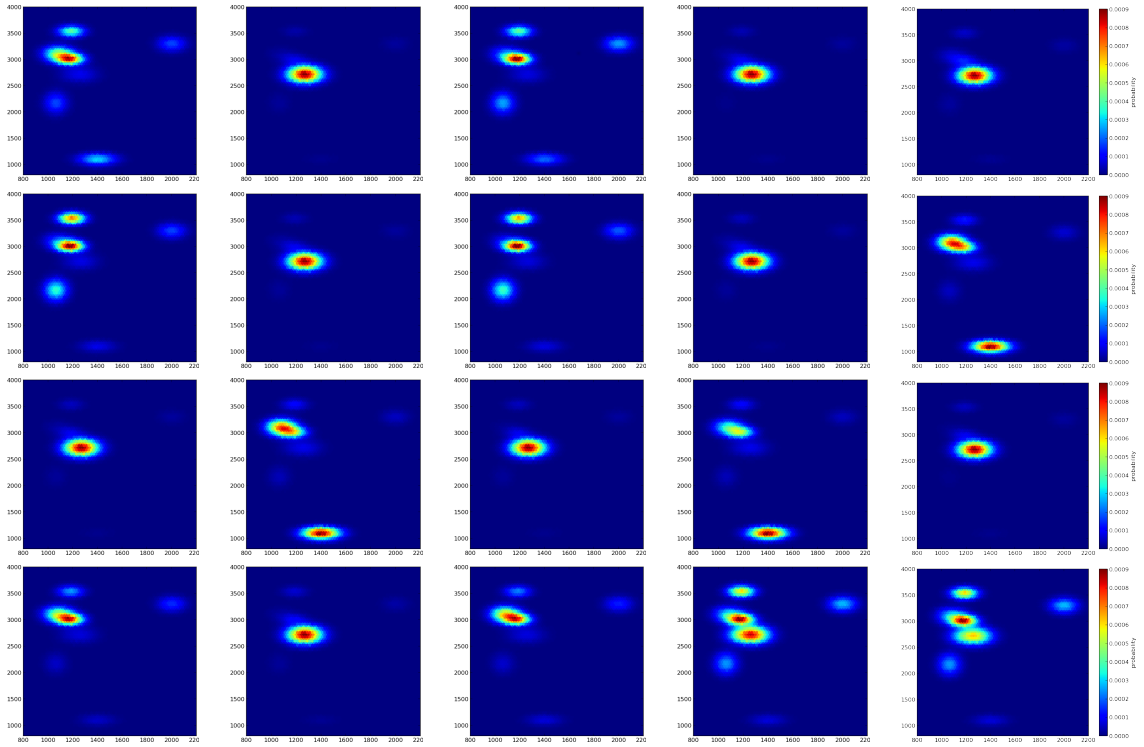


Figure 7.7.: The spatial model displays probabilities about where the human is expected to go next. These 20 images show the predictions of likely next locations of the person each time, a new observation is added to the HHMM.

several locations that are possibly visited in near future. The advantage of such a weighted spatial model is the fact that even in situations when activity recognition can only provide uncertain results, a robot has an impression about which regions are more likely to be visited by the person than others. In the case of a household robot, the robot could try to avoid these regions in case it does not want to disturb the human or search those locations if it is looking for a person.

An exemplary, qualitative evaluation is presented in Figure 7.8. Here, we see the probabilities of predicted locations plotted over time for one day of the Morning Routine Dataset. We can see that mostly, our approach can quite reliably predict a person going to the table, while most other location predictions have a lower probability of being visited. This can be explained by similarities between the different activities the person performed during his/her morning routine. In most of the activities, like table setting, table cleaning, etc, the person visits the

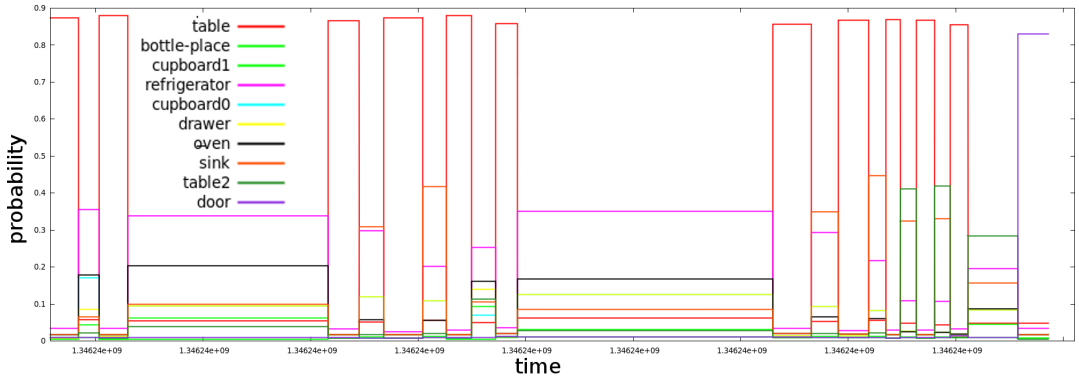


Figure 7.8.: This image shows likely next locations of a person during his/her morning routine according to SPRAM.

table after he/she visited another location. This way, our system exploits those similarities to generate reliable predictions even when activity recognition is not exactly sure which activity is actually performed.

Although, the advantage of the proposed approach is the fact that no unique prediction of one CDSR is needed to be useful, we still want to present a quantitative evaluation. The next table shows the correctness of the predictions that were generated by our system. A prediction is defined to be correct, if the observed person visits the predicted CDSR with the highest likelihood in his/her next step.

Day	Correct predictions (%)	Day	Correct predictions (%)
1	42	7	68
2	61	8	64
3	61	9	60
4	60	10	65
5	65	11	60
6	72	12	54

Table 7.8.: Correctly predicted next locations of the person averaged over each of the 12 experiments.

We can see that the correctness of the predictions ranges between 42 % and 72 % with an average of 61 %. While this might seem like a rather low value, one should keep in mind that those predictions were made based on unreliable data and even when a prediction in this table has been wrong, the robot still has its weighted spatial model.

7.4 Application: Expectations for a Household Robot

This section will demonstrate the application of the expectations framework described in chapter 3 in two simulated scenarios and one real-world scenario. The simulated scenario features a domestic robot assistant using a variety of different expectations to detect unexpected situations in an apartment. Furthermore, we use the Morning Routine Dataset described in chapter 6 to equip a domestic robot helper with expectations about human routine behavior and showcase the applicability of the expectations framework by enabling it to distinguish normal from unexpected behavior in a kitchen environment. The expectations are generated based on the integration of activity recognition into the expectations framework as explained in section 5.4

7.4.1 Simulated Human Apartment

In this simulated scenario, we demonstrate the applicability of the expectations framework explained in chapter 3 in different situations. We use 5 to 15 expectations grouped in different categories to estimate the normality of situations in a simulated apartment environment including a person and a domestic service robot as shown in Figure 7.9. The person is modeled using the avatar of the MORSE simulator that can be controlled like players in 3D computer games as explained in 7.1. The robot is equipped with an object detection sensor that returns the name of objects within the field of view of the robot and their positions. The robot is also able to detect if doors are open or closed as well as the positions of humans, while it is not aware of the identity of different persons. The robot is controlled using the ROS middleware and the "move_base"[4] package for 2D navigation.

[4]http://wiki.ros.org/move_base

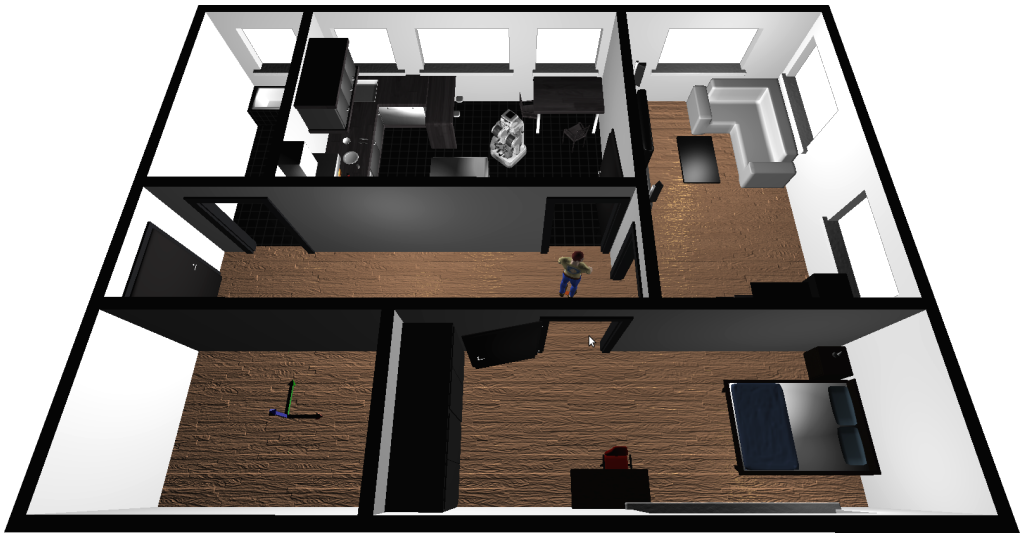


Figure 7.9.: A simulated apartment scenario including a person and a domestic service robot in the MORSE simulator.

7.4.1.1 Kitchen robot

For this scenario, we assume the robot to be located in the kitchen where the table has been set for breakfast as illustrated in Figure 7.10 on the left. Expectations about the objects on the table tell the robot that the objects are not expected to lie on the ground and the objects are not expected to move. For each object, we create one expectation to express that the object is not expected to move and another one modeling that the object is not expected to lie on the ground. The following objects were considered in this scenario: fork, knife, jam, hazelnut spread, cereals, bowl and plate. We furthermore generate an expectation modeling that the table is expected to stand still. We obtain 15 expectations in total that are continually validated every two seconds, returning an average normality value. A summary of the types of expectations used for this scenario along with their corresponding category (according to section 3.1.1) and weight in the normality tree is illustrated in table 7.9.

Expectation Type	Category	Weight
Object-on-floor	Logical Expectation	1
Object-movable	Logical Expectation	1

Table 7.9.: Expectation types used in the Kitchen Robot scenario.

Figure 7.10.: The kitchen robot scenario in the simulated apartment. Left: A simulated PR2 robot starts navigating away from the table. Middle: The robot inadvertently hits the table causing objects to fall to the ground. Right: The robot looks back, detecting more objects lying on the ground.

In our scenario, we assume the robot to just have finished setting the table and it is navigating towards the door when it inadvertently hits the table with its side. The hit causes some objects on the table to drop onto the kitchen floor as illustrated in Figure 7.10 in the middle. The detections of moving objects and objects lying on the ground immediately causes the average normality to drop. When the robot turns around at a later point in time, it detects more objects lying on the ground as shown in Figure 7.10 on the right. Figure 7.11 shows the average normality plotted over time.

As soon as the robot hits the table, the average normality decreases since it detects objects within its field of view moving unexpectedly, falling onto the floor. Since the robot is moving away from the table, not all objects that fell onto the floor are in the field of view of the robot's object detection sensor. After all objects stopped moving, the average normality is around 0.7. When the robot later turns around and detects more objects lying on the ground, it also recognizes that some of the objects have moved since their last detection, which leads to another drop of the average normality to a value around 0.4. It finally slightly increases since no objects are moving any more and converges at a value of around 0.5. The normalities in this scenario are a valuable indicator to the robot that an action of the robot could have caused a change in the normality of the situation and could serve as input for an Explanation Generator, which could use discrepancies between expectations and measured facts to create a new goal and ultimately react appropriately. Such an approach has, for instance, been proposed in the work of Klenk et al. [56].

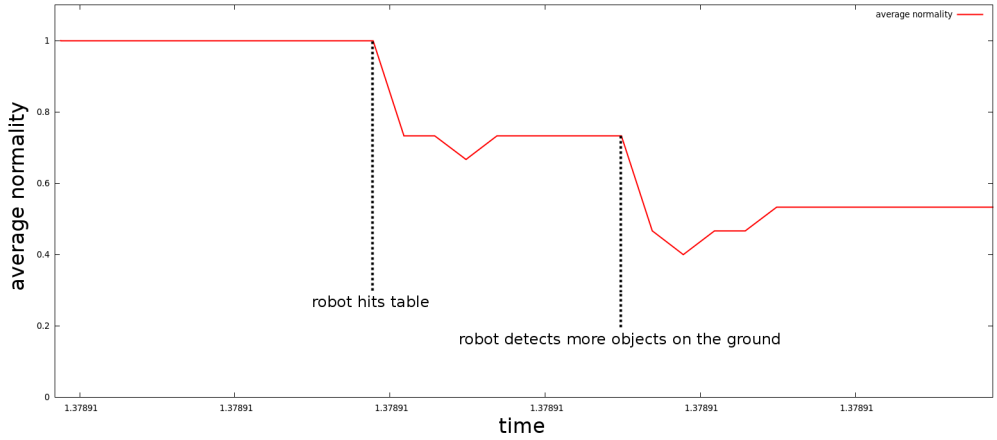


Figure 7.11.: Average normality of the simulated kitchen robot scenario plotted over time.

The same set of expectations could also enable a robot to detect damages caused by an earthquake or an infant throwing objects on the floor. Although, the robot may not be equipped with the appropriate sensors to detect the cause of an abnormal situation, it will at least be aware that something is unusual and it might have to take action.

7.4.1.2 Patrol Robot

The second scenario is located in the same apartment, where our household robot — while having no other duties over night — is guarding the apartment while the human is sleeping. It patrols between several locations, staying two seconds at each location, checking if everything is as expected. Therefore, five expectations are used: The TV is expected to be on the console in the living room and it is expected not to move. Humans are only expected to be in the bedroom or outside the house during nighttime and the entrance door of the apartment should be closed. We furthermore expect the navigation of the robot to function normally. This can, for instance, be of advantage if a burglar would try to block the robot. The robot navigation expectation is generated as a temporal expectation each time the robot starts moving to a new waypoint. It queries the navigation path planner for a new path plan and estimates the time to the location given the length of the path plan and the robot's average speed. It can then estimate

the expected time needed to reach the next waypoint and generate a temporal expectation. When a waypoint is reached, the expectation is removed until the robot navigation computes a new navigation plan for the next waypoint. This means that, while the robot is moving, five expectations are validated, but when the robot is standing still for a short time, only four expectations are active. Table 7.10 lists the different expectation types that were used in this scenario along with their categories (according to section 3.1.1) and weights in the normality tree.

Expectation Type	Category	Weight
Object-at-location	Logical Expectation	1
Object-movable	Logical Expectation	1
Human-at-location	Logical Expectation	1
Door-closed	Logical Expectation	1
Navigation-functional	Temporal expectation	1

Table 7.10.: Expectation types used in the Patrol Robot scenario.

We use the human avatar to simulate a burglar entering the apartment during the robot's patrol. He opens the entrance door and passes the hallway, entering the living room. There, he grabs the TV and leaves the apartment through the door, passing the hallway a second time. Figure 7.12 illustrates the simulated burglary. In Figure 7.13, the average normality during the burglary is plotted over time. As soon as the open entrance door is detected and a person is walking in the hallway, the average normality drops. The slight increase after the first drop is caused by the robot navigation expectation. When the average normality drops for the first time, only four expectations are active since the robot was temporarily standing still. When the robot starts moving again, a new navigation expectation is generated, causing a slight increase of the average normality. When the TV is detected to have moved and is outside of the living room, the average normality drops to its lowest value of around 0.2. However, the average normality never drops to 0 since the navigation of the robot is still working correctly. After the burglar has left the apartment, the average normality slightly increases again. Leaving the apartment with the TV, the burglar has to pass the robot really closely, which causes the robot's navigation to temporarily being blocked due to security reasons. Due to this blocking of the navigation, the robot is not able to reach its

Figure 7.12.: The patrol robot scenario. Left: The robot detects the open door and the burglar in the hallway. Middle: The robot detects the TV moving and outside in the living room. Right: The burglar has left the apartment with the TV.

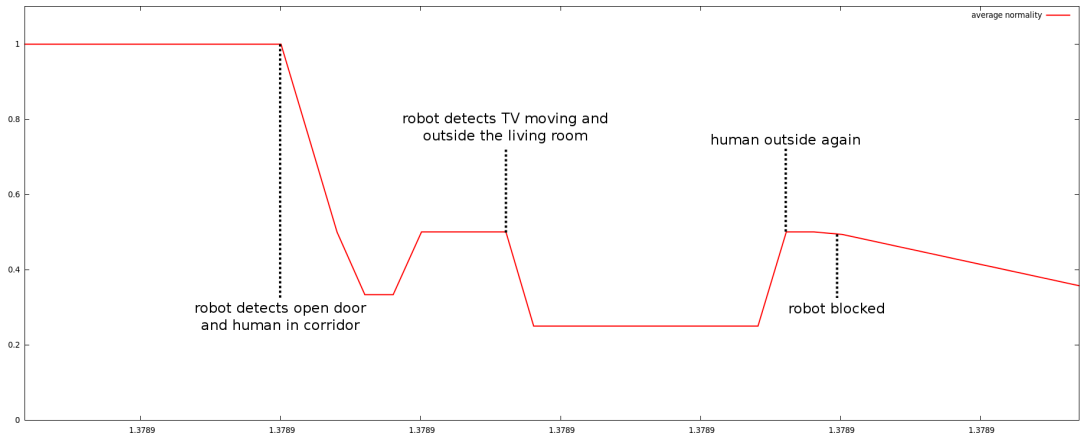


Figure 7.13.: Average normality of the patrol robot scenario plotted over time.

next waypoint in time, and the robot's temporal navigation expectation causes the average normality to decrease linearly towards the end.

In an integrated system, the robot should of course not just observe the situation until a burglar has left, but start to take action in the case of such a low average normality. An open question here is the threshold of starting a corrective action. Just noticing a person might not be enough evidence to call the police since the person might be another inhabitant coming home late. In this scenario, an adequate reaction of the robot could be trying to identify the intruder, notify the inhabitant of the flat or even calling the police at some point.

7.4.2 Scenario: Expectations about a Human Morning Routine

This last scenario will demonstrate the applicability of the expectations framework in a real world scenario using motion tracking data of the Morning Routine Dataset explained in chapter 6. It uses a semantically annotated map of the environment in a module that performs SPRAM as explained in chapter 5 based on models of human activities as explained in chapter 4. The SPRAM module enables us to detect typical human activities from the observation of human task performance and to predict future locations of the observed person as explained in section 5.3.2. The semantically annotated environment map is combined with learned models of locations where humans typically are located when picking up objects from cupboards, drawers and planes as explained in chapter 4. The activity models used for activity recognition were hand-coded in this experiment and include locations that a person visits during task performance as well as the durations that he/she commonly spends at the corresponding locations. However, such models of activities of daily living can also be generated automatically by observing motion tracking data as explained in chapter 4. The SPRAM module provides our robot with a probability distribution about which activity the human is likely to be executing as well as probabilities about which location the human is likely to visit next.

Given the limited data of the Kinect sensors, we used the following two expectations, grouped in a composite expectation that we call *human activity expectations*:

- *Location expectation:* A probabilistic expectation about the locations that the person is likely to visit next;
- *Duration expectation:* A temporal expectation about the duration that a human usually stays at certain locations.

Each time the observed person is detected to be standing still, we query our spatial model to which location the current position most likely corresponds. We then add an observation to the HHMM and update the location expectation and the duration expectation. The person is considered to be "standing still" when the center of mass of the person is moving less than 25 cm within 0.5 seconds (0.5

m/s). In this case, the SPRAM module returns a new probability distribution estimating probabilities for each location being visited next by the person. This probability distribution is used to generate a probabilistic expectation over likely next locations of the person. Given the durations that a person typically spends at certain locations, we can also create a temporal expectation about these durations when the person is standing still. We define the validation function of the temporal expectation to linearly decrease its normality value when the expected time at a location is exceeded. Both expectations generated so far are dynamically updated each time the person is standing at a new location. Table 7.11 shows the expectation types used in this scenario along with their categories (according to section 3.1.1) and weights in the normality tree.

Expectation Type	Category	Weight
Duration-at-location	Temporal Expectation	1
Next-location	Probabilistic Expectation	1

Table 7.11.: Expectation types used in the Morning Routine scenario..

In Figure 7.14, the estimated average normality over time is shown for three different situations: The green line represents a typical morning routine of the participant, mostly corresponding to the robot's expectations. In this case, the average normality is quite high (0.83 on average). In some cases, the normality drops to almost 0.6, which is due to uncertainties in the activity recognition. The blue dotted line shows the average normality of an experiment where the person starts performing his typical morning routine but then just stays sitting at the table instead of continuing his usual activity. In this case, the situation is classified as quite normal as long as the duration expectation does not exceed its expected duration. As the expected duration is exceeded, the temporal expectation causes the average normality to decrease linearly as time advances. One possible reason for the person not to move any more could be that he/she is sick or even unconscious, but maybe he/she just needs more time than usual due to any other reason. The red line illustrates the average normality when the robot observes a task that is not part of the morning routine and thus is not expected. In this case, the person is performing a cleaning task that was generated using the data of the TUM Kitchen Dataset 2. In this case, one can clearly see that

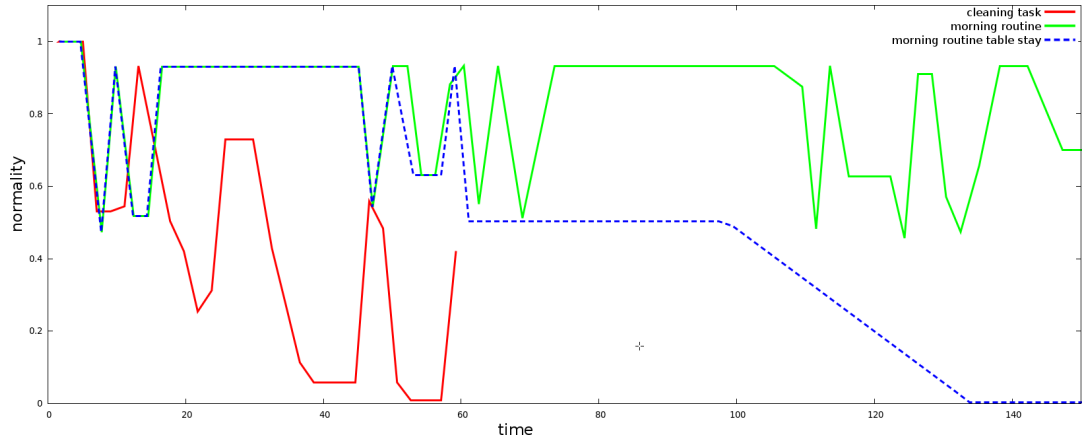


Figure 7.14.: Estimated average normality of three different situations over time. Green line: average normality during a typical observed morning routine; dotted blue line: average normality for an interrupted morning routine; red line: average normality when observing a cleaning task.

Day1	Day2	Day3	Day4	Day 5	Day 6	Day 7
0.76	**0.75**	**0.84**	**0.82**	**0.76**	**0.81**	**0.80**
Day8	Day9	Day10	Day11	Day12	Day13	Day14
0.75	**0.84**	**0.80**	**0.81**	**0.83**	**0.82**	**0.83**

Table 7.12.: Average normalities while the participant performed his morning routine using expectations about the next expected location and the duration the person stays maximally at each location. Averaging over all days, the overall normality of the observed situations is **0.80**.

the average normality is significantly lower most of the time when the person executed the unexpected cleaning task.

The average normality values averaged over time for all 14 days of the morning routine data set are illustrated in Table 7.12. The average normality of the cleaning task data (when expecting the morning routine) averaged over time is shown in Table 7.13. The tables indicate that in most cases only the two simple expectations about the predicted future location and the duration can distinguish between expected and unexpected behavior in a real-world setting. For the last example, we used data of the nine different persons that were performing a cleaning task in the TUM Kitchen Dataset. We are aware that the use of data of

Person	1	2	3	4	5	6	7	8	9
Avg. normality	**0.40**	**0.45**	**0.53**	**0.49**	**0.48**	**0.57**	**0.71**	**0.51**	**0.46**

Table 7.13.: Average normalities while the the nine participants of the TUM Kitchen Dataset 2 performed a cleaning task which is not expected during the morning routine. Averaging over all persons, the overall normality of the observed situations is **0.51**.

different persons is not optimal in this case, but we think this still provides an indication about the way, our system reacts to activities that are not expected.

7.5 Conclusion

In this chapter, we evaluated the techniques that were proposed in this thesis to enable a robot to explicitly account for persons and distinguish normal from unexpected situations. We therefore investigated the use of the times that persons spend at specific locations during the execution of different activities as features for activity recognition. We found that solely the use of such durations does not provide sufficient evidence to enable a reliable distinction between activities. As a second feature for activity recognition, we examined the order in which specific locations were visited by persons during task execution and found that the use of these patterns of locations can enable a robot to distinguish between activities. We identified the use of HHMMs to be superior to using STPRs and the GLS metrics, especially in confined spaces. We showed, how HHMMs for activity recognition can be learned from training data and set up an activity recognition system that performs live-activity recognition in an experimental kitchen using simulation and real motion tracking data from Kinect sensors. We evaluated the quality of our system and showed how recognition rates can be improved by varying the detail of activity models and the consideration of object detections. We thereby found that when integrating object detections, the activity recognition heavily relies on the quality of the object detection system, thus only a reliable object detection can improve recognition results. We furthermore showed how an extension of the activity recognition approach can enable a robot to estimate future locations of persons based on models of common activities of persons and we assessed the quality of such predictions. Finally, we proposed scenarios where the

expectations framework proposed in chapter 3 can be used to enable a robot to detect unexpected situations by combining and validating various expectations in two simulated scenarios. We concluded with a real-world scenario where we demonstrated an integrated system that uses the expectations framework in combination with predicted locations of the activity recognition approach to estimate the normality of a situation while observing a person's morning routine.

Chapter 8

Discussion

In this chapter, we discuss the advantages and limitations of the work introduced in this thesis. We will therefore highlight the challenges we met during the implementation of the approaches described in chapter 3, 4 and 5 and the insights we gained by the evaluation of the approaches in chapter 7. We will furthermore propose possible future applications of our work which showcase possibilities for generalization.

8.1 Expectations Framework

The expectations framework described in chapter 3 is designed to offer a general, modular and knowledge-based approach to assess the normality of situations by validating different expectations. This capability can be useful for an autonomous robot to increase its situational awareness and detect and adequately react upon unexpected events.

In the current implementation, we make no assumptions about the knowledge used for the generation of expectations, except that it can be used to quantify the normality of a situation. The framework sets a high value on modularity, making it extensible to arbitrary types of expectations, depending on the needs of client applications. By the combination of different expectations in the normality tree, a robot is provided with (limited) means of diagnosing the cause of surprise. This can be achieved by traversing the normality tree and checking which of its expectations has not been fulfilled, thus giving the robot a first hint towards possible reactions. Using knowledge that the robot may also need for its decision making

or state estimation, like the activity executed by a person or storage locations of objects, the robot can itself detect unexpected situations, freeing engineers from considering every situation that the robot might encounter.

8.1.1 Generation of Expectations

An open question is how to generate expectations from available data. The number and types of expectations a robot or technical system needs to identify abnormal situations largely depends on the desired application. While, for instance, a vacuum robot can benefit from expectations about typical locations of persons at different points in time, expectations about storage locations of tableware will be of no interest for such a robot. In contrast, a more elaborate household robot that is designed to assist a person with activities of daily living will find such information useful. In most of the application scenarios that we presented in chapter 7, we partly manually define expectations that seem useful for the specific situation. We also make use of models about the expected behavior of a person by learning models about typical routine activities from motion tracking data. In section 7.4.2, we demonstrated how to use such learned models of human task execution in combination with an activity recognition module to dynamically generate and validate expectations about human task performance.

It is also possible to infer expectations from common sense knowledge. Tenorth et al. [85], for instance, use Knowledge-Linked Semantic Object Maps in the knowledge processing system KnowRob [84] to infer likely storage locations of objects in a kitchen and model those as expectations. The following example query uses common-sense knowledge about objects stored in ontologies and a semantic map of the environment (both included in the KnowRob Knowledge Processing System) to infer the most likely storage location for a piece of butter.

```
?- storagePlaceFor(Place, map_obj:'butter1').
Place = 'http://ias.cs.tum.edu/kb/knowrob.owl#Refrigerator67';
```

The query returns the object *Refrigerator67* as most likely storage location which is a furniture object that can be located using the semantic map. Such information can easily be modeled as a logical expectation using our expectations-framework

by creating an "object-in-container" expectation where the validation method queries KnowRob for the dimensions of the container called *Refrigerator67* in our example and checks if the object is inside the container. The bounding boxes of the container are stored in the semantic map of the environment in KnowRob. Being embedded in the CRAM plan language [5], our expectations framework already comes with an interface to the KnowRob System. The validation should be triggered as soon as the object in question is detected. Thus, a the robot would for example consider butter stored in a refrigerator as "normal" and butter stored in an oven as "not normal".

Similar queries can even be made for categories of objects or for their attributes. The following query returns likely storage locations for all known tableware objects:

```
?- owl_individual_of(Object, knowrob:'HandTool'),
   storagePlaceFor(StoragePlace,Object).
A = 'http://ias.cs.tum.edu/kb/ccrl2_map_objects.owl#knife1',
StoragePlace = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer1' ;
A = 'http://ias.cs.tum.edu/kb/ccrl2_map_objects.owl#knife1',
StoragePlace = 'http://ias.cs.tum.edu/kb/knowrob.owl#Drawer103' ;
(...)
```

Here, likely storage locations for tableware like a knife or spoon are all drawers in the environment. The inference does not provide us with probabilities for any of the storage locations, so one could think of modeling such information as logical expectations using a logical "or":

$$location_{knife1} = Drawer1 \;||\; Drawer103 \;||\; ...$$

But one could as well model such information as one probabilistic expectation with a probability distribution that is uniformly distributed directly after the inference process. Although, the inference process does not provide us with probabilities in the first place, a representation in a probabilistic way has the advantage that the expectations can be updated when new information is obtained. Given the example above, a robot observing that *knife1* is located in *Drawer103*, could update its expectation about knife objects, increasing the probability of

knives being in *Drawer103* while decreasing the others. A robot could even infer likely storage locations for other cutlery objects like spoons or forks and update corresponding expectations, since humans tend to store cutlery in the same container. Such information can for instance be obtained using semantic object similarity measures like the WUP similarity[1]. This functionality is also available in KnowRob, enabling queries for likely storage locations of objects based on semantically similar objects [76], assuming prior information about storage locations is available. It can, for instance, be inferred that a likely storage location for a fork is a drawer in which spoons and knifes are stored. As Beetz et al. [4] showed, one can also query for the likely storage locations of objects that have certain properties like the typical location of objects that are perishable:

```
?- owl_individual_of(A, knowrob:'Perishable'),
   storagePlaceFor(Place, A).
A = 'http://ias.cs.tum.edu/kb/ccrl2_map_objects.owl#butter1',
Place = 'http://ias.cs.tum.edu/kb/knowrob.owl#Refrigerator67';
A = 'http://ias.cs.tum.edu/kb/ccrl2_map_objects.owl#buttermilk1',
Place = 'http://ias.cs.tum.edu/kb/knowrob.owl#Refrigerator67';
A = 'http://ias.cs.tum.edu/kb/ccrl2_map_objects.owl#cheese1',
Place = 'http://ias.cs.tum.edu/kb/knowrob.owl#Refrigerator67';
(...)
```

This query equips the robot with the knowledge that perishable goods like the butter, buttermilk or cheese are typically stored in a refrigerator and, in the specific environment in our example, in Refrigerator67, which can be found in the semantic map. This information can be translated into expectations that a robot can validate when it detects objects. So our expectations framework enables a robot to use already available common-sense knowledge to create expectations about objects, thereby increasing its situational awareness and enabling it to detect anomalies like perishable goods that a person forgot to put back into the fridge. Furthermore, when the robot uses an AI planner, the effects of the planning operators can directly be used as expectations about the outcome of the action.

[1] http://search.cpan.org/dist/WordNet-Similarity/lib/WordNet/Similarity/wup.pm

8.1.2 Weighting of Expectations

Another difficulty that we did not consider so far lies in the weighting of different pieces of information. Even though, we used the average of different expectation values, probabilistic expectations will implicitly have a lower weight if the probability distribution over possible outcomes is no degenerate distribution at one specific outcome. Let us consider the example of a robot observing a person that performs an activity and trying to predict future locations of that person.

Given its observations $o_{1:n}$ of locations visited by the person so far, the robot might generate the following discrete probability distribution $P(l_i^{t+1}|o_{1:n})$ about likely next locations l_i^{t+1} of the person:

$$P(l_i^{t+1}|o_{1:n}) = \{\text{refrigerator: } 0.75,\ \text{oven: } 0.15,\ \text{drawer: } 0.05,\ \text{table: } 0.05\}$$

The robot predicts the refrigerator as the most likely next location of the person (with probability 0.75) and models this information in a probabilistic expectation. Given, the person will actually go to the refrigerator next, the validation of the expectation will return a normality of 0.75 % using the validation method described in section 3.1.1.3. Although, the expectation has been perfectly fulfilled, the normality value is still lower than a value returned by the validation of a logical expectation. In another case, the robot might not be quite certain about the next location of the observed person e.g. due to uncertainty in its activity recognition or noisy sensor readings. The probability distribution of the expectation about likely next locations in this case might look as follows:

$$P(l_i^{t+1}|o_{1:n}) = \{\text{refrigerator: } 0.4,\ \text{oven: } 0.4,\ \text{drawer: } 0.15,\ \text{table: } 0.05\}$$

The robot is quite certain that the person will either go to the refrigerator or the oven next, with equal probabilities assigned to both locations and validation will return a normality of 0.4 if the person will go to one of the two most likely locations. The robot's expectation has basically been fulfilled, but the validation of the expectation returns a rather low normality value of 0.4 due to uncertainties in the predictions. Generally, this means we make the normality of a situation dependent on the uncertainty of our predictions when using probabilistic expecta-

tions. This can be problematic, especially when such expectations are combined with other expectations that do not include any models of uncertainty such as logical expectations. Thus, the comparability of the expectations is affected. One possible solutions towards this problem could be a weighting of probabilistic expectations in proportion to the uncertainty, so expectations that have high uncertainties have a lower impact on the overall normality, while expectations that have no uncertainty have the same impact on the overall normality as logical expectations (and thus are equally weighted).

8.1.3 Combination of Normalities

So far, we chose the average as combination function for normality values of expectations, which can be justified by the tallying heuristic of humans, where weights are ignored for decision making [22]. However, the choice of an appropriate combination function in general is an open question and it is unclear whether our proposed representation of the normality tree can cover all necessary interactions of the normality values. As already indicated in the last subsection, in some cases weights may be necessary and in other situations, non-linear combination methods may be appropriate.

Consider, for instance, situations including expectations that imply a state of danger, like the detection of a fire or smoke. In this case, a robot should immediately raise an alarm or take appropriate actions, but just averaging over all normalities might not affect the overall normality strong enough. A possible way to achieve such behavior is to not only monitor the average normality, but also single critical values like the expectations about fires or smoke in the environment.

8.1.4 Integration into Cognitive System

The integration of our expectation framework into a complete cognitive system is a crucial question that was only mentioned briefly so far. On the one hand, many single expectations are based on the input from the robot's perception,

respectively the knowledge-base in which percepts can be stored. Considering, for instance, expectations about the locations of a person, the robot will need to know at which locations the person has been observed; analogously expectations about likely storage locations of objects need information about object detections. But expectations can also serve as an input to perception. An unmet expectation can also simply be caused by a wrong interpretation of the world. Thus, the expectations framework can serve as a bridge between perception and reasoning by causing the perception system to re-evaluate the situation.

Similarly, an adequate action might have to be taken when the situation strays too far away from the expected. Ideally, knowledge sources that indicate strong deviations of normality could also provide means for determining the severity of the unexpected situation as well as possible reactions. Action selection and planning methods can be invoked to find the most appropriate reaction. The action that the robot takes can then again interact with its perception since in many cases the best reaction might be a clarification of the situation by acquiring more detailed sensor information or communicating with a person.

Our proposed expectations framework constitutes a first step towards a general handling of failures and unusual situations by autonomous robots. Being an extension to the CRAM plan language, it can easily be integrated with existing methods of failure recognition and handling, which offer a robust way of handling frequent failures and dangerous situations. Our knowledge-based approach increases the situational awareness of autonomous robots and enables it to cover a large variety of situations that an engineer might not think of when designing a robot or for which specific coding would be too costly. In this thesis, we demonstrated how the expectations framework can also be used to seamlessly integrate expectations about human task performance into the contextual awareness of a robot, thus bringing service robots one step further towards the goal of being aware of human presence and behavior.

8.2 Human Activity Models

In chapter 4, we illustrated how we can use Context Dependent Spatial Regions (CDSRs) to generate models of human activities without explicitly modeling ac-

tions but by using action related context of spatial regions instead. Such models enable us to perform a cheap and simple activity recognition without the need for complex action recognition systems or object detections, which will be specifically interesting for low cost robots in domestic environments. However, these advantages do not come for free and there are several points that should be discussed.

8.2.1 Spatial Model Generation

When generating the spatial model, we cluster the 2D positions of a person using an Expectation Maximization based clustering method. The advantage of such a clustering is that, in contrast to a simpler nearest neighbor clustering, the clusters can be of different sizes and resulting clusters can be easily returned as two dimensional gaussian distributions, which we directly use to describe CDSRs. However, such a clustering has the disadvantage that the number of clusters has to be known beforehand. We argue that in our application scenario, we can get the number of clusters from prior information about the objects by querying their storage location but such information might not always be available. In this case, one has to use other clustering techniques that are not subject to this restriction such as variants of density-based clustering algorithms [18, 75].

Anther restriction of the current generation of the spatial model is the fact that the relation of the locations to the furniture objects has to be unique. For the table-setting activities used in the experiments in section 7.2.1.2, this relation can easily be obtained by just querying the semantic map for the object storage location that is closest to the mean of the current gaussian. But in other, more complex environments, simply the smallest distance of the positions towards the containers might not be enough to identify the correct container object or supporting plane. There might be several containers or objects in reach of the human - like cupboards that are mounted above a kitchenette -, so the relation of the location to an storage location might not be unique. Possible solutions to this problem can either be the inclusion of more data to detect the furniture object that the human is interacting with. Beetz et al. [7], for instance, use full body motion tracking data and identify movements of the arms and hands that describe the opening of cup-

boards and drawers or pick- and place actions. Also, the use of contact sensors in cupboards or drawers would be possible to guarantee a correct and unique detection of the correct furniture object. Another possibility of accounting for ambiguous relations of locations towards furniture objects is to relax the property that this relation has to be unique. One location could as well have several furniture objects assigned, possibly including probabilities about the likelihood of a human interaction with each container from the current position.

8.2.2 Transferring the Spatial Model to Other Environments

When transferring a spatial model to other environments, we assume the human locations to be dependent on the relative positions towards the opening side of the container and the position of the hinge. This enables a robot to get an impression about where to expect a person when he/she interacts with furniture objects. But the relative position of the human towards container objects like drawer or cupboards and especially the relative position towards planes when performing pick- and place actions depends on other factors as well. Especially when objects are picked up from supporting planes, a person's location is also influenced by furniture objects that limit the reachability of the object. To additionally include such information, one would have to perform a sophisticated analysis of the environment structure. Also, the direction from which a person approaches an object storage location as well as the direction the person will go after picking up or putting down an object seems to be of relevance. While information about the approaching direction can easily be obtained, information about the direction, that the human heads to after the object interaction is harder to gather since it requires a projection into the future. However, the transfer of the spatial model as explained in section 4.1.2 can offer a robot a first impression that can already be used to distinguish between different activities as we showed in chapter 7. Also, it can further be refined by learning from more observations of task execution or the inclusion of more information about the environment and the walking direction of the person.

8.2.3 Comparison of STPRs

When comparing STPRs, we evaluated the use of durations that a person typically spends at certain locations as feature as well as the order in which certain locations are visited during task execution. This comparison can be seen as a simple kind of activity recognition and the evaluation of the use of durations as features proved to be rather unreliable in our experiments as shown in section 7.2.1.1. The use of patterns of visited locations in combination with the Generalized Levenshtein Similarity (GLS) worked well as long as the detection of visited locations of a person is reliable, thus offering a fast and simple way of distinguishing activities. However, in some cases, the location detection might not be able to reliably detect the CDSR where a person is standing or it will possibly detect a wrong location. This problem arises when furniture objects are located close to each other as in the second experiment in section 7.2.1.2. To be able to cope with unreliable location detections, a more elaborate, probabilistic approach should be used to enable the comparison of activities while accounting for uncertainties in the detection of locations. In section 7.2.1.3, we showed how to overcome this problem in an approach for activity recognition based on Hidden Markov Models.

8.3 Activity Recognition

In chapter 5, we introduced an approach for activity recognition and monitoring based on CDSRs, a semantic environment map and motion tracking data. The proposed approach enables a robot to make assumptions about human task execution and generate predictions about likely next locations of a person during task performance. The approach is designed to work with low cost sensors like a Microsoft Kinect in spatially limited environments and the evaluation proved the applicability in simulated and real-world settings. However, we made certain assumptions during the design of our system and the most important of which will be discussed in the following.

8.3.1 Occlusions

Our approach does not explicitly handle occlusions. Since it only depends on the two dimensional position of a person according to his/her center of mass, the occlusion of single body parts of the tracked person it not a big issue as long as not too much of the body is occluded. For our experiments with real-world data in chapter 7, we used a motion tracker that does not specifically include occluded body parts into it's model. Nevertheless, we showed that our approach is robust even though in some cases some body parts of the tracked person are occluded and even sometimes the motion tracker looses track of the person. This is one advantage of the use CDSRs for activity recognition instead of full-body motion tracking data or sequences of object detections. To further improve tracking results, one can think of using a motion tracker the specifically models occlusions as, for instance proposed by Buys et al. [12].

8.3.2 HHMMs as Activity Models

The use of HHMMs as models for activities in the activity recognition framework provides us with means of modeling variations in subtasks of single activities as well as uncertainties in the observations. However, in contrast to STPRs, the fully automated generation of such models is still an issue, especially when a person performs activities in parallel. In such a case, a robot observing such activities would not be able to know if e.g. drinking a glass of water is part of a cleaning task or if it is to be considered an own activity. It is even hard to detect when one specific activity has ended an another activity is starting. However, such decisions are also treated very differently by humans and there is no correct or incorrect way of modeling them. Variable subtasks during an activity can be modeled with our HHMM by providing different transition probabilities from one CDSR belonging to a specific sub-task to another one. In chapter 7, we gave an example on how such an HHMM can be generated and how the most likely set of parameters can be learned from observations. To be able to learn the parameters, we assumed to know which activity is currently being observed. This restriction is due to the fact that the detection of transitions between activities and thus the

hierarchical topology of a our HHMM cannot be generated autonomously in the current implementation. But once the HHMM is generated, we showed that such a model can successfully be used to perform activity recognition using models generated from prior knowledge and with learned parameters.

8.3.3 Action Detection

One initial assumption for our system is the ability to detect locations where a person is standing still for a short time while performing actions. In our current system, we therefore use a constant monitoring of the current moving speed of the person and additionally apply simple motion patterns. The motion patterns consist of movements where the observed person is moving towards a furniture object, spends a little while at a furniture object and moves away again. We assume such a motion model to be a good indicator for a person performing a pick- and place action at the corresponding furniture object. However, such a motion might not be general enough for all pick- and place actions in every environment. Although, the following did not happen during our experiments, a person might, for instance, grab objects while passing a table. Our rather simplistic preprocessing might not be able to detect such situations (especially when it comes to pick- and place actions from planes). One possibility to improve this detection would be the employment of more sophisticated motion patterns like the Qualitative Trajectory Calculus [23]. Of course, the inclusion of object detections can also widely improve the recognition of locations in which object interactions occur as we showed in section 7.2.2. But a reliable object detection system would mean a huge effort and possibly equipping the environment with loads of sensors [60] (which we wanted to avoid in the first place). For interactions with containers like drawers or cupboards, the use of magnetic sensors in the doors of the containers is widely used [82, 25] but it would also require the environment to be equipped with those sensors before usage.

So far, we only considered CDSRs that are related to pick- and place actions. When it comes to the detection of other actions than pick- and place, one might also have to use other motion patterns and sensor modalities. Fogarty et al. [20], for instance, use water flow sensors in the basement of an apartment to perform

low-cost and unobtrusive activity recognition. Such sensors could also be used to extend our approach to the detection of CDSRs that are related to activities using water like doing the dishes.

8.3.4 Monitoring

In the monitoring part in section 5.3.1, we briefly mentioned that we suggest to monitor activities that have a high probability of being executed. However, we cannot provide a general answer about the quantity of these probabilities. As our robot is ultimately intended to react upon uncertain information from activity recognition, our system has to run in real-time, which is the limiting factor for the amount of activities that can be monitored. As we use a very simple approach for monitoring, we did not run in any problems when monitoring up to eight activities at once, but the scalability of the approach towards more activities and/or more elaborate monitoring techniques will remain a question for future research. The monitoring technique that was used for our system only considered the amount of times that certain CDSRs were visited, not even considering their order. This technique is rather a proof-of-concept and can be replaced with more elaborate monitoring approaches, such as Quantitative Temporal Bayesian Networks [70] or Constraint-Based Temporal Knowledge [13], both of which have successfully been used for activity monitoring.

8.3.5 Prediction of Likely Next Locations

In section 5.3.2, we predicted likely next locations of a person based on the results of our HHMM-based activity recognition, The evaluation in section 7.3 showed, that in our experiments, some locations could be predicted with very good accuracy due to similarities between different activities. While this enabled us to exploit such similarities to increase the accuracy of our predictions, similarities in the order of locations might not be given when modeling other activities. While we expect our activity recognition to perform better when detecting activities that share less similarities, we expect worse results when using activities that are dissimilar and consist of CDSRs that are hard to distinguish. In such a worst-

case scenario, activity recognition is likely not able to distinguish which activity is currently performed and a precise prediction for the next location of a person presumably cannot be generated. However, the weighted spatial model can still be used to obtain a set of possible next locations that is weighted by the probabilities of the person visiting them as we showed in 7.3.

Chapter 9

Conclusion

In this final chapter, we will first sum up the contents of this thesis and then provide one possible outlook into the future of autonomous service robots.

9.1 Summary

In this thesis, we illustrated techniques that enable service robots to become better aware of situations to ultimately enable adequate reactions in unexpected situations. Therefore, we proposed an expectations framework that can easily be integrated into existing approaches in reactive robot planning and enable a robot to estimate the normality of situations by the validation of various expectations. While such expectations can be used to model anything that is measurable, this thesis focused on expectations about human behavior.

We therefore showed how to automatically generate models of habitual human behavior from motion tracking data and a semantically annotated map of the environment. We demonstrated how such models can be used by a robot to perform activity recognition and showed that a low-cost sensor setting in a spatially limited environment can suffice to enable human activity recognition and the generation expectations about future locations of a person. Such expectations were used to estimate the normality of the situation by fully integrating human activity recognition and monitoring into the expectations framework. The components of the approach as well as the fully integrated system were evaluated in simulated as well as in real-world settings. The results indicate that the seamless integration of human activity models, low-cost activity recognition and our

proposed expectations framework into robot planning can enable a robot to detect situations that diverge from normality, thus making it better aware of the situation and its human partner.

The expectations framework is not limited to human behavior. On the contrary, its modular structure enables a robot to model and combine any kind of expectations and to obtain a general estimate about the normality of a situation. The approaches presented in this thesis constitute only a small, first step towards situation-aware, adaptive domestic service robots and hopefully will motivate researchers to make robots more situation-aware to enable user-friendly and adaptive robots in near future.

9.2 Related Visions for Robotics and AI

As for many technologies we use today, the ease of use seems to be a key-enabler when it comes to commercial success. Consider, for instance, the development of tablet computers which started in the 1980s. Since then, several devices were available for sale with one of the first ones being the EO Personal Communicator released in 1992. Many other manufacturers - including Microsoft, Apple, Atari and Palm - released tablet computers but commercial success failed to appear. It was until the release of the Apple iPad in 2010 that tablet computers were commercially successful and today, in 2014, they even fully replace traditional computers for many people. What were the reasons that the iPad is such a commercial success while other products failed completely? While some might say, Apples huge success was caused by releasing their product at just the right time, I think that there is another significant reason. In contrast to other manufacturers, Apple put a huge effort into their operating system making it easy to use for a wide variety of people, including some that otherwise rarely have a relation to technology. One key objective was to design a technical device that is extremely simple and intuitive and is consistent over several platforms (like the iPhone or the iPod). Most applications were very simple and intuitive to use and only needed a few settings that had to be made instead of complex setup routines that could only be handled by experts. One might say, the operating system and

the applications of the iPad were specifically designed towards the needs of their (non-expert) users.

Maybe, development in robotics can be seen in a similar fashion. So far, most robots, especially robots that are capable of complex tasks, can only be operated by people that had sophisticated training or a spend a huge amount of time working with the robot. But recent development in robotics aims towards robots that are easy to use and capable to perform complex tasks widely autonomously. This development seems to not only appear for end-users, but also for developers, which the example of the Robot Operating System (ROS)[1] illustrates. Many platforms in robotics were based on individual control- and operating systems, leading to developers re-implementing things that already existed in other platforms due to incompatibilities. This made the development and use of robots hard, even for the developers (who basically are users of their robots as well!). The PR2 robot was one of the first universal robotics platforms available that came with a common software and community. ROS was specifically designed to make components of the robot software reusable on a variety of robotic platforms, preventing developers to *reinvent the wheel* over and over again. It provided a common platform to be used on very different robotic systems like humanoids, autonomous cars and boats, quadrotors and robot arms. While those first improvements in the usability might be of more interest from a developers point of view, there are already the first commercial applications that focus on enabling a robot to learn new pick- and place actions trained from non-expert users by demonstration [67]. Robots that are equipped with such capabilities will lower the learning curve for non-expert users and hopefully make service robots usable for a wider range of people. As more and more people are able to use robots, sales will continue to increase. One example for this effect are vacuum robots that autonomously clean parts of the room only providing a very limited user interface that is understandable for everybody (mostly they offer only one ON/OFF button). Sales of such vacuuming robots make up a wide part of robot sales in general[2].

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

[2] http://www.worldrobotics.org/uploads/media/Executive_Summary_WR_2013.pdf

As robots enter human-inhabited domains and perform complex tasks in cooperation with humans, I think the usability of such robots will strongly depend on the ability to adequately react upon situations and especially consider human behavior. As the operating system of the iPad mentioned above, robots should be designed towards the needs of their users. A robot intended to work with a person will not be very user-friendly if it has to be told every detail in advance, it should rather learn from observations. To furthermore be able to work together with persons, the ability to consider human behavior is a prerequisite for a user-friendly robot and I think, without accounting for different situations and human behavior, personal robots cannot be made as user-friendly as they have to be to succeed as an effective commercial product. If robots will more and more be able to perform complex manipulation tasks while at the same time becoming more and more user-friendly, there might be a chance that robotics will experience a development similar to tablet computers in the last decades. If the user-friendliness will include persons and their behavior, maybe someday in near future, we will see domestic service robots to become a successful commercial product.

Appendix A

Appendix

A.1 Law of Total Probability

In probability theory, the law of total probability relates the total probability of the outcome of a random variable by the probabilities of each of its distinct events:

$$P(A) = \sum_{i=1}^{N} P(A \wedge B_i) = \sum_{i=1}^{N} P(A \mid B_i)P(B_i) \tag{A.1}$$

A.2 Markov Assumption

In Bayesian networks, the Markov Assumption states that the current state only depends on a finite fixed number of previous states. Bayesian networks that satisfy the Markov Assumption are called Markov Processes or Markov Chains.

A.3 Product Rule

The product rule follows from the definition of a conditional probability and is defined as follows:

$$P(A \wedge B) = P(A \mid B)P(B) = P(B \wedge A)P(A). \tag{A.2}$$

A.4 Bayes' Rule

The Bayes' Rule is defined as follows:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \alpha P(B \mid A)P(A). \tag{A.3}$$

We abbreviate $\frac{1}{P(B)}$ with the normalization constant α since it is independent of A and can $P(B)$ be computed using the law of total probability (see section A.1):

$$P(B) = \sum_{i=1}^{N} P(A_i \wedge B) = \sum P(B \mid A_i)P(A_i) \tag{A.4}$$

List of Prior Publications

The work presented in this document is partly based on prior publications. Sections of this work that drew upon content from prior publications cited the respective publications where appropriate. A complete list of publications that were (co-)authored during my research as a doctoral candidate is provided below.

Michael Karg and Alexandra Kirsch. Low Cost Activity Recognition Using Depth Cameras and Context Dependent Spatial Regions In *Proceedings of the Workshop on Autonomous Robots and Multirobot Systems (ARMS), in conjunction with the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014.

Michael Karg and Alexandra Kirsch. Low Cost Activity Recognition Using Depth Cameras and Context Dependent Spatial Regions In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Extended Abstract*, 2014.

Michael Karg and Alexandra Kirsch. A Human Morning Routine Dataset In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Extended Abstract*, 2014.

Michael Karg and Alexandra Kirsch. An Expectations Framework for Domestic Robot Assistants In *Proceedings of Conference on Advances in Cognitive Systems (ACS)*, 2013.

Michael Karg and Alexandra Kirsch. Situation Awareness for Goal-Directed Autonomy by Validating Expectations In *Proceedings of the Workshop on Goal Reasoning at the Conference on Advances in Cognitive Systems (ACS)*, 2013.

Michael Karg and Alexandra Kirsch. Simultaneous Plan Recognition and Monitoring (SPRAM) for Robot Assistants In *Proceedings of the Human Robot Collaboration Workshop 2013, in conjunction with RSS 2013*, 2013.

Severin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, Pierrick Koch, Charles Lesire, Gilberto Echeverria, Serge Stinckwich. Simulating complex robotic scenarios with MORSE In *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomouns Robots (SIMPAR)*, 2012.

Michael Karg, Alexandra Kirsch. Acquisition and Use of Transferable, Spatio-Temporal Plan Representations for Human-Robot Interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Christina Lichtenthaeler, Tamara Lorenz, Michael Karg, Alexandra Kirsch. Increasing Perceived Value Between Human and Robots - Measuring Legibility in Human Aware Navigation. In *Proceedings of the IEEE workshop on Advanced Robotics and its Social Impacts (ARSO)*, 2012.

S. Lemaignan, Echeverria G., M. Karg, M. Mainprice, A. Kirsch, R. Alami. Human-Robot Interaction in the MORSE Simulator. In *In Proceedings of the 2012 Human-Robot Interaction Conference (HRI), late breaking report*, 2012.

Michael Karg, Martin Sachenbacher, Alexandra Kirsch. Towards Expectation-based Failure Recognition for Human Robot Interaction. In *In Proceedings of the 22nd International Workshop on Principles of Diagnosis, Special Track on Open Problem Descriptions*, 2011.

List of Figures

Bibliography

[1] N. Akhtar and A. Kuestenmacher. Using naive physics for unknown external faults in robotics. In *22nd International Workshop on Principles of Diagnosis (DX-2011)*, volume 1, page 23, 2011.

[2] A. Aydemir, K. Sjoo, J. Folkesson, A. Pronobis, and P. Jensfelt. Search in the real world: Active visual object search based on spatial relations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2818–2824, May 2011.

[3] J. Bandouch and M. Beetz. Tracking humans interacting with the environment using efficient hierarchical sampling and layered observation models. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2040–2047, Sept 2009.

[4] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Rühr, and M. Tenorth. Robotic roommates making pancakes. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 529–536, Oct 2011.

[5] M. Beetz, L. Mosenlechner, and M. Tenorth. Cram - a cognitive robot abstract machine for everyday manipulation in human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1012–1017, Oct 2010.

[6] M. Beetz, L. Mösenlechner, M. Tenorth, and T. Rühr. Cram – a cognitive robot abstract machine. In *5th International Conference on Cognitive Systems (CogSys 2012)*, 2012.

[7] M. Beetz, M. Tenorth, D. Jain, and J. Bandouch. Towards automated models of activities of daily life. *Technology and Disability*, 22(1):27–40, 2010.

[8] M. Bennewitz, J. Pastrana, and W. Burgard. Active localization of people with a mobile robot based on learned motion behaviors. 2008.

[9] N. Blodow, L. C. Goron, Z.-C. Marton, D. Pangercic, T. Rühr, M. Tenorth, and M. Beetz. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, September, 25–30 2011. Accepted for publication.

[10] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall. Recognizing daily activities with rfid-based sensors. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, Ubicomp '09, pages 51–60, New York, NY, USA, 2009. ACM.

[11] H. H. Bui, D. Q. Phung, and S. Venkatesh. Hierarchical hidden markov models with general state hierarchy. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 324–329. AAAI Press, 2004.

[12] K. Buys, C. Cagniart, A. Baksheev, T. De Laet, J. De Schutter, and C. Pantofaru. An adaptable system for rgb-d based human body detection and pose estimation. *Journal of Visual Communication and Image Representation*, 25(1):39–52, Jan. 2014.

[13] A. Cesta, G. Cortellessa, R. Rasconi, F. Pecora, M. Scopelliti, and L. Tiberio. Monitoring elderly people with the robocare domestic environment: Interaction synthesis and user evaluation. *Computational Intelligence*, 27(1):60–82, 2011.

[14] G. Diego and K. Arras. Please do not disturb! minimum interference coverage for social robots. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1968–1973, Sept 2011.

[15] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Confer-*

ence on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.

[16] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular open robots simulation engine: Morse. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 46–51, May 2011.

[17] G. Echeverria, S. Lemaignan, A. Degroote, S. Lacroix, M. Karg, P. Koch, C. Lesire, and S. Stinckwich. Simulating complex robotic scenarios with morse. In I. Noda, N. Ando, D. Brugali, and J. Kuffner, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 7628 of *Lecture Notes in Computer Science*, pages 197–208. Springer Berlin Heidelberg, 2012.

[18] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

[19] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Mach. Learn.*, 32(1):41–62, July 1998.

[20] J. Fogarty, C. Au, and S. E. Hudson. Sensing from the basement: A feasibility study of unobtrusive and low-cost home activity recognition. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, pages 91–100, New York, NY, USA, 2006. ACM.

[21] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design patterns: Elements of reusable object-oriented software. 1995.

[22] G. Gigerenzer and W. Gaissmaier. Heuristic decision making. *Annual Review of Psychology*, 62(1):451–482, 2011. PMID: 21126183.

[23] M. Hanheide, A. Peters, and N. Bellotto. Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus. In *RO-MAN, 2012 IEEE*, pages 689–694, Sept 2012.

[24] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots*, 34(3):189–206, Apr. 2013.

[25] S. Intille, K. Larson, E. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson. Using a live-in laboratory for ubiquitous computing research. In K. Fishkin, B. Schiele, P. Nixon, and A. Quigley, editors, *Pervasive Computing*, volume 3968 of *Lecture Notes in Computer Science*, pages 349–365. Springer Berlin Heidelberg, 2006.

[26] D. Joho and W. Burgard. Searching for objects: Combining multiple cues to object locations using a maximum entropy model. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 723–728, May 2010.

[27] M. Karg and A. Kirsch. Acquisition and use of transferable, spatio-temporal plan representations for human-robot interaction. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5220–5226, Oct 2012.

[28] M. Karg and A. Kirsch. An Expectations Framework for Domestic Robot Assistants. In *Conference on Advances in Cognitive Systems (ACS)*, 2013.

[29] M. Karg and A. Kirsch. Simultaneous Plan Recognition and Monitoring (SPRAM) for Robot Assistants. In *Proceedings of Human Robot Collaboration Workshop at Robotics Science and Systems Conference (RSS) 2013*, 2013.

[30] M. Karg and A. Kirsch. Situation Awareness for Goal-Directed Autonomy by Validating Expectations. In *Proceedings of the Workshop on Goal Reasoning at the Conference on Advances in Cognitive Systems (ACS)*, 2013.

[31] M. Karg and A. Kirsch. A human morning routine dataset. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Extended Abstract*, 2014.

[32] M. Karg and A. Kirsch. Low cost activity recognition using depth cameras and context dependent spatial regions. In *Workshop on Autonomous Robots and Multirobot Systems (ARMS), Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2014.

[33] M. Karg and A. Kirsch. Low cost activity recognition using depth cameras and context dependent spatial regions. In *Proceedings of the Ninth Inter-*

national Conference on Autonomous Agents and Multiagent Systems (AAMAS), Extended Abstract, 2014.

[34] M. Karg, M. Sachenbacher, and A. Kirsch. Towards expectation-based failure recognition for human robot interaction. In *22nd International Workshop on Principles of Diagnosis, Special Track on Open Problem Descriptions*, 2011.

[35] T. Kelly, Y. Wang, S. Lafortune, and M. Welsh. A formal foundation for failure avoidance and diagnosis. Technical Report HPL-2009-203, HP Laboratories, 2009.

[36] W. G. Kennedy, M. D. Bugajska, M. Marge, W. Adams, B. R. Fransen, D. Perzanowski, A. C. Schultz, and J. G. Trafton. Spatial representation and reasoning for human-robot collaboration. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1554–1559. AAAI Press, 2007.

[37] M. Klenk, N. Hawes, and K. Lockwood. Representing and reasoning about spatial regions defined by context. In *AAAI Fall Symposium Series*, 2011.

[38] T. Kruse, A. Kirsch, E. Sisbot, and R. Alami. Exploiting human cooperation in human-centered robot navigation. In *RO-MAN, 2010 IEEE*, pages 192–197, Sept 2010.

[39] L. Kuhn, B. Price, J. de Kleer, M. Do, and R. Zhou. Pervasive diagnosis: Integration of active diagnosis into production plans. In *proceedings of AAAI*, 2008.

[40] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz. Simulation-based temporal projection of everyday robot object manipulation. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, pages 107–114, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[41] L. Kunze, M. Tenorth, and M. Beetz. Putting people's common sense into knowledge bases of household robots. In *Proceedings of the 33rd Annual German Conference on Advances in Artificial Intelligence*, KI'10, pages 151–159, Berlin, Heidelberg, 2010. Springer-Verlag.

[42] L. Kunze, M. Tenorth, and M. Beetz. Putting people's common sense into knowledge bases of household robots. In *Proceedings of the 33rd Annual German Conference on Advances in Artificial Intelligence*, KI'10, pages 151–159. Springer-Verlag, Berlin, Heidelberg, 2010.

[43] U. Kurup, C. Lebiere, A. T. Stentz, and M. Hebert. Using expectations to drive cognitive behavior. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, July 2012.

[44] S. Lemaignan, G. Echeverria, M. Karg, J. Mainprice, A. Kirsch, and R. Alami. Human-robot interaction in the morse simulator. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 181–182, March 2012.

[45] M. Leo, P. Spagnolo, T. DÓrazio, and A. Distante. Human activity recognition in archaeological sites by hidden markov models. In *Proceedings of the 5th Pacific Rim Conference on Advances in Multimedia Information Processing - Volume Part II*, PCM'04, pages 1019–1026. Springer-Verlag, Berlin, Heidelberg, 2004.

[46] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *Int. J. Rob. Res.*, 26(1):119–134, 2007.

[47] C. Lichtenthaler, T. Lorenz, M. Karg, and A. Kirsch. Increasing perceived value between human and robots - measuring legibility in human aware navigation. In *Advanced Robotics and its Social Impacts (ARSO), 2012 IEEE Workshop on*, pages 89–94, May 2012.

[48] K. Lockwood, J. D. Kelleher, M. Klenk, and N. Hawes. The role of context in spatial region identification. In *Proceedings of Second Annual Conference on Advances in Cognitive Systems (Poster Collection)*, Baltimore, MD, 2013.

[49] M. Luber, G. Diego Tipaldi, and K. O. Arras. Place-dependent people tracking. *Int. J. Rob. Res.*, 30(3):280–293, Mar. 2011.

[50] W. Maier and E. Steinbach. A probabilistic appearance representation and its application to surprise detection in cognitive robots. *Autonomous Mental*

Development, IEEE Transactions on, 2(4):267–281, Dec 2010.

[51] D. McDermott. A reactive plan language. Technical report, Citeseer, 1991.

[52] G. J. McLachlan and T. Krishnan. *Basic Theory of the EM Algorithm*. John Wiley and Sons, Inc., 2007.

[53] D. Minnen, I. Essa, and T. Starner. Expectation grammars: leveraging high-level expectations for activity recognition. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–626–II–632 vol.2, June 2003.

[54] T. Mitzner, C. Smarr, J. Beer, T. Chen, J. Springman, A. Prakash, C. Kemp, and W. Rogers. Older adults' acceptance of assistive robots for the home. 2011.

[55] M. Molineaux, M. Klenk, and D. Aha. Goal-driven autonomy in a navy strategy simulation. In *AAAI Conference on Artificial Intelligence*, 2010.

[56] M. Molineaux, M. Klenk, and D. W. Aha. Goal-driven autonomy in a navy strategy simulation. to appear. 2010.

[57] T. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6):47–60, Nov 1996.

[58] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121, Mar 1985.

[59] L. Mosenlechner and M. Beetz. Fast temporal projection using accurate physics-based geometric reasoning. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1821–1827, May 2013.

[60] N. Nguyen, H. Bui, S. Venkatsh, and G. West. Recognizing and monitoring high-level behaviors in complex spatial environments. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–620–5 vol.2, June 2003.

[61] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 955–960, Washington, DC, USA, 2005. IEEE Computer Society.

[62] D. Norman. *The design of future things. 2007*. New York: Basic Books, 2007.

[63] J. Orkin and D. Roy. The restaurant game: Learning social behavior and language from thousands of players online. *Journal of Game Development (JOGD)*, 3(1):39–60, December 2007.

[64] A. Ortony and D. Partridge. Surprisingness and expectation failure: what's the difference? In *Proceedings of the 10th international joint conference on Artificial intelligence - Volume 1*, IJCAI'87, pages 106–108, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[65] D. Pangercic, B. Pitzer, M. Tenorth, and M. Beetz. Semantic object maps for robotic housework - representation, acquisition and use. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4644–4651, Oct 2012.

[66] D. Pangercic, M. Tenorth, D. Jain, and M. Beetz. Combining perception and knowledge processing for everyday manipulation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1065–1071, Oct 2010.

[67] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal. Skill learning and task outcome prediction for manipulation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3828–3834, May 2011.

[68] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson. Mining models of human activities from the web. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 573–582, New York, NY, USA, 2004. ACM.

[69] M. Piunti, C. Castelfranchi, and R. Falcone. Expectations driven approach for situated, goal-directed agents. In *WOA*, pages 104–111. Citeseer, 2007.

[70] M. E. Pollack, L. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos. Autominder: An intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems*, 44:273–282, 2003.

[71] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[72] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[73] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[74] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995.

[75] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Min. Knowl. Discov.*, 2(2):169–194, June 1998.

[76] M. Schuster, D. Jain, M. Tenorth, and M. Beetz. Learning organizational principles in human environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3867–3874, May 2012.

[77] G. Steinbauer and F. Wotawa. Detecting and locating faults in the control software of autonomous mobile robots. In *19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1742–1743. Citeseer, 2005.

[78] P. Struss. Model-based problem solving. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, pages 395–465. Elsevier, 2008.

[79] F. Stulp, A. Fedrizzi, and M. Beetz. Action-related place-based mobile manipulation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3115–3120, Oct 2009.

[80] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from rgbd images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 842–849, May 2012.

[81] A. Tapus, M. Mataric, and B. Scasselati. Socially assistive robotics [grand challenges of robotics]. *Robotics Automation Magazine, IEEE*, 14(1):35–42, March 2007.

[82] M. Tenorth, J. Bandouch, and M. Beetz. The tum kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1089–1096, Sept 2009.

[83] M. Tenorth and M. Beetz. Knowrob - knowledge processing for autonomous personal robots. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4261–4266, Oct 2009.

[84] M. Tenorth and M. Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. *International Journal of Robotics Research (IJRR)*, 32(5):566 – 590, April 2013.

[85] M. Tenorth, L. Kunze, D. Jain, and M. Beetz. Knowrob-map - knowledge-linked semantic object maps. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 430–435, Dec 2010.

[86] G. D. Tipaldi and K. O. Arras. I want my coffee hot! learning to find people under spatio-temporal constraints. In *ICRA*, pages 1217–1222. IEEE, 2011.

[87] D. Townsend and T. Bever. *Sentence comprehension: The integration of habits and rules*. MIT Press, Cambridge, MA, USA, 2001.

[88] T. T. Truyen, D. Q. Phung, H. H. Bui, and S. Venkatesh. Hierarchical Semi-Markov Conditional Random Fields for Recursive Sequential Data. *ArXiv e-prints*, Sept. 2010.

[89] R. Ueda, T. Ogura, K. Okada, and M. Inaba. Design and implementation of humanoid programming system powered by deformable objects simulation. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems*, pages 374–381, 2008.

[90] T. van Kasteren, A. Noulas, and B. Kröse. Conditional random fields versus hidden markov models for activity recognition in temporal sensor data. Advanced School for Computing and Imaging (ASCI), 2008.

[91] M. Weiland, A. Smaill, and P. Nelson. Learning musical pitch structures with hierarchical hidden markov models. *Journees d'Informatique Musical*, 2005.

[92] B. C. Williams, M. D. Ingham, S. Chung, P. Elliott, M. Hofbaur, and G. T. Sullivan. Model-based programming of fault-aware systems. *AI Mag.*, 24(4):61–75, Jan. 2004.

[93] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[94] S. Woods, M. Walters, K. Koay, and K. Dautenhahn. Comparing human robot interaction scenarios using live and video based methods: towards a novel methodological approach. In *Advanced Motion Control, 2006. 9th IEEE International Workshop on*, pages 750–755, 2006.

[95] L. Yujian and L. Bo. A normalized levenshtein distance metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1091–1095, June 2007.

[96] H. Zender, O. Martínez Mozos, P. Jensfelt, G. J. M. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robot. Auton. Syst.*, 56(6):493–502, June 2008.